CSCI 460: Theory of Computing

David M. Keil, Framingham State University, Spring 2013

SYLLABUS

Invitation

What does a computer-science professional need to know about abstract *models of computation*?

Do you have a precise way to define *computation* and *computing power*?

Having learned something about programming and about operating systems, applications, and databases, would you like to go deeper? Would you like to explore what the core of *computing* is, what the outer limits of computing are, and what separates the hardest computable problems from ones that can't be solved at all? If so, then this course has something to offer you.

Our inquiry

This course seeks to provide an environment to explore *theoretical* questions that have *practical* implications. One question is: What should matter to you in the course material? What does the course offer that you can use, as a computing professional?

Topic 1 (sets, languages, and models): Are there more *functions*, or *programs*? Does it matter?

Topic 2 (finite automata): What can state-transition systems model?

Topic 3 (pushdown automata): Can a stack enable state-transition systems to solve more problems? How do compilers work?

Topic 4 (Turing machines): Can a state-transition system with infinite tape model more computations than with a stack?

Topic 5 (random-access machines and recursive functions): What functions are computable?
What computational problems can't be solved?

Topic 6 (interactive computation):
Is communication part of computation?
Does concurrency require new models?

Topic 7 (multi-stream interaction): Is multi-agent, asynchronous interaction more powerful than sequential interaction? How does the brain work?

Meeting times

Tue 6:30-9:50 p.m., Hemenway Hall 132

To contact me

Office hours (Hemenway Hall 318A):
M 3:30-4:30 p.m.; W 10:30 a.m.-11:30 p.m.;
F 9:30-10:30 a.m.; others by appointment

Telephone: (508) 626-4724
Email: dkeil@framingham.edu
URL: www.framingham.edu/~dkeil/toc-matls.html

Course description (FSU catalog)

An introduction to theoretical computer science and some key applications. Course examines models of computation, including finite automata, transducers, pushdown automata, and Turing machines. Concepts of formal language theory are applied to lexical analyzer and compiler construction in programming-language translation. The course will include an introduction to the notions of computability and computational complexity, concepts used in parallel computation, and some aspects of artificial intelligence.

Prerequisites: CSCI 271 Data Structures *and either* MATH 292 Discrete Mathematics I or MATH 215 Finite Mathematics.

Reading

Hopcroft, Motwani, and Ullman, *Introduction to Automata Theory, Languages, and Computation*, 3rd Ed. (Addison-Wesley, 2007), 0-321-45536-3

Handout material that will include papers on computability and interaction

Slides: See handouts and course web site.

I like the textbook because it explains most of the main concepts of the course. The slides and study questions help tell which concepts are most important to the course.

Core objectives

The entire course will be guided by the following objectives and intended outcomes:

0a. Participate in class activities throughout the semester
0b. Solve problems as part of a team
0c. Present results in the classroom
0d. Present written results
0e. Show knowledge of facts and concepts
0f. Summarize the semester's learning
0g. Relate theoretical concepts to applications
1a. Use the notation of formal language theory
1b. Describe the logic-circuit model of computation
2a. Construct a finite automaton with a given behavior
2b. Prove that an automaton accepts a given language
2c. Give the regular expressions for a finite automaton
2d. Use constructive proof to show expressiveness of finite automata
3a. Explain the pushdown-automaton model
3d. Give a proof of expressiveness for pushdown automata
4a. Describe the Turing-machine model of computation
4b. Show a problem to be TM decidable
4c. Show a problem to be undecidable
5a. Describe the random-access-machine model
5b. Explain the relation between recursive definability and computability of functions
5c. Show expressiveness of the TM or RAM model
5d. Describe the Chomsky hierarchy of models of computation
6a. Distinguish algorithmic from interactive computation
6b. Describe a model of sequential interactive computation
7a. Describe forms of concurrency
7b. Describe a model of parallel computation
7c. Describe connectionist and multi-agent models

For other objectives, see topic slides.

Basic or prerequisite skills

a. Evaluate a formula in propositional or predicate logic
b. Explain what are sets, relations, and functions
c. Explain what are strings, arrays, and stacks
d. Explain proof by contradiction and the inductive proof method
e. Define and describe directed graphs

Classroom format and grades

The essay, "What we do in my classroom," *is part of this syllabus*. It has guidelines for assignments, collaboration, and grading. As it explains, for each topic, we have presentations, group work, discussion, assignments, and quizzes. Assigned work and quiz questions help to assess attainment of learning objectives.

Our classroom environment emphasizes active inquiry, participation, respect, and support among all participants. Learning is seen as the interactive construction of knowledge by the learner. We ask each other questions and investigate problems together.

Work includes small groups and blackboard work and report backs from each student. A semester project brings together the learning from the different topics and assignments. Frequent assignments and quizzes monitor progress and enable second chances.

Grades assess learning based on attainment of the stated objectives of the course. I score each item of work, or grading criterion, on a scale of 0 to 1.0.

Models of computation

Our central inquiry is about simple abstract representations of systems that transform symbols; i.e., *models of computation*. By the end of the course, you will have assembled a set of explanations, in your own words, about the power of some models of computation.

Our foundation is a group of concepts presented in discrete-mathematics courses: logic, proofs, sets, cardinality, relations, graphs, and functions. We will start with proofs that certain objects, such as real numbers, predicates, and functions, are uncountable.

We will use logic and methods of proof to explore relationships among certain sets (e.g., languages), certain graphs (e.g., automata), and certain functions (e.g., computable ones).

We will investigate three increasingly powerful machine models associated with state-transition systems, each of which corresponds to a language class (kinds of input sequences the model can recognize). This hierarchy of simple models of computation consists of finite automata, stack machines, and three equivalent models, Turing machines, random access machines, and recursively definable functions.

As we are discussing finite-state and pushdown automata, we will address program compilation, which puts these theoretical concepts into practice.

We will also look at theoretical models of interactive systems and parallel and distributed systems. We will present results showing that interaction requires more expressive models than those for algorithms. Our hypothesis is that multi-stream interaction requires more expressive models than sequential interaction.

This course differs from the other theoretical course in Computer Science, CSCI 347, Analysis of Algorithms, in that our concern here is models and their expressiveness or computational power, rather than design, performance, or efficiency.

It is expected that before taking CSCI 460, students are familiar, from their work with Discrete Math and Data Structures, with the notions of logic, sets, functions, relations, graphs, mathematical proof, different implementations of collections of data, and algorithms used to manipulate them.

Semester grading weights

The following categories group course objectives and outcomes (see previous page), which are assessed by means of assignments, quizzes, exams, and records of classroom discussion and presentations.

Application of concepts	
core topic objectives	35
other topic objectives	10
Knowledge of facts	10
Written contribution	20
Presenting results in person	10
Group activity	5
Summary and reflection	5
Attendance	5
	100

Semester project

Each student will summarize the learning in this course in a project that includes proofs of expressiveness and limitations of models of computation, done as part of assignments. Optional parts of the project include research and coding related to the course content. This project is part of the capstone project of the Computer Science major's CS concentration.

Research interest of instructor

Two special research themes are:

- Can systems of three or more interacting entities solve problems that two computing entities cannot?
- Can *indirect interaction via the environment* (e.g., bulletin boards, markers left on trails) give multi-agent systems more power than *message passing*?

I'm writing a doctoral thesis on scalable models of computation for multi-agent systems. Students in Theory of Computing are invited to participate in the research process by learning about these models and brainstorming about this question.

Accommodations

"Students with disabilities who request accommodations are to provide Documentation Confirmation from the Office of Academic Support within the first two weeks of class. Academic Support is located in the Center for Academic Support and Advising (CASA). Please call (508) 626-4906 if you have questions or if you need to schedule an appointment." (See *www.framingham.edu/ CASA/ Accommodations/accomm.htm*.)

Course Plan

Dates	Topic	Readings
1/22	*Introduction and discrete-math review*	Handouts[1,2]; Hopcroft-Motwani-Ullman, Ch. 1; Savage[3], Ch. 1
1/29	1. Sets, languages, and models	Handouts[4,5,6,7]
2/5-2/12	2. Deterministic finite automata, regular languages, and lexical analysis	H-M-U, Ch. 2-4
2/12	*Research proposals*	
2/19	*Problem-solving quizzes on topics 1-2*	
2/19 – 3/5	3. Pushdown automata, context-free grammars, and parsing	H-M-U, Ch. 5-7
3/5 – 3/12	4. Turing machines and undecidability	H-M-U, Ch. 8-9
3/12	*Research reports and quiz review*	
3/26	*Problem-solving quizzes on topics 3-4*	
3/26 – 4/2	5. Random access machines and recursively definable functions	Handout[8]
4/2	*Make-up quizzes on topics 1-4*	
4/2 – 4/9	6. Models of interactive computation	Handouts[9,10,11]
4/16	*Research reports and quiz review*	
4/16	*Problem-solving quizzes on topics 5-6*	
4/16 – 4/23	7. Models of concurrency and multi-stream interaction	Handouts[12,13,14]
4/23 – 4/30	*Course summary, quizzes, and research reports*	
4/30	*Final exam (problem solving)*	
5/7	*Final exam (multiple choice)* Optional objectives questions	

[1] D. Keil, Theory uses definitions, examples, theorems, and proofs, 2011.
[2] D. Keil, Abstractions, functions, and recursion, 2011.
[3] John E. Savage, *Models of Computation* (Addison Wesley, 1998).
[4] D. Keil, The diagonal proof technique, 2008.
[5] P. Fejer and D. Simovici, *Mathematical Foundations of Computer Science*, 1991, excerpt.
[6] D. Keil, Why induction and coinduction are important, 2010.
[7] Handout on Gödel's theorem
[8] Davis, Sigal, Weyuker, excerpt.
[9] D. Keil, Definitions related to interaction, 2009.
[10] P. Wegner, Why interaction is more powerful than algorithms, *Comm. ACM*, 5/97.
[11] D. Goldin, Persistent Turing Machines as a model of interactive computation, 1999.
[12] B. Maggs, L. Matheson, R. Tarjan, Models of Parallel computation: A survey and synthesis, 1995.
[13] D. Keil and D. Goldin, Indirect interaction vs. message passing, 2009.
[14] Handout on neural nets