CS 245

Logic and Computation

Winter 2017

Instructor

Richard Trefler

office: DC 2336

office hours: after class

Text books:

Mathematical Logic
For Computer Science
Second Edition
Lu Zhongwan

Logic in Computer Science: Modelling and Reasoning about Systems second edition

Michael Huth and Mark Ryan

Tutorials:

F	3:30 – 4:20	MC 4058
	8:30 – 9:20	MC 4060
	10:30 – 11:20	MC 4060
	11:30 – 12:20	MC 4042

Mid term Test

Thursday, Feb. 16th

4:30 - 6:20

Rough outline of topics:

- Syntax of propositional logic
- Semantics of propositional logic

- Reasoning using propositional logic

- Introduction to predicate logic

- Syntax of predicate logic

- Semantics of predicate logic (first order logic)
- Reasoning using predicate logic
- Logic for reasoning about program behavior

- Undecidable problems in program analysis

Mathematical logic has spread throughout the field of Computer Science

Concepts and methods of (mathematical) logic are of such utility in Computer Science ...

that logic has been called "the calculus of computer science."

The intended analogy is that logic is as important to computer science as mathematics is to physics and understanding the laws of nature.

In particular logic plays key roles in the development of

- the theory of computational complexity
- the use of first order logic as a data base query language

- the role of type theory in programming languages

- reasoning about knowledge in multi-agent systems

- automated (and semi-automated) program analysis engines for design verification

For example, in first order logic we can write a formula saying that every node in a graph has exactly two edges leaving the node.

Using second order logic one can write a formula expressing that a graph is three-colorable. That is, every node has one of three colors and no two adjacent nodes have the same color.

Further more, one can relate the computational complexity of certain classes of problems to the ability to express the problems in a variant of first order logic.

For instance, this may shed light on whether the class of problems solvable in polynomial time by a deterministic computer is equal to the class of problems solvable in polynomial time by a non-deterministic computer ($P \stackrel{?}{=} NP$).

There is also a strong connection between logic and database systems.

Relational databases can be understood as a finite structure described using first order logic formulae.

These logical formulae form the basic elements used in structured Query Language (SQL) and Query By Example (QBE).

Then the first order logic descriptions can be effectively implemented in relational algebra engines.

Over several years type theory has emerged as a unifying frame work for the design, analysis and implementation of programming languages.

In these contexts, type theory helps clarify concepts such as data abstraction, polymorphism, and inheritance.

Type theory itself grew out of ideas that joined together from mathematical and philosophical logic in combination with computer science.

As a final example, consider the influence of logic on the (automated) analysis and verification of computer designs and computer programs.

Significant parts of mathematical logic were developed in an attempt to formalize mathematical reasoning.

David Hilbert's Program included as a goal the formalization of all of mathematics in a system that would be complete and decidable.

Very roughly, completeness of the system would mean that all true mathematical statements could be written in the system.

That the system was also to be decidable meant that there should be a mechanical set of rules to determine the truth of any given statement.

However, kurt Gödel showed that the standard first order axioms of mathematics were incomplete.

Furthermore, Alan Turing, Alonzo Church and Alfred Tarski showed that first order logic was undecidable.

In contrast, formal verification of computer programs and designs was shown to be decidable. Key aspects of this work are finite state models of program semantics and

specifications of program behavior written in a propositional temporal logic.

Given the sheer size and complexity of modern hardware, software and embedded systems, automated reasoning engines are of critical importance in providing assurence that systems behave as intended.

Automated analysis tools are now used and developed at NASA, Intel, IBM, Siemens, Fujitsu, etc.

Martin Davis, in 1988 wrote "the connections between logic and computers are a matter of engineering practice at every level of computer organization."

"Overall, logic provides computer science with both a unifying foundational framework and a powerful tool for modeling and reasoning about aspects of computing."

The previous quote and the source material for these notes "On the Unusual Effectiveness of Logic in Computer Science"
J. Halpern, R. Harper, N. Immerman, P. Kolaitis and M. Vardi:

logic : reasoning conducted or assessed according to strict principles of validity.

: the quality of being justifiable by reason

logic: a system or set of principles underlying the arrangement of elements in a computer or electronic device so as to perform a specific task.

Definition taken from Google search

Example: If the train arrives late and there are no taxis at the station, then John is late for his meeting. John is not late for his meeting. The train did arrive late. Therefore, there were taxis at the station. (From H&R pg 1)

Is the argument correct? Valid? Why? What makes it a correct (or incorrect argument)?

Another example: If it is raining and Jane does not have her umbrella with her, then she will get wet. Jane is not wet. It is raining. Therefore, Jane has her umbrella with her.

(H&R, pg 2)

Is this a valid argument? Why? Why not?

What are the similarities between the two arguments?

Do they have a common structure?

Consider the first argument.

Let p represent
the train arrives late

Let q represent
there are taxis at the station

r represents John is late

Then the entire argument can be written

If p and not q then r.
Not r. p. Therefore q.

(H&R, pg 2)

In a similar way if we represent the 2nd argument by

p - it is raining

q - Jane has her umbrella

r - Jane gets wet.

If p and not q then r.
Not r. p. Therefore q.

Another example.
Every middle school student plays tennis. Z does not play tennis. Therefore Z is not a middle school student.

(Z, pg 5).

Is this argument correct?
Notice the quantification over middle school students and the comparison with the element z.

There is something very similar about the structure of the three arguments.

Consider, however, whether the arguments are 'true.'

In particular, we know very little about 'all middle school students' or which set of students are being compared.

Rather, the truth of the argument depends on whether the conclusion follows under the assumption that the premises hold.

In the example arguments, the premises and conclusions are propositions. That is, statements, each one of which may be 'true' or 'false.'

Not all sentences of English can be easily translated into propositions.

For instance, questions, imperatives (orders), ambiguous statements, etc., are all difficult to translate into propositrons.

Even for declarative statements translation can be tricky.

For example "there are no taxis at the station" is translated as "not q" with q saying "there are taxis at the station."

The question is then could we translate "there are no taxis at the station" into ؋? Which translation is better?

Goal: systematic study of rigorous arguments with in the context of computer science.

To begin, we should carefully spell out what are the "statements" of a logic and what are the relationships between the statements.

For instance, we may use logic connectives to represent words such as "and," "or," "not," "implies," "if and only if," and so on.

Next topic:

syntax and semantics of propositional logic.

Readings: Chap 1 and 2 of Zhongwan