Harvey Mudd College

Computer Science 81

Computability and Logic

Spring 2007

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
Office hours: Mon 3-5:30 pm, Tue 7:30-10 pm, or by drop-in

Grader/Tutors (hours will be posted):

Sarah Fletcher
Kevin Oelze
Steven Smith
Faith Dang

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 (or Pomona CS 55), Computer Science 60 (or Pomona CS 52). 3 credit hours. (Both semesters.)

Grading:

Homework 50%
Exams 40% (1 midterm, 1 comprehensive final)
In-Class participation 10%

Homework submisisons:

Homework can be hand-written or typed. If you elect the former, make sure your exposition is readable and not crammed into too little space. Graders can deduct for lack of readability.

Collaboration Policy (on a trial basis):

You may discuss problems with others. You may get help from the professor and tutors, who are happy to render it. However, the solution write-ups must be your own, in your own words. You may not look at or transcribe written material from others to enable writing your own solution. In other words, the solution you submit is based on your own assimilated knowledge. If there are any doubts as to interpretation of this policy, please ask before you act.

Participation Policy:

You are expected to come to class, arriving on time. You should ask questions. Laptop use that does not relate to class work is prohibited.

Exam Policy:

Closed book, often with a single two-sided crib sheet allowed. On the final, there may be a portion that is strictly closed book.

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

Approach and Overview:

We will first examine **finite-state automata** (computing machines) and their relationship to languages and **regular expressions**. Because you've seen it CS 60, 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 concepts in the form of **truth** (languages) vs. **proof** (machines and grammars). There are two levels of interest: structuring proofs and proving things *about* proofs, so-called **metamathematics**. 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 more mechanized methods for proof, such as **semantic tableaux** and **resolution**.

Next we'll look at **Turing machines**, which are the most powerful, defining effective **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**, which shows the limitations of proof systems similar to the limitations of computability.

The outline below is approximate.

CS 81 Outline (Spring 2007) Subject to Revision

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 from Automata	K: Lec.6-10

3	Limitations of Finite Automata, Pumping Lemma, State Minimization	K: Lec.11-14
4	Abstract States of a Language, 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	Semantic tableaux (tree) method for proposition logic	Supplement
10	Natural deduction for proposition logic	D: Sec. 1.4, 1.6
11	Completeness for proposition logic **The mid-term exam will be about here.**	D: Sec. 1.5
12	Predicate Logic	D: Sec. 2.1-2.3
13	Predicate Logic Semantics	D: Sec. 2.4
14	Predicate Logic Properties	D: Sec. 2.5
15	Identity, Examples from algebraic structures	D: Sec. 2.6, 2.7
16	Semantic tableaux (tree) method for predicate logic	Supplement
17	Natural Deduction for predicate logic	D: Sec. 2.8
18	Adding the Existential Quantifier and Identity	D: Sec. 2.9, 2.10
19	Natural Deduction and Inductive Proofs	Supplement
20	Application to Program Correctness	Supplement
21	Completeness for Predicate Logic	D: 3.1
22-23	Automated Reasoning, Resolution Method	Supplement
24	Turing Machines and Effective Computability, Church-Turing Thesis, Equivalent Models	K: Lec. 28-30
25	Universal Machines and Diagonalization, Decidable and Undecidable Problems, Reduction	K: Lec. 31-33
26	Rice's Theorem	K: Lec. 34
27	Type 0 Grammars, Partial Recursive Functions	K: Lec. 36
28	Incompleteness Theorem	K: Lec. 38-39, D: Sec, 7.1-7.7
	The final exam will be about here.	