Theory of Computation, CSCI 438 spring 2021
Final Review, Final is April 22, 3-5pm (6pm if needed)

Regular Languages

- Know the definition of a DFA.
- Be able to convert:

Conversion	Using What
Picture ⇒ M= (…), i.e. the formal definition	Definition
M= (…) ⇒ Picture	

- Be able to convert:

Language description ⇒ DFA	Problem solving
DFA ⇒ Language description	

- Know the signature of δ* for a DFA and be able to use it to define what it means for a DFA to accept a string, and therefore for the DFA to recognize a language
- Know the definition of a regular language
- Be able to prove that regular languages are closed under complementation, union and intersection

Nondeterminism

- Know the definition of an NFA
- Be able to convert:

DFA ⇒ NFA	Theorem: A language is regular iff some NFA recognizes it
NFA ⇒ DFA	

- Be able to prove closure of regular languages under concatenation and star-closure

Regular Expressions

- Know the definition of regular expressions
- Know situations where regular expressions are used
- Be able to convert:

Language description ⇒ Reg-ex	Problem solving
Reg-ex ⇒ Language description	

- Be able to convert:

~~NFA ⇒ Reg-ex~~	Problem solving. Once you have a solution you could prove that your solution works via induction on the length of the string, but I won't ask you to do this.
Reg-ex ⇒ NFA	

- Be able to prove that a language is not regular using the pumping lemma for regular languages

Context-Free Languages

- Know the definition of a context-free grammar
- Know situations where context-free grammars are used
- Know what it means for a variable to be able to "**derive**" a string
- Know the definition of a context free language
- Know what it means for a grammar to be "**ambiguous**" and what it means for a language to be "**inherently ambiguous**"
- Be able to prove that context-free languages are closed under union, concatenation and star-closure
- Be able to convert:

Language description ⇒ CFG	Problem solving
Language description ⇐ CFG	

- Know the definition of a PDA.
- Be able to convert:

Language description ⇒ PDA	Problem solving
Language description ⇐ PDA	

- Know the format of a grammar which is in Chomsky Normal Form.
- Be able to convert:

CFG ⇒ CFG in Chomsky normal form	Theorem 2.29: Any context-free language is generated by a context-free grammar in Chomsky normal form.

- Be able to convert:

~~PDA ⇒ CFG~~	Theorem 2.20: A language is context-free iff some PDA recognizes it. For both directions you only need to be able to do the conversion
PDA ⇐ CFG	

- Be able to prove that a language is not context-free using the pumping lemma for context-free languages

Turing Machines

Turing machines are describable at various levels:

- Completely defined using a single-tape. Give the complete automaton, with circles and arrows.
- Completely defined using some variation of Turing machine (2-tape, k-tape, TM with 'STAY', non-deterministic Turing machine, etc.)
- High level – describe what the Turing machine will do, but don't give the automaton
- Algorithm level – M = "On input <…>
    1. ….

2. …..”

I intend that all Turing Machine exam questions specify the level wanted. If there is any confusion, however, please ask.

- Know the definition of a TM
- Know what it means for a language to be Turing-decidable and Turing-recognizable.
- Be able to convert:

Language description ⇒ TM	Problem solving.

- Given a variation on the definition of a TM, be able to show that it computes the same class of languages as a TM (that is, it computes the class of Turing-recognizable languages).

Variation of a TM ⇒ TM	Use constructive proofs
Variation of a TM ⇐ TM	

- Know the idea behind enumerators, and be able to formally define an enumerator and use it to enumerate a language.
- Know what is meant by an algorithm and be able to write high level algorithms given a problem description.
- Know the Church-Turing Thesis and its implications.

Decidability

- Know what is meant by acceptance problems (A_{DFA}, A_{NFA}, A_{CFG}, etc.), be able to formulate them, and to prove if they are decidable or recognizable.
- Similar to the above, know what is meant by the “empty” (E_{DFA}, E_{NFA}, E_{CFG}, etc.) problems, and the “equivalence” problems ($EQ_{m1,m2}$), be able to formulate them, and to prove if they are decidable or recognizable.
- Be able to prove that A_{TM} is not decidable, yet is Turing-recognizable.
- Know what a Universal Turing machine is.
- Be able to state the halting problem, tells its significance and prove that it is not decidable, yet is Turing-recognizable.
- Know what is meant by co-Turing recognizable
- Be able to prove that if a language is Turing-recognizable and co-Turing recognizable, then it is decidable.
- Be able to use the previous fact to define a language which is not Turing-recognizable.
- Know what it means for two sets to be the same size, even when the sets may be infinite.
- Know what it means for a set to be countable.
- Know Cantor’s diagonalization process and be able to use it to show that the real numbers are not countable.

- Be able to use Cantor's diagonalization process to show that A_{TM} is not decidable.
- Know that the class of Turing-deciders, as well as the class of Turing-recognizers, is enumerable.

Complexity Theory

- Know that computation theory does not depend on the model of computation (as long as that model is "reasonable") while complexity theory does depend on the computation model.
- Be able to determine the complexity of deciding a language on a Turing machine using big-O notation.
- Know and be able to explain why every multi-tape TM that runs in t(n), $t(n) \geq n$, time has an equivalent single-tape Turing machine that runs in $O(t^2(n))$ time.