

# Illinois State University

ITK 328-1, Spring 2008

## Introduction to The Theory of Computation

---

**Instructor:** Chung-Chih Li, Ph.D.

**Class Time:** MW STV-108 11:00~12:15

**Office Room:** Old Union 105

**Office Hours:** MW - 13:00 ~ 13:50

Tue - 10:10 ~ 12:15, 13:00 ~ 13:50

Thu - 11:00 ~ 11:50, 13:00 ~ 13:50

**Contact:** Tel:(309) 438-7952, Email: cli2@ilstu.edu (email is preferred, unless it's an emergency)



**WebPage of the course:** <http://www.itk.ilstu.edu/faculty/chungli/ITK328>

Students should regularly check the webpage for important information about assignments, data, due dates, lecture notes, and announcements. *However, announcements made in the class should be considered as official if they are not consistent with the ones on the webpage, since I may not be able or remember to update every announcement.*

**Prerequisites:** ITK 279 with a C or better grade.

We assume that students already have fair experience and skill to solve nontrivial problems with algorithms and are able to implement them in some contemporary programming language. A certain maturity of mathematical concepts are essential to succeed in this course. Before come to the class, students are expected to be familiar with some subjects mostly in discrete mathematics such as logics, sets, relations, graphs, trees, mathematical induction and other proof techniques. We will try to fix and clarify any misconceptions whenever needed in the class or in instructor's office hours. However, due to the time constraints, we are not able to review all the basic mathematical concepts. For example, we are not going to teach you how to find the gcd of two natural numbers.

**Textbook:**

- *An Introduction to Formal Languages and Automata*, by Peter Linz, Jones and Bartlett Publishers, 4<sup>th</sup> Edition, 2006.

The textbook is required and the students are asked to bring the textbook to the class. Xerox copies are not allowed. Older version of the book is acceptable.

**Reference Books:**

- (1) *Introduction to Automata Theory, Languages and Computation*, by John E. Hopcroft and Jeffrey D. Ullman, Addison-Wesley Publishing Company, 1979.
- (2) *Computability, Complexity, and Languages*, by Martin Davis and Elaine J. Weyuker, Academic Press, 1983.
- (3) *The Universal Computer: The Road from Leibniz to Turing*, by Martin Davis, W. W. Norton & Company, 2000.
- (4) *Problems on Discrete Mathematics, Volumes I & II*, by Chung-Chih Li and Kishan Mehrotra, Syracuse University Press, 1998.

Books listed in (1) and (2) are two classic textbooks in the field. Some revisions and up-to-date theory are added to their new editions. (3) is written for the general public by eminent Professor Davis, which is an introduction to the western tradition of seeking computing machines. Students who are lack of mathematical background can quickly pick up the required concepts from Volume I of the book listed in (4) above. The book is out of print, but an e-copy is available on the webpage of the class.

### What is Theoretical Computer Science:

Theoretical computer science extends across three major areas: Automata, Computability, and Complexity. Since the three areas form the firm foundation of computer science from both applicative and theoretical aspects, we believe that students majoring in computer science should be familiar with the topics in a certain depth.

Each of the three areas puts different emphases on the notion of computation. The theory of automata dealing with the relations between formal languages and computing machines directly benefits many applications in computer science such as compilers, control formalisms, and programming and natural languages processing. We will discuss various finite state machines and examine a few language families in the standard hierarch of formal languages. In particular, we will have a close look at regular languages and context-free languages.

As one of the oldest intellectual inquiries, the study of computability can be traced back to ancient Greece. In modern study, the theory systematically characterizes what can be solved by mechanical procedures, and the notion turns out to be the very idea of the modern computer. Here we will use Turing Machines as our standard computing formalism to explain the notion of computability. Then, some important concepts such as the well-known Church-Turing thesis, decidability, reducibility, and the recursion theorem will also be briefly discussed.

A theoretical reflection on the study of algorithms gave birth to this relatively new research area called Computational Complexity Theory. Problems that cannot be solved efficiently by computers in many cases are not because of the lack of computer power or good programming skill, but because of the inherited nature of the problems. Thus, we need to provide a theoretical framework to characterize computable problems in terms of their innate difficulties. Classes such as P, NP and NP-complete, LogSPACE, PSPACE, and BPP are some typical and well-studied complexity classes. However, we are not able to cover them in depth due to the time constraints. This topic is indeed more suitable to be studied at the graduate level.

**Course Objectives:** While the course is designed to give students a general picture of the underlying theory of computation, we rather want to emphasize on the automata part and make this course more applicable. As a matter of fact, automata theory directly related to the design of programming language and compilers. We will perforce leave the introductions to computability and complexity theory brief and superficial. We expect students to be familiar with the following topics upon their exit of the course:

1. Finite Automata
2. Conversion between DFA and NFA
3. Minimize DFA
4. Regular Languages and Regular Expressions
5. Conversion between Regular Languages, Regular Expressions, and NFA
6. Context-free Grammars and Languages
7. Pushdown Automata
8. Conversion between Context-free Grammars and Pushdown Automata
9. Nondeterministic Pushdown Automata
10. Exclusive Properties and Pumping Lemmas
11. Context-sensitive Grammars
12. Turing Machines

13. Computability and Decidability
14. Recursive and Primitive Recursive Functions
15. A insight of NP and NP-Complete problems

**Examinations:** (400 points)

Two midterms and one final exam; 100 points for each midterm and 200 points for the final.

- Unless announced otherwise, all tests are accumulative, closed book, and indispensable. No makeup test will be given unless a documented absence is authorized by the university.
- Every student is allowed to bring a *self-prepared hand-writing* crib sheet to the test. You can **write** down anything on both sides of **one** letter-sized paper. No circulation during the test.

Tentative Test Schedule

6 <sup>th</sup> week	Midterm I	100 points	Feb. 18
12 <sup>th</sup> week	Midterm II	100 points	Mar. 31
17 <sup>th</sup> week	Final Exam	200 points	May 5, 7:50 AM

**Pop quizzes:** (100 points)

Some pop quizzes will be given without notice in advance. Each quiz carries 10~20 points towards students' final scores. The coverage of every quiz is also accumulative, including the materials that are three-month-old and those covered in the class right before the quiz. A typical quiz takes about 10 minutes. No makeup quiz will be given if missed. If you miss a quiz due to a university authorized absence, we will use the average of your rest quizzes as the score; otherwise, you get a 0 for the missing quiz.

**Homework Assignments:** (200 points) About 15 homework assignments will be given, each carries 10 ~ 20 points, depending on the difficulty. Follow the following guideline for submission:

1. A cover page with your name and student ID on it.
2. Your solutions; handwriting is acceptable *only if* your writing is clear and with appropriate spacing and organization.
3. Staple all pages together and put them in a letter-sized Manila folder with your name on it. If your paper is torn from a spiral notebook, fine, but trim the edge.
4. Every assignment is due at the beginning of the class on the due day. No late work will be accepted unless under some inevitable circumstances with proof.

**Programming Assignments:** (100 points) There will be 3 programming assignments as follows:

Program I (100 points): Conversion an input regular expression to a minimized DFA.

Program II (optional, 30 points): Simulate a pushdown automata to recognize a language generated by a given context-free grammar.

Program III (optional, 30 points): Simulate a Turing machine to compute some arithmetic operations.

Detailed descriptions for each programming assignment will be given.

See the section of program submission guideline for details

**Reading Assignment:** (100 points)

To let students be aware of The history of computing machines and how people push the limit of the power of mechanical/symbolic rules is one of the most important (but forgotten) goal of this course. Students lost the whole picture in technical details. Thus, instead of asking you to write some programs

that are mostly irreverent to the concepts we want you to pick up from taking ITK328, I rather want you to have a chance to explore to this wonderful journey of seeking rationality in the long Western tradition and know how could this journey possibly bring us to the point as we are now where the Internet and digitized daily life are already taken for granted. Therefore, I decide to take this approach that is not a traditional one in this course. There are a few excellent books written for the general public but Martin Davis's *The Universal Computer: The Road from Leibniz to Turing* seems to me the best one on the subjects. The publisher changes the book title for its paperback edition to *Engines of Logic: Mathematicians and the Origin of the Computer*. (Professor Davis regrets to let the publisher make the change.) Detailed information about the books are as follows:

- *The Universal Computer: The Road from Leibniz to Turing* by Martin Davis. Hardcover: 256 pages Publisher: W. W. Norton & Company; ISBN-10: 0393047857 ISBN-13: 978-0393047851. (\$17.79 from Amazon)
- *Engines of Logic: Mathematicians and the Origin of the Computer* by Martin Davis. Paperback: 272 pages, Publisher: W. W. Norton & Company; ISBN-10: 0393322297 ISBN-13: 978-0393322293. (\$11.66 from Amazon)

While the book is written for the general public, there are some nontrivial mathematical concepts and notations scattered through out the book. ITK328's students should have a better chance than ordinal readers to understand and appreciate Professor Davis's book.

**What to do:** You are asked to read the book carefully and write a report in at least 2500 words to comment and summarize the book.

**Due date of the reading report: The 15<sup>th</sup> week's Wednesday (April 23)**

**Class Participation and Attendance :** (100 points) Attendances will be taken impulsively; 10 points will be added if you are present when the attendance is taken. To get these 100 points as many as possible, you should take care of the attendance, class behaviors, helping peer students inside and outside the classroom, and participation in the googlegroups setup for this course (see below).

**Googlegroups:** One email discussion group will be setup using googlegroups for the class. Every students who currently registered in the course should subscribe to the discussion group at

ITK328@googlegroups.com.

You can subscribe to the group with more than one email address for your convenience. You are encourage to discuss anything related to to the class in this forum such as sharing your experience, asking for help, offering help, hints, opinions, suggestions, even complains or setting up a party for the class. No disgraceful languages, of course. I will not intervene the discussion unless it is necessary. This is your forum.

### Academic Honesty:

In addition to receiving 0 point, cheating, plagiarism, collusion, abuse of resource materials, and their consequences are defined and described in *ISU 2006-2007 Undergraduate Catalog*, Section: Academic Policies and Practices, Article: Academic Integrity (Page 63) and *Code of Student Conducts* under X.C. Disciplinary Bodies And Procedures – Academic Honesty Cases. Students giving away academic works for assignment offered for credit to other students working on the same assignment will be considered as guilty as academic dishonesty, and will receive the same penalty. More information can be found at: [http://www.deanofstudents.ilstu.edu/crr/downloads/Code\\_of\\_Student\\_Conduct.pdf](http://www.deanofstudents.ilstu.edu/crr/downloads/Code_of_Student_Conduct.pdf)

**Program Submission Guideline:** Follow the following guideline for program submission:

1. Put a few comment lines at the beginning of your program file, in which you should clearly indicate your name and ID and claim your copyright. **Student who fails to do so will receive 0 point on the assignment.**

2. A cover page with your name and student ID on it.
3. A brief summary about the assignment and your approach to the problem. You may include the difficulties you had faced, if any, or why you think your program doesn't work. It is very common and not a shame to admit that your program doesn't work under the time constraints, but a reasonable self-diagnosis deserves reasonable partial credit.
4. A hard-copy of the source codes.
5. A hard-copy of the directly output of your program, if any.
6. All item above should be put in a letter-sized Manila folder with your name on it.
7. As homework assignment, every programming assignment is due at the beginning of the class on the due day. No late work will be accepted **under any circumstances**.
8. I do not need a soft copy, *but you have to make it ready for me to check any time after the due day*. I will ask you to demo your program if I have doubts or simply by random.

Do backup your works as often as possible. Remember: bad things do happen, and "my dog ate my works" is not a good excuse.

Students are encouraged to discuss assignments and help each other. However, this does not mean that you can either entirely or partially copy or modify other's works.

**Try very hard to avoid the following troubles:**

1. Any form and any degree of plagiarism will receive 0 point.
2. If your program contains syntax error, you will receive 0 point.
3. If the hard-copy of the direct output of your program is inconsistent to your program's design, you will receive 0 point. This is a kind of cheating.

### Grading Policy:

Given 1000 as the perfect score, your grade is based on the percentage of the total points you receive according to the following scheme.

Percentage of total points	Grade	
90 %	A	Excellent
80 %	B	Good
65 %	C	Satisfactory
50 %	D	Passing
- - -	F	Failure

**I do not curve!!**

Thus, you don't have to knock down your friends to get a good grade. In other words, you can't hide behind someone else, because you two could be both shot down. So, do help your classmates if they need you.

Moreover, if I have good reasons, I have no problem to give all A's, neither do I have problem to give all F's.

### Tentative Topics and Schedule:

Keep the table of tentative topics and schedule in the following handy, and try to keep up with the schedule. Read the assigned materials before the class.

Week	Topics	Reading
1: Jan. 14	Introduction to theoretical computer science, and mathematics preliminaries.	Syllabus 1.1, 1.2, 1.3
2: Jan. 21	(Jan. 21, MLK Jr. Holiday). finite automata, languages, deterministic vs. nondeterministic automata.	2.1, 2.2, 2.3
3: Jan. 28	Equivalence and reduction of automata, regular expressions, regular languages.	2.4, 3.1, 3.2
4: Feb. 4	Regular grammars, right- and left-linear grammars.	3.3
5: Feb. 11	Closure properties and membership of regular languages.	4.1, 4.2
6: Feb. 18	<b>(Midterm 1)</b> , identifying non-regular languages: the use of the pigeonhole principle and a pumping lemma.	4.2, 4.3
7: Feb. 25	Context-free languages, leftmost and rightmost derivations, derivation trees, sentential forms.	5.1, 5.2
8: Mar. 3	Context-free grammars and programming languages, simplification.	5.3, 6.1
9: Mar. 10	(Spring Break)	
10: Mar. 17	Chomsky normal forms, Greibach normal forms, a parser algorithm, nondeterministic pushdown automata.	6.2, 6.3, 7.1
11: Mar. 24	Pushdown automata and context-free languages, deterministic vs. nondeterministic pushdown automata.	7.1, 7.2, 7.3
12: Mar. 31	<b>(Midterm 2)</b> , pumping lemmas for context-free languages and linear languages.	7.4, 8.1
13: Apr. 7	More on pumping lemmas, closure and other properties of context-free languages.	8.1, 8.2
14: Apr. 14	Turing Machines, Church-Turing's thesis.	9.1, 9.2, 9.3
15: Apr. 21	Language hierarchy, recursive, recursively enumerable, and undecidable languages.	Ch. 11, 12
16: Apr. 28	Scratch the issues can't be covered in details.	Ch. 13, 14
17: May 5	<b>Final Examination.</b>	

**Play Ball!!**