

CSCI 460: Theory of Computing

David Keil, Framingham State University, Spring 2012

SYLLABUS

Invitation

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.

Course description (FSC 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.

Meeting times

Tue 6:30-9:50 p.m.
Hemenway Hall 132 (annex)

To contact me:

Office hours (Hemenway Hall 318A):
M 12:30-1:30 p.m., W 10:30-11:20 a.m.,
Tue. 4:00-4:30 p.m.; F 1:30-2:20 p.m.
Others by appointment
Telephone: (508) 626-4724
Email: dkeil@framingham.edu
URL: www.framingham.edu/~dkeil/toc-matls.html

Course overview

Our central theme is *models of computation*. We will inquire about a number of matters under discussion. Our foundation for this course is a group of concepts presented in Discrete Mathematics courses: logic, proofs, sets, cardinality, relations, graphs, and functions. We will present proofs that certain objects, such as real numbers, predicates, and functions, are uncountable.

We will use logic and methods of proof to show 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 diagrams*, each of which corresponds to a *language model* (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 *functions* that may be defined recursively.

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 *expressiveness* or *computational power* rather than *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.

What we will investigate

1. What is computation?
2. How many of the following exist? (Natural numbers; real numbers; functions; programs)
3. What problems can be solved by computation?
4. How do compilers work?
5. What can state-transition systems model?
6. Can a state-transition system with stack model more computations than without?
7. Can a state-transition system with infinite tape model more computations than with stack?
8. What functions are computable?
9. What problems can be solved?
10. Is the Von Neumann model complete as a model of computation?
11. Is *communication* part of *computation*?
12. Does concurrency require new models?
13. Is multi-agent, asynchronous interaction more powerful than sequential interaction?
14. How does the brain work?

Objectives and desired learning outcomes

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

- 0a. Participate in class activities throughout the semester
- 0b. Solve a problem as part of a team
- 0c. Present a short talk in the classroom
- 0d. Write a documented research paper
- 0e. Relate theoretical topics to applications (e.g., queuing, robotics, OCR, vision image processing, and information retrieval)
 - 1a. Write a proof showing countability of a set
 - 1b. Write a diagonal proof showing uncountability of a set
 - 1c. Explain or write a coinductive definition
 - 1d. Use the notation of formal language theory*
- 2a. Construct a finite automaton with specified behavior *
- 2b. Convert between regular expressions and finite automata*
- 2c. Use non-diagonal proof by contradiction, e.g., the Pumping Lemma
- 2d. Use constructive proof to show expressiveness of finite automata*
- 2e. Prove that a given finite automaton accepts a given language*
 - 3a. Explain the pushdown-automaton model*
 - 3b. Define a context-free grammar

- 3c. Perform a derivation using a CFG*
- 3d. Give a proof of expressiveness for pushdown automata.
 - 4a. Describe the Turing-machine model of computation
 - 4b. Show a problem to be decidable
 - 4c. Show a problem to be undecidable
 - 4d. Explain the notion of reducibility of problems*
 - 5a. Explain the relation between recursive definability and computability of functions*
 - 5b. Describe the random access machine model
 - 5c. Show expressiveness of the RAM model
 - 5d. Describe the Chomsky hierarchy of models of computation*
 - 5e. Distinguish decidability from semidecidability
 - 6a. Distinguish algorithmic from interactive computation
 - 6b. Compare and contrast models of algorithmic and interactive computation
 - 6c. Define a persistent Turing machine
 - 7a. Describe forms of concurrency*
 - 7b. Describe some models of parallel computation
 - 7c. Describe connectionist and multi-agent models

*Core learning outcome

Grades and classroom format

The essay, "What we do in my classroom," attached, *is part of this syllabus*. See especially guidelines there for assignments and collaboration. As explained there, for each topic, we have presentations, discussion, assignments, and quizzes. Grades are based on attainment of objectives as follows.

I score each item of work submitted, or each grading criterion, on a scale of 0 to 1.0. Assigned work and quiz questions help to assess attainment of objectives. Students will have multiple opportunities in the semester to show attainment of each desired learning outcome; the highest level of attainment will be recorded as a score for the outcome. The entire semester grade is computed from numbers showing attainment of these objectives.

Basic or prerequisite skills

1. Explain strings, arrays, linked lists, and stacks
2. Explain proof by contradiction and the inductive proof method
3. Define and represent directed graphs
4. Explain what are sets, relations, and functions
5. Evaluate formulas in propositional and predicate logic

Strongly recommended 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.

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 capabilities and knowledge	
core objectives	25
other objectives	25
Knowledge of concepts and facts	15
Independent inquiry	10
Presenting results in person	10
Participation and keeping pace	<u>15</u>
	100 %

Optional coding project

Students may optionally complete a short project to implement some computation or model discussed in topics of the course. One option is to write a lexical analyzer and parser for arithmetic expressions. This project relates topics 2 and 3 to compiler construction.

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

	Topic	Readings
1/25	<i>Introduction and discrete-math review</i>	Handouts ^{1,2} ; Hopcroft-Motwani-Ullman, Ch. 1; Savage ³ , Ch. 1
1/31	1. Countability, diagonal proof, incompleteness, coinduction	Handouts ^{4,5,6,7}
2/7-2/14	2. Deterministic finite automata, regular languages, and lexical analysis	H-M-U, Ch. 2-4
2/21	<i>Research proposals and quiz review</i>	
2/21	<i>Problem-solving quizzes on topics 1-2</i>	
2/21-2/28	3. Pushdown automata, context-free grammars, and parsing	H-M-U, Ch. 5-7
2/28-3/6	4. Turing machines and undecidability	H-M-U, Ch. 8-9
3/20	<i>Research reports and quiz review</i>	
3/20	<i>Problem-solving quizzes on topics 3-4</i>	
3/20-3/27	5. Random access machines and recursively definable functions	Handout ⁸
3/27	<i>Make-up quizzes on topics 1-4</i>	
3/27-4/3	6. Models of interactive computation	Handouts ^{9,10,11}
4/10	<i>Research reports and quiz review</i>	
4/10	<i>Problem-solving quizzes on topics 5-6</i>	
4/10-4/17	7. Models of parallel and distributed computation	Handouts ^{12,13,14}
4/24-5/1	<i>Quizzes, research reports, and course summary</i>	
5/8	<i>Final exam (Topics 1-7)</i> 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