

*David Keil*  
*Framingham State University*

# Theory of computing

## Introduction

1. What this course offers
2. Models and languages
3. Theory in computer science
4. Course details and overview

David Keil Theory of Computing 1/12
1

## 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
  - what separates the hardest computable problems from ones that can’t be solved at all?
- If so, then this course has something to offer

## A software-engineer's tool

- *Imagine* a program that reads a Java file and describe the behavior of the Java program
- *A simpler problem:* For a given Java file, are there *any* inputs that produce an infinite loop?
- *Even simpler:* For a given Java file and a given input, does the program hang?
- *For ten million dollars, would you participate in a project to solve this problem?*

## 1. What this course offers

### *Goals:*

- To get to the core of computing and leave some details aside
- To link theory to practical concerns
- To *challenge the way you think* about computing and to help you *build your own new conception* of computing
- To allow you to work *collaboratively* with others in theory of computing

## Inquiry

- What problems can computers solve?
- Does the brain work like a computer?
- What is computation?
- What is theory?
- What is a model?

## A proposed agreement

### *Instructor commitments:*

- know the course material well and present it clearly
- return submitted work within a week
- answer questions in a helpful way

### *Student commitments:*

- ask questions
- answer reasonable questions, risking being wrong
- submit work on time, even if incomplete
- work sometimes in groups
- present results to the class, at the blackboard

*Both:* respect all contributions to classroom discussion

## How do you learn?

- How do you learn best (reading, listening, writing, group work, class discussion, research)
- What's your interest in the subject?
- How do you think about the topic?
- Your interests?
- Your work?

## Some questions

- How can we think about *computation* (*computing*), without thinking about particular computers?
- What is computing?
- Are some computers and programs *more powerful* than others? In what ways?
- What is the simplest machine like that can solve the *hardest* of a category of information-processing problems?

## What is computing?

*Proposition:*

Computing is the execution of algorithms

- Turing's model of algorithmic computing "had all the capabilities of today's computers, at least as far as in what they could compute," with the goal of "describing precisely the boundary between what a computing machine could do and what it could not do" (Hopcroft-Motwani-Ullman, 2007)

## Is computing more than the execution of algorithms?

- "Most of the computing agents with which computing science is concerned . . . exhibit a behavior which is not just the computation of a mathematical function of their inputs . . . but rather a possibly infinite sequence of communications with their environment" (Milner, 1975)
- "[The] assertion that algorithms capture the intuitive notion of what computers compute is invalid" (Wegner, 1997)

## Algorithms vs. interaction

*Proposition:* Interaction is a kind of computing that is more powerful than algorithms (P. Wegner)

- *Example:* Driving while looking at road is more successful than driving looking only at a list of instructions
- *Example:* By interrogation a D.A. exposes a suspect who would escape if given a questionnaire
- *Result:* New models of interactive computation are needed

## Intelligence and interactive problems

- *Thesis:* Intelligence is developed by and for interaction
- *Learning* is adaptation to partially observable, unpredictable, complex environment
- Intelligent agents conduct search for satisfactory *policy*, of responses to states of environment, based on rewards

## 2. Models and languages

- *Automata* (DFA, PDA, TM, RAMs, PRAMS, sequential interaction machines, multistream interaction machines) give *interior views* of processes (white-box)
- Some automata are expressed as graphs or transition systems
- *Languages* (regular languages, context-free languages, recursive sets, stream languages), express *observable behavior*
- *Functions*: transformation vs. acceptance

## Models

- *Model*: A representation that abstracts from details
- Chomsky hierarchy of computational models:
  - Deterministic finite automata recognize regular languages
  - Pushdown automata recognize more languages
  - Turing machines recognize recursive (computable) sets
- Models of algorithms:  
Turing machine; random-access machine
- Models of interaction
- Models of reasoning and learning

## More inquiry

- How many of the following exist? (Natural numbers; real numbers; functions; programs)
- What problems can be solved by computation?
- How do compilers work?
- What computations can state-transition systems model?
- Can a state-transition system with stack model more computations than without?

## Yet more inquiry

- Can a state-transition system with infinite tape model more computations than with stack?
- What functions are computable?
- What problems can be solved?
- Is the Von Neumann model complete as a model of computation?
- Is *communication* part of *computation*?
- Does concurrency require new models?
- How does the brain work?

## Expressiveness of models

- Models are comparable in *power* or *expressiveness*
- *Example:* There are more reals than integers, functions than programs, truths than proofs.
- *Examples:* A device with finite memory or stack memory cannot model all algorithms; a Turing machine or random-access machine can model any algorithm
- *Example:* Turing machines cannot model interaction; Persistent TMs can
- *Instructor's research goal:* To show that models of indirect interaction in multi-agent systems are more expressive than message-passing models

## 3. Theory in computer science

- *Examples:*
  - Computability/uncomputability of certain functions
  - Church-Turing Thesis identifies algorithm computation with Turing Machine and random-access machine models
  - Complexity theory identifies tractability with polynomial-time problems
- Practical results made possible by work in theory:
  - The general-purpose computer
  - Programming languages
  - The Internet

## Theory in the sciences

- *Physics*: Theory is the study of the most basic particles and forces
- *Biology*: Theory includes frameworks of chemistry and evolution
- *Computer science*: Theory is mostly a set of mathematical proofs about models, classes of algorithms, and problems
- Bottom-up theoretical results are empirical, based on experiment and experience
- Top-down results use conjecture and proof

## Theory and practice

- Like physics or sociology, computer science has a *theory*, expressed mathematically
- The basic *concepts* and *foundations* of a discipline are its theory
- Practical applications:
  - *Compilation*: lexical analysis and parsing
  - Recognition of *limits* of solvable problems
  - Guidance in assessing computational power (expressiveness) of a system

## Theory includes proof

- *Constructive* -- e.g., showing that something is possible (Kleene)
- By *contradiction* (e.g., that something is impossible) – (diagonal proofs)
- *Diagonal* -- e.g., sizes of sets (Cantor, Turing, Gödel)
- By *induction* (if we can show some simple property, then we can show the same property for a whole class of objects)

## 4. Course details and overview

### Topics

Introduction and discrete-math review

1. Countability, incompleteness, coinduction
2. Deterministic finite automata
3. Pushdown automata
4. Turing machines and computability
5. RAMs and recursively definable functions
6. Models of interactive computation
7. Models of parallel and distributed computation

## Multiple-topic objectives

- 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, information retrieval)

## Topic 1: Countability, incompleteness, coinduction

- 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\*

## Topic 2: Deterministic finite automata and regular languages

- 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\*

## Topic 3: Context-free languages and pushdown automata

- 3a. Explain the pushdown-automaton model\*
- 3b. Define a context-free grammar
- 3c. Perform a derivation using a context-free grammar\*
- 3d. Give a proof of expressiveness for pushdown automata.

## Topic 4: Turing machines

- 4a. To describe the Turing-machine model of computation
- 4b. To show a problem to be decidable
- 4c. To show a problem to be undecidable
- 4d. To explain the notion of reducibility of problems

## Topic 5: Random-access machines and recursively definable functions

- 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

## Topic 6: Interactive computing

- 6a. To distinguish algorithmic from interactive computation
- 6b. To compare and contrast models of algorithmic and interactive computation

## Topic 7: Models of parallel and distributed computing

- 7a. To describe forms of concurrency
- 7b. To describe some models of parallel computation
- 7c. To describe connectionist and multi-agent models

## Grading weights

Application of knowledge	
core objectives	25 %
other objectives	25
Knowledge of facts	15
Independent inquiry	10
Presenting results in person	10
Participation	15

## Assessment and grading methods

- Topic problem-solving quizzes
- Multiple-choice quizzes
- Final exam
- Assignments
- Research paper
- Presentations
- Data on keeping pace and participation in class discussion

## Prerequisites

- *Data Structures concepts*: stack, queue, tree, root, leaf, graph, vertex, edge
- *Discrete Math concepts*:
  - negation, disjunction, conjunction, implication, logical equivalence, quantifiers ( $\forall$ ,  $\exists$ )
  - set membership, inclusion, intersection, union, complement, power set, partition
  - Relations, functions, injections, bijections, surjections, Cartesian product
  - Direct proof, proof by contradiction, induction, construction

## Questions

- What most stayed in your mind in discussing this topic?
- For you, what was the *least* clear concept that you encountered in this topic?

## References

John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*, 3<sup>rd</sup> Ed. Addison-Wesley, 2007.

Robin Milner. Processes: A Mathematical Model of Computing Agents. In H. E. Rose and J. C. Shepherdson, Eds., *Logic Colloquium '73*, North-Holland, 1975.

Peter Wegner. Why Interaction is More Powerful than Algorithms. *Communications of the ACM* 40(5), 1997.