Topics for Exam 2

- Open books and notes but no electronic aids
- Pages are for Version 3
- #s in “()” refer to homework problems; [] = Exercises

Chap. 2.1 Context Free Grammars (p.102)
- [2.3] Understand the parts of a CFG
- [2.4, 2.6, 2.9, 2.28] (5.1, 5.2, 5.4) Create formal description of a CFG from language description (p.104)
- Describe a language given a CFG
- [2.1] (5.5a) Given a CFG and a string: show a parse tree and/or a derivation
- Know different kinds of derivations (esp. left-most)
  - (p. 107) [2.8,2.27, 2.29] Understand ambiguous grammars
- Know/use/prove rules about combinations of CFLs, esp. via either PDA or CFG
  - [2.16] (5.3) Closed under U, Concat, *
  - [2.2] () Not closed under ∩, complement
  - [2.18a] () CFL ∩ RE = CFL
- [2.23,24] Also () Show two set descriptions of a CFL are equal

Chap. 2.2 Push Down Automata (p.111)
- Understand formal definition of PDA (p. 111)
- Understand role of εs in transition rules (p. 114)
- [2.5] (5.1, 5.4) Create formal description of a PDA from a language description
- [2.11] (5.1, 5.4, 5.5b) Create formal description of a PDA from a CFG (pp. 119-120)
- Given a PDA description and a string, show a derivation sequence
- [2.12] (5.6) Given a PDA, construct CFG (Lemma 2.27) (p. 122)

Chap. 2.3 Non CFG Languages (p.125)
- (5.5c) Be able to show how a string that is known to be in a CFL partitions into substrings so that when pumped, strings are still in L
- [2.34] Be able to estimate pumping length
  - from parameters of a CFG (p127)
  - (6.3) by looking at actual strings
- [2.30-33] Apply CFL pumping lemma to show a language is not CFL (p.126)
  - (6.1, 6.2, 6.8) When language is a mix of terminals
  - (6.7) When language has only 1 terminal

Chap. 3.1. Turing Machines (p. 165)
- Understand formal definition of TM (p. 168)
• [3.5] Understand what a TM can and cannot do at each step
• [3.1,2] Be able to specify configurations a TM goes thru during its computation, esp. accepting and rejecting (p. 1698)
• Understand differences between formal, implementation, hi-level (p. 185)
• [3.8] Write formal description of TM
  • from language description (p. 171-174)
  • (6.9) as a simulator of a FA or PDA
• [3.8] Write implementation description of TM from language description
• Understand difference between a recognizer and a decider (p. 170)
• Be able to define both an informal and a formal TM for either a decider (accept or reject) or (6.10) a computation (e.g. add)
• Understand closure properties of languages
  • [3.15](6.5) decidable languages closed under U, concat, *, ~, ∩
  • [3.16] (6.6) Turing-recognizable languages closed under U, concat, *, ∩, homomorphism

• Chap. 3.2. Variants of TMs (p. 176)
  • Understand variations of TMs and what transition rules for them look like
    • TM that can stay in place
    • Multiple tapes (p. 177)
    • [3.11] Infinite in both directions
    • Nondeterministic (p. 178)
    • [3.10] Write-once TM
    • [3.11] Left reset TM
  • [3.4, 3.6] (6.4) Understand concept of a TM enumerator (p. 180)

• Chap. 3.3. (p. 185) Terminology for describing TMs
  • **Formal**: the complete 7 tuple
  • **Implementation**: English prose of what happens to the tape
  • **High level**: English prose of the algorithm, ignoring details of tape movements