CSCE 355: Foundations of Computation

1. Course number and name: CSCE 355: Foundations of Computation

2. Credit: 3-hrs; Contact: 3 lectures of 50 minutes each or 2 lectures of 75 minutes each per week

3. Instructor: Fall 2010: Stephen Fenner

4. Text book: J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley-Pearson 2007.

5. Specific course information
   a. Catalog description: Basic theoretical principles of computing as modeled by formal languages, grammars, automata, and Turing machines; fundamental limits of computation.
   b. Prerequisites: CSCE 211, 212, and 350, MATH 374
   c. Required in CS curricula

6. Specific goals for the course
   a. Specific outcomes of instruction:
      1. Prove theorems in discrete math by induction, contradiction, or cases
      2. Analyze, design, and manipulate finite state acceptors
      3. Design and manipulate regular expressions
      4. Prove languages not regular or context-free
      5. Design and analyze context-free grammars and push-down automata
      6. Analyze and simulate a Turing machine
      7. Prove problems undecidable via reduction
   b. Relation of course outcomes to Student Outcomes: CE: see page 2; CS & CIS: see page 3

7. Topics covered and approximate weight (14 weeks, 3 hours/week, 42 hours total)
   1. Proof techniques, numbers, sets, relations
   2. Deterministic and nondeterministic finite automata
   3. Regular expressions and regular languages
   4. Grammars, push-down automata, and context-free languages
   5. Turing machines and undecidability

c.

Computer Engineering

Relation of Course Outcomes to EAC Student Outcomes*

Course Outcomes (CE)	Student Outcomes											
	(a) apply knowledge of mathematics, science, and engineering	(b) design and conduct experiments, … interpret data	(c) design a system, component, or process to meet desired needs …	(d) function on multidisciplinary teams	(e) identify, formulate, and solve engineering problems	(f) an understanding of professional and ethical responsibility	(g) communicate effectively	(h) the broad education to understand the impact of engineering solutions …	(i) a recognition of the need for, and an ability to engage in life-long learning	(j) a knowledge of contemporary issues	(k) use the techniques, skills, and modern engineering tools ….	(CE) demonstrate knowledge of discrete mathematics [CE]
Criteria	a	b	c	d	e	f	g	h	i	j	k	CE
1. Prove theorems in discrete math by induction, contradiction, or cases.	3						1					3
2. Analyze, design, and manipulate finite state acceptors.	1		2		1							
3. Design and manipulate regular expressions.	1		1		2							
4. Prove languages not regular or context-free.	3						1					
5. Design and analyze context-free grammars and push-down automata.	1		3		1							
6. Analyze and simulate a Turing machine.	2											
7. Prove problems undecidable via reduction	3				1		1					

* 3 = major contributor, 2 = moderate contributor, 1 = minor contributor; blank if not related

d.

Computer Science & Computer Information Systems

Relation of Course Outcomes to CAC Student Outcomes*

Course Outcomes (CS & CIS)	Student Outcomes											
	All									CS		CIS
	(a) apply knowledge of computing and mathematics appropriate to the discipline	(b) analyze a problem, and identify and define the computing requirements …	c) design, implement, and evaluate a computer-based system, …	(d) function effectively on teams to accomplish a common goal	(e) An understanding of professional, ethical, legal, … responsibilities	(f) communicate effectively with a range of audiences	(g) analyze the local and global impact of computing on … society	(h) Recognition of the need for … continuing professional development	(i) current techniques, skills, and tools necessary for computing practice	(j) apply mathematical foundations, algorithmic principles, and CS theory …	(k) apply design and development principles	(j) An understand processes that support the information systems environment.
Criteria	a	b	c	d	e	f	g	h	i	j	k	j
1. Prove theorems in discrete math by induction, contradiction, or cases.	3					1				2		
2. Analyze, design, and manipulate finite state acceptors	2	2	1							3	1	
3. Design and manipulate regular expressions.	1									2	1	
4. Prove languages not regular or context-free.	3					1				2		
5. Design and analyze context-free grammars and push-down automata.	2	2	1							2	2	
6. Analyze and simulate a Turing machine.	1		1							1	2	
7. Prove problems undecidable via reduction.	2					1				3		

* 3 = major contributor, 2 = moderate contributor, 1 = minor contributor; blank if not related