

## Lecture Notes

### Chapter 1: Regular Languages

- DFA vs. NFA, design of DFA and NFA, conversion of NFA to DFA
- Moore-Mealy machines (concept *only*) [handout]
- regular expressions, conversion of regular expression to NFA
- conversion of DFA/NFA to regular expression (through GNFA)
- closure properties of regular languages (how to prove the properties)
- non-regular languages, pumping lemma for regular languages
- Myhill-Nerode Theorem (concept *only*) [handout]

### Chapter 2: Context-Free Languages

- definition of CFG, ambiguous grammar, design of CFG
- conversion of CFG to Chomsky normal form (CNF)
- definition of pushdown automaton (PDA), design of PDA
- conversion of CFG to PDA (conversion of PDA to CFG is *not* required)
- closure properties of context-free language (how to prove the properties)
- pumping lemma for context-free languages
- CYK algorithm (membership testing for CFL) [handout]

### Chapter 3: The Church-Turing Thesis

- definition of Turing machine, Turing-recognizable, Turing decidable
- design of TM (draw the state diagram of a TM)
- variants of Turing machines (enumerator is *not* required)
- Church-Turing Thesis (concept *only*)
- 3 levels of TM description (formal, implementation-level, and high-level)
- configuration of TM, computation of TM (sequence of configurations)
- closure properties of Turing recognizable (decidable) languages
- Minsky's Theorem (concept *only*)

### Chapter 4: Decidability

- acceptance problems:  $A_{DFA}$ ,  $A_{NFA}$ ,  $A_{REG}$ ,  $A_{CFG}$ ,  $A_{TM}$ , etc.
- emptiness testing problems:  $E_{DFA}$ ,  $E_{NFA}$ ,  $E_{REG}$ ,  $E_{CFG}$ ,  $E_{TM}$ , etc.
- equivalence testing problems:  $EQ_{DFA}$ ,  $EQ_{DFA-REG}$ ,  $EQ_{CFG}$ ,  $EQ_{TM}$ , etc.
- decidable languages:  $A_{DFA}$ ,  $A_{CFG}$ ,  $E_{DFA}$ ,  $E_{CFG}$ ,  $EQ_{DFA}$ , etc.
- $A_{TM}$  is *not* decidable (proof *not* required), but it is Turing-recognizable

## Chapter 5: Reducibility

- the halting problem  $HALT_{TM}$  is *not* decidable, but it is Turing-recognizable
- undecidable languages:  $A_{TM}$ ,  $E_{TM}$ ,  $EQ_{CFG}$ ,  $EQ_{TM}$ , etc.
- $A_{TM}$  is Turing-recognizable, but it is *not* co-Turing-recognizable
- $E_{TM}$  is *not* Turing-recognizable, but it is co-Turing-recognizable
- $EQ_{CFG}$  is *not* Turing-recognizable, but it is co-Turing-recognizable
- $EQ_{TM}$  is neither Turing-recognizable nor co-Turing-recognizable
- definition of mapping reducibility, proof of  $A \leq_m B$  by designing TM  $F$  that computes the reducing function of  $A$  to  $B$
- two ways to prove  $HALT_{TM}$  is undecidable: proof by contradiction – a solution to  $HALT_{TM}$  gives a solution to  $A_{TM}$ ; proof by mapping reducibility, i.e.,  $A_{TM} \leq_m HALT_{TM}$

If  $A \leq_m B$ ,

- (1)  $B$  is decidable  $\Rightarrow A$  is decidable (Theorem 5.22, p. 208)
- (2)  $A$  is undecidable  $\Rightarrow B$  is undecidable (Corollary 5.23, p. 208)
- (3)  $B$  is Turing-recognizable  $\Rightarrow A$  is Turing-recognizable (Theorem 5.28, p. 209)
- (4)  $A$  is not Turing-recognizable  $\Rightarrow B$  is not Turing-recognizable (Corollary 5.29, p. 209)

## Chapter 7: Time Complexity

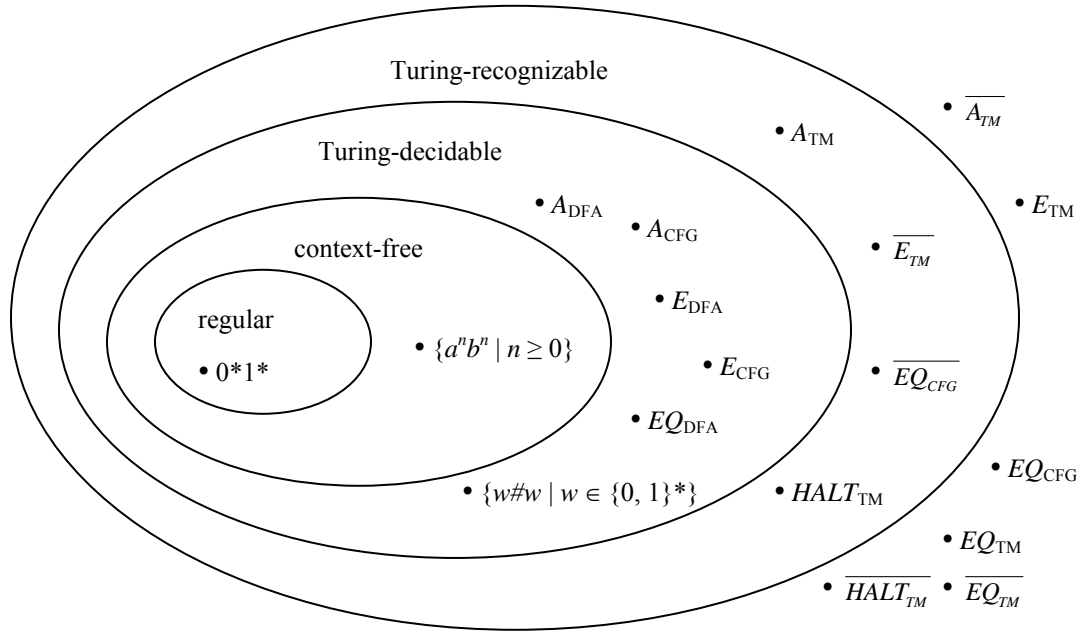
- definition of time complexity or running time of a TM
- Big-O notation, polynomial bounds, exponential bounds
- the class P and NP, proof of a language in P or NP
- closure properties of P and NP languages
- polynomial time mapping reducibility, proof of  $A \leq_p B$  by designing TM  $F$  that computes the reducing function of  $A$  to  $B$  in polynomial time
- definition of NP-complete problems
- examples of NP-complete problems:  $SAT$ ,  $3SAT$ ,  $CLIQUE$
- additional NP-complete problems:  $SUBSET-SUM$ ,  $HAMPATH$ ,  $VERTEX-COVER$  (proof idea *only*, detailed proof for these problems are *not* required)
- two ways to prove a language is NP-complete (by definition and by Theorem 7.36)

If  $A \leq_p B$ ,

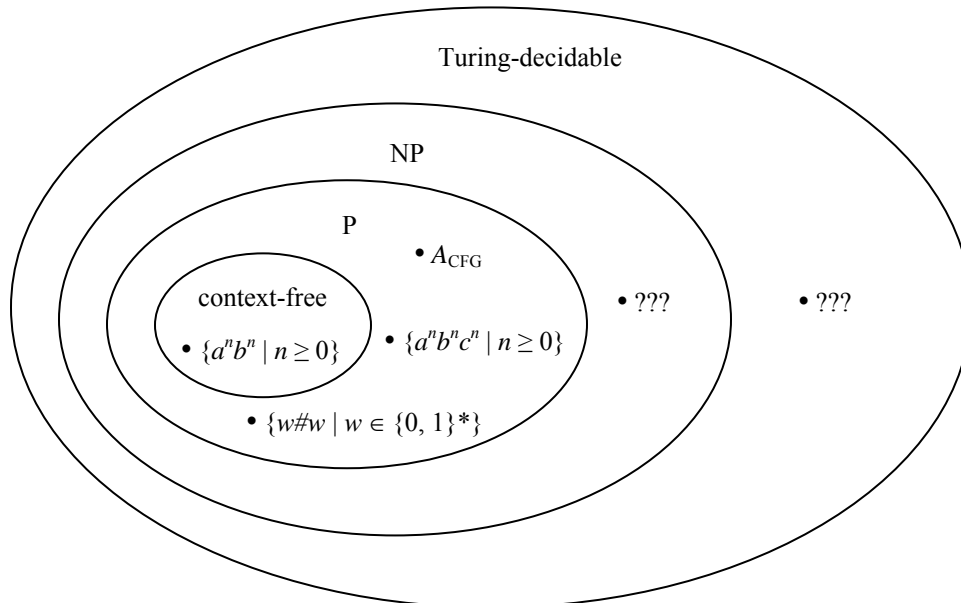
- (1)  $B$  is in P  $\Rightarrow A$  is in P (Theorem 7.31, p. 273)
- (2)  $A$  is NP-complete and  $B$  is in NP  $\Rightarrow B$  is NP-complete (Theorem 7.36, p. 276)

**Note:** In the above, “concept *only*” implies that the related concept will only appear in “True/False” questions; while “*not* required” implies that the mentioned concept/proof will not appear in the exam.

**Language examples in different classes of languages**



**P = NP?**



The class of Context-free languages  $\subset P \subseteq NP \subseteq$  The class of Turing-decidable languages