CS46 review for midterm exam 2

Midterm exam 2 will focus on material from chapters 3 through 5 of the textbook. While this is mostly Turing machine material, you will still need to remember earlier material. For example, the language A_{CFG} is defined in terms of context-free grammars, but we proved things about it using Turing machines — this kind of problem is fair game for midterm 2. Please refer to the review sheet for midterm 1 if you want to review terminology and techniques from chapters 1 and 2.

You should be able to define, explain, and work with the following terms.

- Turing machine
  - formal definition (7-tuple / state diagram)
  - implementation-level description
  - high-level description
  - configuration
  - halting
  - looping
  - input alphabet, tape alphabet, $\sqcup$
- multi-tape Turing machine
- Turing machine with move left/right/stay
- nondeterministic Turing machine
- Turing-recognizable (and not Turing-recognizable)
- co-Turing-recognizable (and not co-Turing-recognizable)
- decidable (and undecidable)
- enumerator
- the Church-Turing thesis
- computable function
- mapping reduction / mapping reduciblility
- the halting problem

Make sure you understand how your knowledge in this course fits together. This table (below) might help. You should certainly revisit the homeworks and practice problems to help fill in the "examples" rows, as well as to cement your understanding of different techniques we've seen.

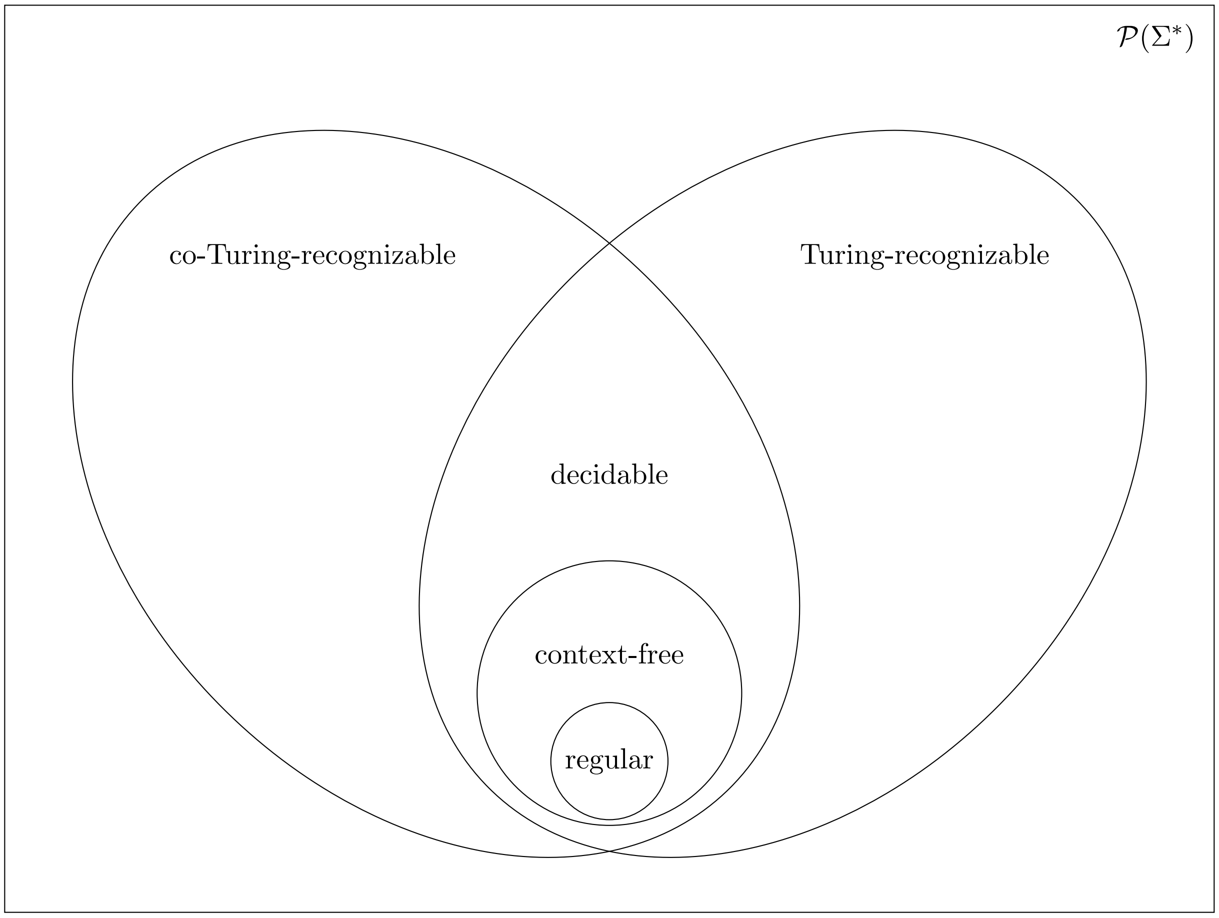
	decidable languages	Turing-recognizable languages	co-Turing-recognizable languages
definition:			
techniques to prove a language *IS* in this class:			
techniques to prove a language *IS NOT* in this class:			
example language(s) in this class:			
example language(s) not in this class:			
this class is closed under operations:			
this class is not closed under operations:			

Reasoning about reductions is important for seeing how different languages relate to each other. Filling in this table is great review/reference. Let A and B be languages.

Suppose $A \leq_m B$.

If we know...	Then we know that...
A is decidable	
A is undecidable	
A is Turing-recognizable	
A is not Turing-recognizable	
A is co-Turing-recognizable	
A is not co-Turing-recognizable	
B is decidable	
B is undecidable	
B is Turing-recognizable	
B is not Turing-recognizable	
B is co-Turing-recognizable	
B is not co-Turing-recognizable	

What is an example language for each set in this diagram?



Behind-the-scenes of Automata Tutor

We fondly recall the days of Automata Tutor.
The way that problems worked on Automata Tutor was:

1. The instructor to submits a canonical DFA D. (It worked the same for NFAs and regular expressions.)
2. The student submits a DFA F.
3. The website checks whether $L(D) = L(F)$. If yes, full credit! If no, then give a counterexample string in one language but not the other.

We now know the algorithms for the check in line (3) — for example, the decider for EQ_{DFA} is a perfectly acceptable thing to plug in to line (3).

Discussion question: why is there no Automata Tutor site for practicing with Turing machines?