Languages and Automata

What are the Big Ideas?

Tuesday, September 7, 2010

Reading: Sipser 0.1

CS235 Languages and Automata

Department of Computer Science
Wellesley College

Why Take CS235?

1. It's required for CS majors. (bad reason)
2. It teaches some important Big Ideas. (good reason)
3. It's a lead-in to Programming Languages (CS251) and Compilers (CS301)*, in terms of both theory and practice (scanning, parsing, ML programming).
4. It's a lead-in to CS310 Theoretical Foundations in Cryptology†.

* Hasn't been taught for awhile, but might be offered again in the future if there's sufficient interest.

† Will be taught by Randy in Spring, 2011

The Big Ideas

1. Models of Computation
2. Undecidability: Not Everything is Computable!
3. Mathematical Foundations for CS
4. Cool Applications

#1 Models of Computation: Regular Languages

E.g.: all binary strings consisting of all 1s or sequences of 10s
(includes the empty string, written ε)

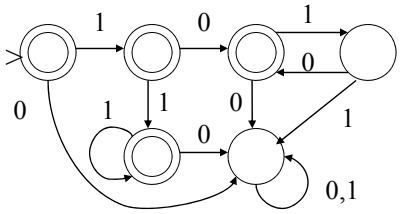
Regular Expression

$1^* \cup (10)^*$

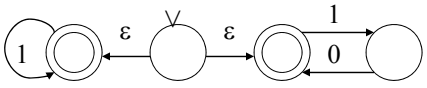
Right-Linear Grammar

$S \rightarrow A$
$S \rightarrow B$
$A \rightarrow \varepsilon$
$A \rightarrow 1A$
$B \rightarrow \varepsilon$
$B \rightarrow 10B$

Deterministic Finite Automaton



Nondeterministic Finite Automaton



#1 Models of Computation: Context-Free Languages

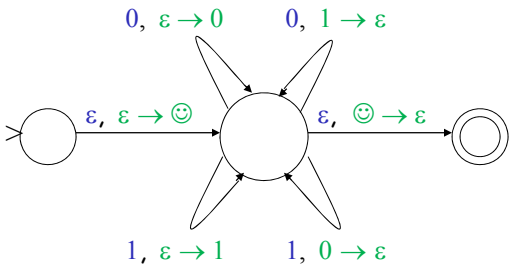
E.g.: all binary strings with an equal number of 0s and 1s.

Note: this language is *not* regular!

Context Free Grammar

$S \rightarrow \%$
$S \rightarrow SS$
$S \rightarrow 0S1$
$S \rightarrow 1S0$

Nondeterministic Pushdown Automaton

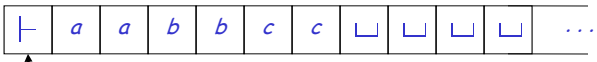


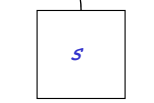
Q: Can we create a deterministic recognizer/parser for a CFL?

A: Yes, but can only do so efficiently for a restricted subclass of CFL.

#1 Models of Computation: Turing Machines

E.g.: all strings of the form $a^n b^n c^n$. (This is *not* context free!)

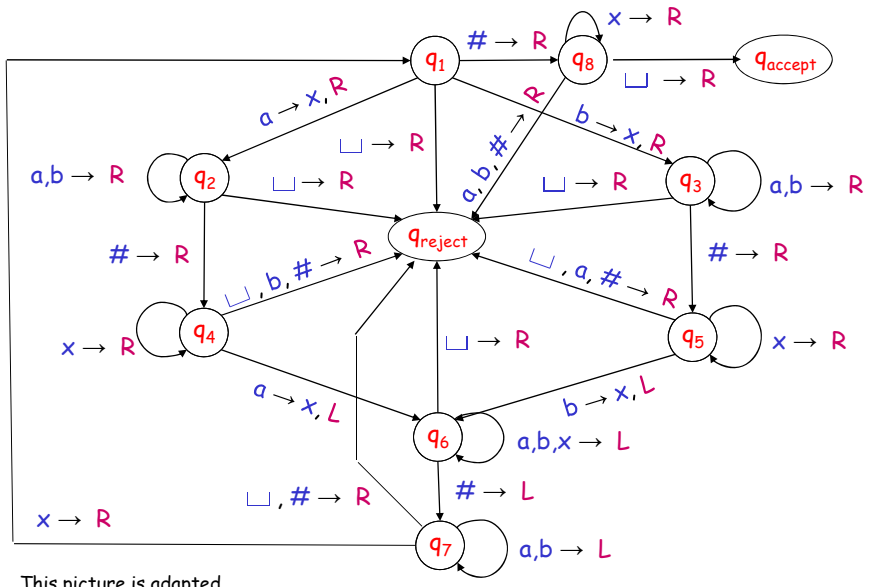




finite control

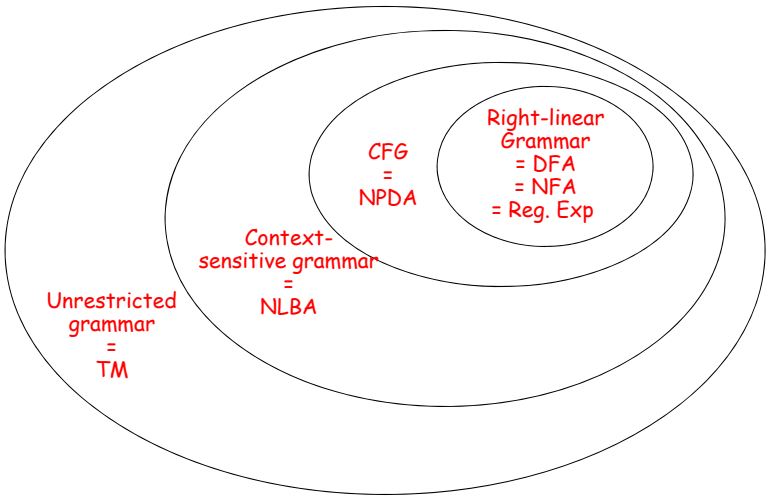
	⊢	a	b	c	⊔	⊣
s	$(s, \vdash, R)$	(s, a, R)	(q_1, b, R)	(q_2, c, R)	$(q_3, \dashv, L)$	–
q_1	–	$(r, -, -)$	(q_1, b, R)	(q_2, c, R)	$(q_3, \dashv, L)$	–
q_2	–	$(r, -, -)$	$(r, -, -)$	(q_2, c, R)	$(q_3, \dashv, L)$	–
q_3	$(t, -, -)$	$(r, -, -)$	$(r, -, -)$	$(q_4, \sqcup, L)$	$(q_3, \sqcup, L)$	–
q_4	$(r, -, -)$	$(r, -, -)$	$(q_5, \sqcup, L)$	(q_4, c, L)	$(q_4, \sqcup, L)$	–
q_5	$(r, -, -)$	$(q_6, \sqcup, L)$	(q_5, b, L)	–	$(q_5, \sqcup, L)$	–
q_6	$(q_7, \vdash, R)$	(q_6, a, L)	–	–	$(q_6, \sqcup, L)$	–
q_7	–	$(q_8, \sqcup, R)$	$(r, -, -)$	$(r, -, -)$	$(q_7, \sqcup, R)$	$(t, -, -)$
q_8	–	(q_8, a, R)	$(q_9, \sqcup, R)$	$(r, -, -)$	$(q_8, \sqcup, R)$	$(r, -, -)$
q_9	–	–	(q_9, b, R)	$(q_{10}, \sqcup, R)$	$(q_9, \sqcup, R)$	$(r, -, -)$
q_{10}	–	–	–	(q_{10}, c, R)	$(q_{10}, \sqcup, R)$	$(q_3, \dashv, L)$

TM State Transitions for {w#w | w ∈ {a,b}*}

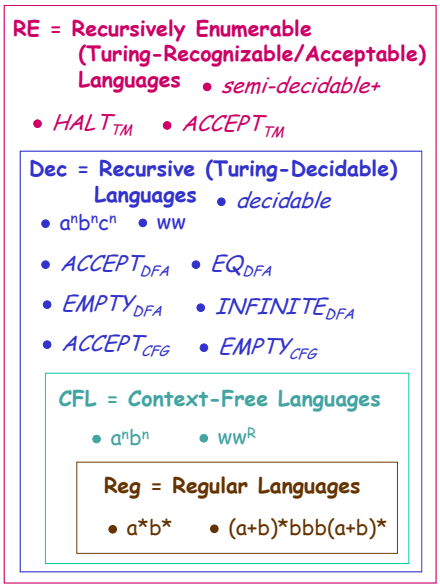


This picture is adapted from p. 145 of Sipser.

#1 Models of Computation: The Chomsky Hierarchy



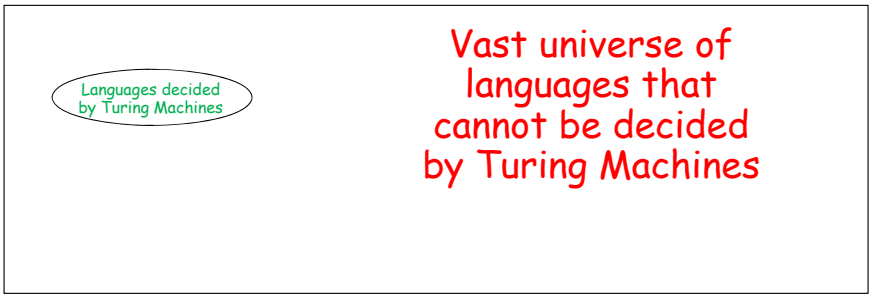
#1 Models of Computation: Language Map



#1 Models of Computation: Forlan

- Forlan is a "calculator" for models of compuations -- a set of modules in the Standard ML programming language that we'll be using in class & assignments to do hands-on experiments with the models of computation we'll be learning
- Created by Alley Stoughton while at Kansas State University.
- See http://alleystoughton.us/forlan/ for:
  - Draft textbook (free online!)
  - Textbook slides
  - Software = Standard ML modules, Java programs (we'll use these in CS235)
  - Alley's KSU course: CIS 570

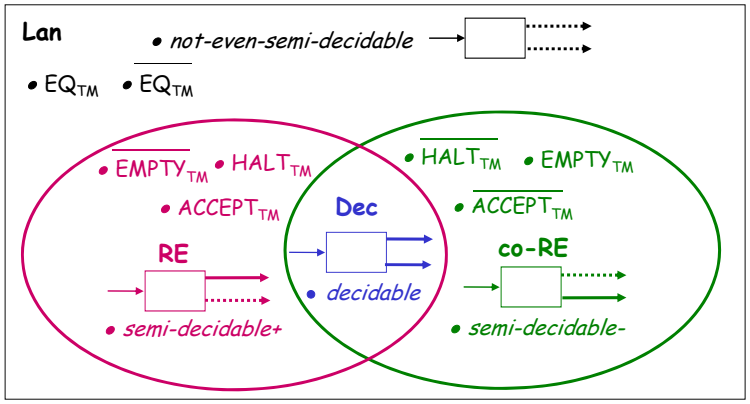
#2 Undecidability: Not Everything is Computable!



The Halting Problem: It is impossible to write a program that always determines whether or not other programs halt.

Rice's Theorem: Any nontrivial question we might ask about programs in general cannot be answered by other programs.

#2: The Landscape of Undecidability



Note: **undecidable** =
semi-decidable+ U semi-decidable- U not-even-semidecidable

#2: With Great Power Comes Great Uncomputability

Turing machines and equivalent models of computation (lambda calculus, Java, SML, etc.) are far more powerful than finite automata and pushdown automata.

But the power is gained via features that can cause programs to loop infinitely. If we want the power, we must live with the looping.

#3 Mathematical Foundations of Computer Science

- Explain computation by using standard mathematical structures:
  - Sets
  - Strings
  - Tuples
  - Relations
  - Functions
  - Languages = sets of strings
  - Trees
  - Graphs
- Use many proof techniques, including
  - Proof by construction
  - Proof by algebra
  - Proof by picture
  - Proof by contradiction (including diagonalization)
  - Proof by induction
- Prove negative results as well as positive results:
  - Pumping lemma for regular languages
  - Pumping lemma for context-free languages
  - Undecidability results

#4 Cool Applications of the Material in this Course

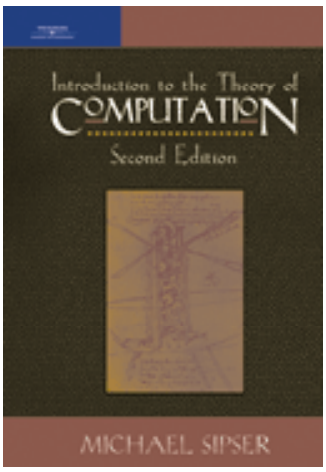
- Using regular expressions in Java, JavaScript, Python, Emacs, etc.
- Using Unix tools like grep, sed, and awk.
- Writing virus detection scanners.
- Designing hardware (CS240).
- Programming language implementation: creating scanners and parsers using tools like PEG parsers, Lex, and Yacc.
- ML programming (appetizer for CS251, CS301).
- Knowing which problems *not* to solve.

Administrivia

- Syllabus, problem sets, office hours, etc. on the CS235 home page: http://cs.wellesley.edu/~cs235
- Weekly problem sets, typically due on Fri. afternoons (start early!)
- I'll try to grade psets in a timely fashion. The next problem set will not be due until at least one day after the previous one has been returned!
- There are three exams:
  - 1. Two 3 or 4 hour self-scheduled take-home midterm exams
  - 2. A 2.5 hour self-scheduled open-book&notes final exam.
- Chelsea Hoover will hold weekly drop-in tutoring hours.

Textbook

Sipser's *Introduction to the Theory of Computation (2nd ed.)*



Great book, but

1. we'll only cover first ~200 pages and
2. it's expensive:

   Bookstore: ~$1?? new
   Amazon: ~$130 new

   (much cheaper used)

I will put copies of book in CS Library (now in SCI E125). Some are 1st ed., which is OK for most purposes.

Other Resources

- Dexter Kozen's *Automata and Computability*
- L.C. Paulson, *ML for the Working Programmer*
- Andrew Appel, *Modern Compiler Implementation in ML*
- A few articles I'll hand out in class

Collaboration Policies

- You ***may*** **talk** with anyone about any nonexam problem, but **must write up** all proofs/programs on your own.
- On every assignment, you ***must*** **list all people with whom you collaborated** on any problem.
- You ***must not*** **look at** anyone else's written proofs/programs.
- You ***must not*** **look at** any solutions from previous semesters of CS235.
- You ***may*** **find other information** in textbooks and on the Internet, but ***must*** **cite** any source you use for solutions.
- I consider violations of the above policies to be **Honor Code violations** and will report them to the Honor Code Council.

Acknowledgment

Many of the assignment problems and slides (esp. ones with clip art) in this course are due to Randy Shull.