The University of Georgia

Department of Computer Science

Course Information Sheet

CSCI 2670

Introduction to Theory of Computing

Brief Course Description
(50-words or less)

This is a first course on the theory of computing. Fundamental Topics include finite automata, regular expressions and languages, context-free grammars and languages, push-down automata, pumping lemmas for regular languages and for context-free grammars, the Chomsky hierarchy of language classes, Turing machines and computability, undecidability of the halting problem, reducibilities among decision problems and languages, time complexity, and NP-completeness and tractability. *Please note, I removed space complexity from the CAPA description. I have never been able to cover that material … I struggle to get to NP-completeness.*

Extended Course Description / Comments

This is a required course for all computer science majors. It is open to any students interested in learning the underlying mathematical models of computation (provided they have the necessary background knowledge).

Pre-Requisites and/or Co-Requisites

CSCI/MATH 2610: Discrete Mathematics or CSCI 2611 Discrete Mathematics for Engineers

Approved Textbooks

Thomas A. Sudkamp
Languages and Machines: An Introduction to the Theory of Computer Science
3rd Edition
ISBN-13: 978-0321322210

Michael Sipser
Introduction to the Theory of Computation
2nd Edition
ISBN-13: 978-0534950972

Learning Outcomes

This course introduces fundamental models relevant to most areas of computer science. At the end of the semester, all students will be able to do the following:

1. Given an NFA M, create a DFA or a regular expression that accepts L(M).
2. Given a regular language L, create an NFA that accepts L.
3. Use pumping lemmas to prove a language is not regular or not context-free.
4. Given a description of a context-free language L, develop a context-free grammar (CFG) G such that L(G) = L.
5. Convert a CFG into Chomsky Normal Form (CNF).
6. Given a context-free grammar G in CNF and a string w, use the CYK algorithm to determine if G generates w.
7. Given an context-free grammar G, create a push-down automaton (PDA) that accepts L(G).
8. Identify if a given language is regular, context-free but not regular, or neither.
9. Given a language L, create a Turing machine L that accepts L.
10. Convert between different variations of the Turing machine model (e.g., multi-tape to single tape).
11. Create a Turing machine that performs a function.

12. Define decidabilty and demonstrate that a language is decidable.
13. Reduce one problem to another one.
14. Use reductions to prove a problem is undecidable.
15. Define P, NP and NP-complete.
16. Show a problem is in P and determine its computational complexity.
17. Write pseudo-code describing a non-deterministic Turing machines steps to solve a problem.
18. Prove a problem is NP-complete.

Relationship Between Student Outcomes and Learning Outcomes

		Student Outcomes										
		A	b	c	d	e	f	g	h	i	j	k
Learning Outcomes	1	•									•	
	2	•									•	
	3	•									•	
	4	•									•	
	5	•										
	6	•									•	
	7	•									•	
	8	•									•	
	9	•									•	
	10	•										
	11	•										
	12	•	•								•	
	13	•										
	14	•										
	15	•	•								•	
	16	•	•								•	
	17	•	•								•	
	18	•	•								•	

Student Outcomes

a. An ability to apply knowledge of computing and mathematics appropriate to the discipline.
b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
c. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
d. An ability to function effectively on teams to accomplish a common goal.
e. An understanding of professional, ethical, legal, security and social issues and responsibilities.
f. An ability to communicate effectively with a range of audiences.
g. An ability to analyze the local and global impact of computing on individuals, organizations, and society.
h. Recognition of the need for and an ability to engage in continuing professional development.
i. An ability to use current techniques, skills, and tools necessary for computing practice.
j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
k. An ability to apply design and development principles in the construction of software systems of varying complexity.

Major Topics Covered (Approximate Course Hours)

Mathematical preliminaries (2-hours)
Languages and regular expressions (2-hours)
Context-free grammars (3-hours)
Chomsky normal form and the CYK algorithm (2-hours)
DFAs and NFAs (6-hours)
Equivalence of finite automata and regular expressions (2-hours)
Pumping lemma for regular languages (2-hours)
DFA state minimization (1-hour)
PDAs (2-hours)
Pumping lemma for context-free grammars (2-hours)
Turing machines (2-hours)
Turing computable functions (2.5-hours)
The Chomsky hierarchy (0.5-hours)
Decidability and problem reductions (3-hours)
Undecidability and the halting problem (5-hours)
Time complexity (1-hour)
P, NP and Cook's theorem (3.5-hours)
NP-completeness (5-hours)

Assessment Plan for this Course

Each time this course is offered, the class is initially informed of the Course Outcomes listed in this document, and they are included in the syllabus. At the end of the semester, an anonymous survey is administered to the class where each student is asked to rate how well the outcome was achieved. The choices provided use a 5-point Likert scale containing the following options: Strongly agree, Agree, Neither agree or disagree, disagree, and strongly disagree. The results of the anonymous survey are tabulated and results returned to the instructor of the course.

The course instructor takes the results of the survey, combined with sample student responses to homework and final exam questions corresponding to course outcomes, and reports these results to the ABET committee. If necessary, the instructor also writes a recommendation to the ABET committee for better achieving the course outcomes the next time the course is offered.

Course Master Course History

X/X/2012 Course approved by CSCI Curriculum Committee
X/X/2012 Course approved by Computer Engineering Faculty
X/Y/2012 Course approved by CSCI Department Faculty