Lecture 1

History of Complexity and Logic

January 6, 1998
Lecturer: Richard Ladner
Notes: Vass Litvinov

1.1. Administrativia

"Complexity and Logic"
Richard Ladner
ladner@cs, Sieg 311, 543-9347
Office hours: whenever the door is open/by appointment

Text: *Computational Complexity* by Papadimitriou (required)
Read Chapters 1, 2, 3, 7

Grades based on:

- problem sets
- class notes
- oral presentation (Finals week). Options:
  - critical analysis of research papers
  - critical analysis of a working verification system
- class participation

Topics

- Logic
  - Propositional Logic
  - First Order Logic
  - Temporal Logic (has to do with programs)
  - Logic of Knowledge (has to do with distributed programs)
- Complexity

- Complexity Classes
- Alternating Computation
- Interactive Proof Systems
- Undecidability of Logics
- Complexity of Logics

1.2. Connections Between Complexity and Logics

1900s **Hilbert** proposed that mathematics have a *logical foundation*, a formalism behind it. He believed that within the formalism every true statement could be "determined." So called *"Hilbert's Plan."*

1910s **Russell** and **Whitehead** published "Principia Mathematica," the first attempt to determine the truth of all mathematical formulas.

1920s **Presburger** proved that every true statement in arithmetic with addition only could be determined.

1930s **Gödel** showed that the class of consequences of a set of axioms for arithmetic with + and * was undecidable.

Turing and **Church** and others proposed formal models of computation (*Turing Machine, Lambda Calculus, ...*). They all turned to be equivalent. *Universal Turing Machine* was conceived. Halting problem also turned to be undecidable. *Turing and Church Thesis* became accepted.

1940s **Post** defined formally the notion of *reducibility* between problems.

Many problems were shown to be undecidable using reducibility arguments.

Mathematical logic flourished.

Working computers were born in wake of WWII.

1950s **Rabin** and **Scott** built the theory of *finite automata* (nondeterministic). Most problems about them were shown decidable.

This marked the birth of theoretical computer science.

1960s **Hartmanis** and **Stearns** invented modern *computational complexity* based on the Turing Machine. Time and space complexity became standards.

Hoare and **Floyd** proposed *formal verification* of programs based on logical reasoning. (Like the "Hilbert's Plan.")

1970s **Cook** (and Levin?) invented *NP-completeness.* SAT was shown to be NP-complete. **Karp** and **Garey** and **Johnson** helped make NP-completeness proofs an industry. Almost all proofs were reductions.

Complexity theorists ignored program verification with some good reason. **Meyer** and **Stockmeyer** and others showed exponential and worse lower bounds on logical theories.

Stockmeyer and **Kozen** invented *Alternating Turing Machines.*

Second half of 1970s: formal verification was taken seriously. **Pnueli** applied temporal logic to program verification. **Pratt** invented *dynamic logic*, the right way to look at Hoare's logic.

1980s Complexity theory became a standard part of Computer Science undergraduate curriculum.

Difficult proofs for weak lower bounds became popular.

Goldwasser and **Mihali** and **Rackoff** invented interactive proof systems as a basis for cryptography. Randomizization became popular.

Parallel complelxity became a rage, technology-driven.

Some theorists continued to work on logic in spite of being ignored by complexity theorists.

1990s Trends from 1980s continue, with some changes.

Approximation algorithms and lower bounds for NP-complete problems became popular. Complexity theory is full blown, but the results became more diffuse, more technical; no breakthroughs in the grand approach.

Practical verification systems begin to emerge. VIS, FormalCheck, SPIN, SMV, others. Mostly verification of hardware; complexity theory is ignored, but some of the systems adopted ideas from the logic area.

Other Connections between Logic and Computer Science

- LISP - functional programming evolved from λ-calculus
- Type theory - meta-logic of programs
- Prolog - logic programming
- Theorem proving, symbolic computation
- Hardware design
- Relational databases
- Semantics of programming languages
- AI