CS46 review

The final exam is cumulative. Because the material later in the course built on earlier material, you should make sure that you have mastered *all* the material: this includes definitions, theorems, and techniques from early in the course, which we later used and referred to in more complicated problems later in the course.

You should be able to define, explain, and work with the following terms. For automata, you should be able to work with the formal definitions and the drawings of finite state machines. For Turing machines and pushdown automata, you should be able to describe how they work. (We did not focus on the state-diagram-approach to these machines.) You do not need to be able to regurgitate the proofs of theorems from class, but you *do* need to be able to apply those theorems.

- proof techniques: direct proof, induction, proof by contradiction
- basic math & set theory terminology
  - countable, countably infinite, uncountably infinite
  - reflexive, symmetric, transitive
  - union
  - intersection
  - concatenation
  - Kleene star
  - complement
- regular language
- deterministic finite automaton
- nondeterministic finite automaton
- regular expression
- Pumping Lemma for regular languages
- context-free grammar, parse tree, ambiguous
- context-free language
- pushdown automaton
- Pumping Lemma for context-free languages
- Turing machine
  - formal definition (7-tuple / state diagram)
  - implementation-level description
  - high-level description
  - configuration
  - halting
  - looping (a useful review of all looping behavior is here[1])

[1] All credit to Prof. Edward G. Okie, who wrote this page for his version of a theory of computation course.

  - input alphabet, tape alphabet, $\sqcup$
  - Turing machines with output
- 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 reducibility
- the halting problem
- Rice's Theorem
- big-O notation
- polynomial-time computable function
- polynomial-time mapping reduction
- verifier, certificate
- P, NP, NP-complete, NP-hard
  - conjunctive normal form, variable, literal, clause
  - Satisfiability, 2-Sat, 3-Sat
  - ThreeColoring
  - VertexCover
  - IndependentSet

Make sure you understand how your knowledge in this course fits together. The tables from review sheets for the midterms might help. You should certainly revisit the homeworks, practice problems, and midterm exams to cement your understanding of the different problems and techniques we've seen.

	regular languages	context-free 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:		

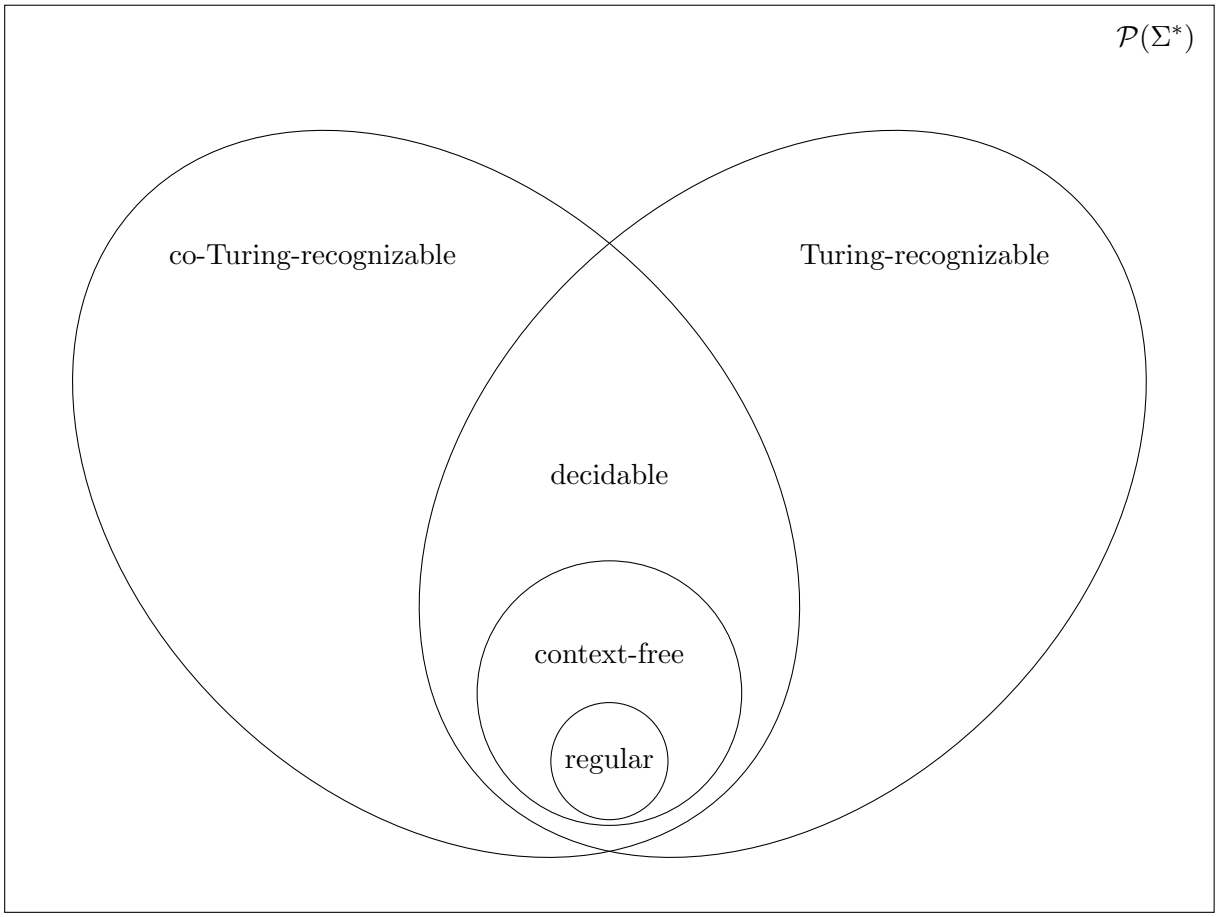
	decidable languages	Turing-recognizable languages	co-Turing-recognizable languages
definition:			
techniques to prove a language *IS* in this class:			
techniques to prove a language *IS NOT* in this class:			
example language(s) in this class:			
example language(s) not in this class:			
this class 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 (from helpful anonymous Piazza post, spring 2017) is great review/reference. Let A and B be languages.

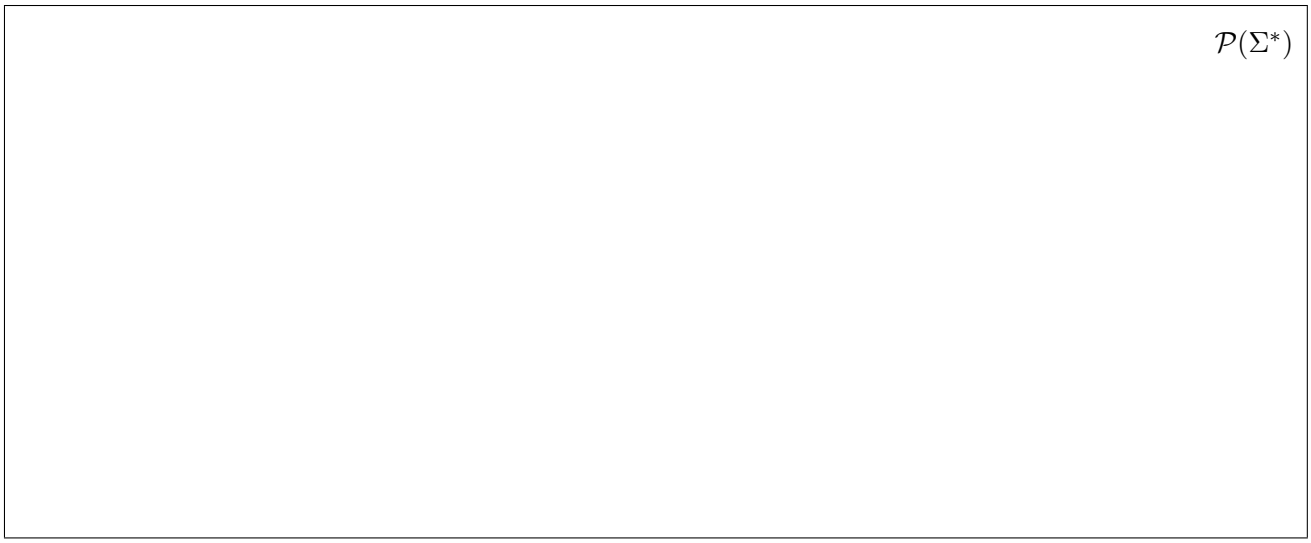
Suppose $A \leq_m B$.

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?



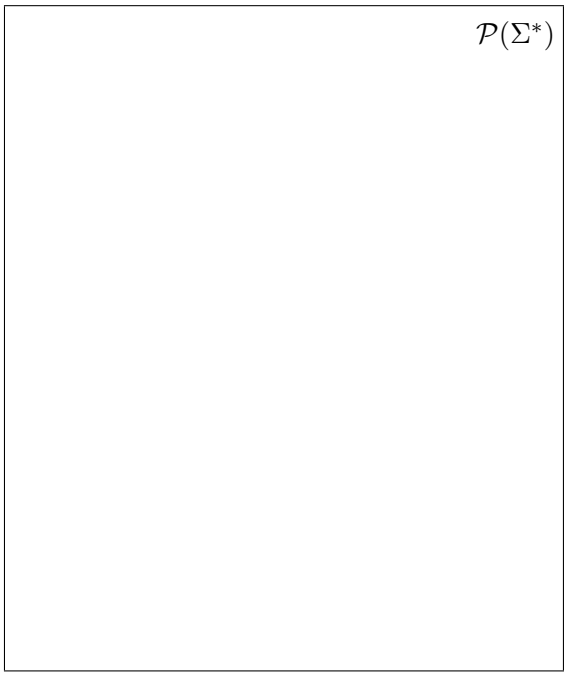
Below, draw the Venn diagram illustrating the (known) relationships between the following sets of languages: decidable languages, P, NP, NP-HARD, NP-COMPLETE, $\mathcal{P}(\Sigma^*)$.



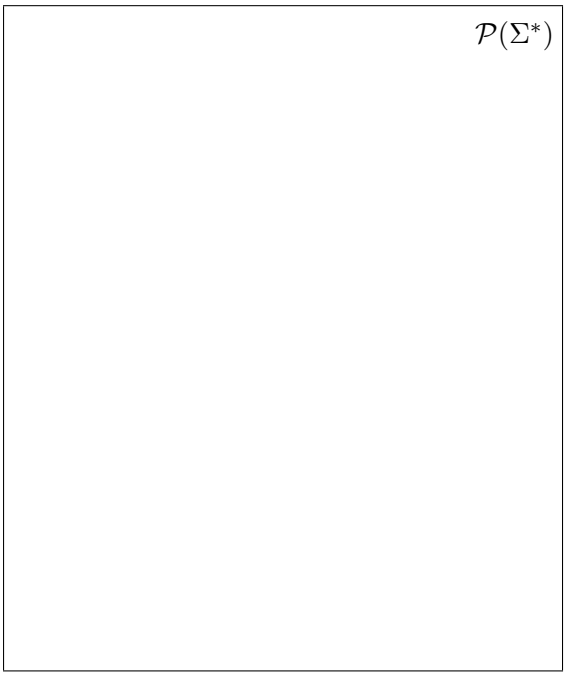
What is an example language for each set in this diagram? (We didn't see examples for every part of the diagram.)

Which parts of the diagram do we not yet know facts about?

If P = NP, then this looks like:



If P $\neq$ NP, then this looks like:



Let A and B be languages.

Suppose $A \leq_p B$.

If we know...	Then we know that...
$A \in$ P	
$A \notin$ P	
$A \in$ NP	
$A \notin$ NP	
$A \in$ NP-COMPLETE	
$A \notin$ NP-COMPLETE	
$A \in$ NP-HARD	
$A \notin$ NP-HARD	
$B \in$ P	
$B \notin$ P	
$B \in$ NP	
$B \notin$ NP	
$B \in$ NP-COMPLETE	
$B \notin$ NP-COMPLETE	
$B \in$ NP-HARD	
$B \notin$ NP-HARD	

(Hint: for many rows in this table, we cannot conclude anything. Refer to the definitions of NP, NP-HARD, and NP-COMPLETE.)

The class P is closed under which operations? The class NP is closed under which operations? How do these differ? Which operations are they *not* closed under?