Harvey Mudd College

Computer Science 81

Computability and Logic

Fall 2006

Trailer:
An understanding of the fundamental theoretical concepts of computability will give you perspectives on problem solving that would be difficult to acquire otherwise. (Applications range from the practice of software engineering and compiler construction to practical aspects such as scoring high on the GRE.) The link between computation and mathematical logic goes back to explorations of formalizing and mechanizing mathematical proofs. The two subjects are intricately interwoven: computation can be represented in logic and logic can be automated as computation. Moreover, each admits forms of self-application, which leads to some very interesting insights on the boundary between computability and incomputability.

Instructor:

Professor Bob Keller, x 18483, 1253 Olin, keller at cs.hmc.edu

Grader/Tutors:

Mike Buchanan
Faith Dang
Phil Miller

HMC Catalog Description:

An introduction to some of the mathematical foundations of computer science, particularly logic, automata, and computability theory. Develops skill in constructing and writing proofs, and demonstrates the applications of the aforementioned areas to problems of practical significance. Prerequisites: Math 55, Computer Science 60. 3 credit hours. (Both semesters.) [For Pomona College students, CS 55 substitutes for Math 55.]

Grading:

Homework 60%
Exams 40% (1 midterm, 1 comprehensive final)

Collaboration Policy:

No collaboration, unless indicated by the instructor.

If in doubt, please ask first, rather than act and have to pay the consequences.

Help is gladly provided by the instructor and tutors. Please take advantage of office hours.

Exam Policy:

Closed book, with a single two-sided crib sheet allowed.

Late Policy:

No late work accepted, unless for a reason approved by the instructor.

Texts:

K: Dexter C. Kozen, ***Automata and Computability***, Springer, 1997, ISBN 0-387-94907-0

D: Dirk van Dalen, ***Logic and Structure***, Fourth Edition, Spring 2004, ISBN 3-540-20879-8

Plan of Attack:

We will first examine **finite-state automata** (computing machines) and their relationship to languages and **regular expressions**. (This will be review for most of you, so we'll move pretty fast here.) We'll look at the applications and limitations of finite-state machines, then move on to the more powerful **context-free languages** and the corresponding machines: **pushdown automata**.

Next we'll turn to logic, where there are parallels with the above material in the form of **truth** vs. **proof**. There are two levels of interest: structuring proofs and proving things about proofs, so-called **meta-mathematics**. We'll gain some experience in doing proofs in a style known as **natural deduction**, both for propositional and predicate logic. This will be helpful as a way of outlining proofs in the

course, and for the rest of our careers. We will also look at methods for **mechanized proof**.

Next we'll look at **Turing machines**, which are the most powerful, defining **computability**, in some sense. We'll discover what kind of **grammars** are equivalent to Turing machines, and along the way look at the special subset of context sensitive grammars and their specialized Turing machines. We will demonstrate the existence of **incomputable** or **unsolvable** problems, and show how the concept of **reduction** can be used to prove that a problem is unsolvable.

Finally, we show parallels to the above limitations in predicate logic, culminating with Gödel's famous **incompleteness theorem**.

The outline below is tentative and approximate. This is the first time I have used either of these texts. The topics won't always align to days in the way I've shown, and we may have to drop certain topics, depending on how things are going at the time.

CS 81 Outline (Fall 2006)

Order	Topics	Reading
1	Strings and Regular Languages, Finite-State Automata, Nondeterministic Finite-State Automata	K: Lec. 1-5
2	Subset construction, Pattern Matching, Regular Expressions	K: Lec.6-10
3	Limitations of Finite Automata, Pumping Lemma, State Minimization	K: Lec.11-14
4	Myhill-Nerode Relations, Two-Way Finite Automata	K: Lec.15-17
5	Context-Free Grammars and Languages, Normal-Forms	K: Lec.19-21
6	Pumping Lemma for Context-Free Languages, Pushdown Automata	K: Lec.22-25
7	Parsing with Pushdown Automata, Cocke-Younger-Kasami Algorithm	K: Lec.26-27
8	Proposition logic	D: Sec. 1.1-1.3
9	Natural deduction	D: Sec. 1.4
10	Completeness	D: Sec. 1.5

11	The Missing Connectives	D: Sec. 1.6
12	Tableau (tree) method	Supplement
	The mid-term exam will be about here.	
13	Predicate Logic	D: Sec. 2.1-2.3
14	Predicate Logic Semantics	D: Sec. 2.4
15	Predicate Logic Properties	D: Sec. 2.5
16	Identity	D: Sec. 2.6
17	Examples from algebraic structures	D: Sec. 2.7
18	Natural Deduction	D: Sec. 2.8
19	Adding the Existential Quantifier	D: Sec. 2.9
20	Natural Deduction and Identity	D: Sec. 2.10
21	Natural Deduction and Inductive Proofs	Supplement
22	Application to Program Correctness	Supplement
23	Completeness for Predicate Logic	D: 3.1
24	Tableau (tree) method	Supplement
25	Automated Reasoning, Resolution Method	Supplement
26	Turing Machines and Effective Computability, Church-Turing Thesis, Equivalent Models	K: Lec. 28-30
27	Universal Machines and Diagonalization, Decidable and Undecidable Problems, Reduction	K: Lec. 31-33
28	Rice's Theorem	K: Lec. 34
29	Type 0 Grammars, Partial Recursive Functions	K: Lec. 36
30	Incompleteness Theorem	K: Lec. 38-39, D: Sec, 7.1-7.7
	The final exam will be about here.	