L is TM-decidable if some TM decides it (& always halts)

(p. 194) **Acceptance problem**: does some FA accept a string?

Can we build a TM that:

- given a representation for some FA and some string,
- tell us if that FA accepts the string, or not
- and do so in finite time
- and never loop

**NOTE: after this section is done, I STRONGLY SUGGEST making your own table of those languages which are decidable.**
• Define \( A_{DFA} = \{<B,w> | \text{B is a DFA that accepts } w\} \)
  • \(<B,w>\) is “encoding” of DFA B and string w in a way that a TM can “interpret” B’s processing of w
  • E.g. \(<B>\) is a list of B’s 5 components
  • \( A_{DFA} \) is set of all encoded DFAs & the strings they accept

• **Is \( A_{DFA} \) decidable?**
  • Does there exist a TM that accepts all members of \( A_{DFA} \) and rejects all other inputs?
    • I.e. does it always halt
  • (p. 194) Theorem 4.1: \( A_{DFA} \) is decidable
  • Proof: M = “On input \(<B,w>\) where B is a DFA & w a string”
    • M receives a tape with \(<B,w>\) on it
    • Determine if representation of \(<B>\) is formatted ok
    • Simulate DFA B on string w
      • Keep track of B’s current state and position into its input w on M’s tape
      • Search for correct transition
      • Update state and index
    • If simulated B ends in accept, accept. If it ends in nonaccept, reject.
    • Note: formatted B always stops after finite # of steps
    • Thus so will TM
• Define $A_{NFA} = \{<B,w>| B \text{ is an NFA that accepts } w\}$
• (p. 195) Theorem 4.2: $A_{NFA}$ is decidable
  • Proof: $N = “\text{On input } <B,w> \text{ where } B \text{ is NFA & } w \text{ a string”}$
    • Convert NFA $B$ into equivalent DFA $C$
    • Encode $C$ and $w$ on tape as $<C,w>$
      • Having a multi-tape TM may be useful
    • Run machine $M$ from Theorem 4.1 on $<C,w>$
    • If $M$ accepts, $N$ accepts, else $N$ rejects
  • Note use of a “subroutine” $M$

• Define $A_{REX} = \{<R,w>| R \text{ is a regex that generates } w\}$
• (p. 196) Theorem 4.3 $A_{REX}$ is decidable
  • Proof: Convert $R$ into an NFA
    • Then run TM $N$
    • If $N$ accepts, then accept, else reject
• Define $E_{DFA} = \{\langle A \rangle | A \text{ is a DFA where } L(A) = \emptyset\}$
  • “E” for “empty”
  • I.e. the set of all DFAs that accept no strings
• (p. 196) Theorem 4.4 $E_{DFA}$ is decidable
  • Proof: Use the BFS algorithm starting on start state of A
    • Mark states that are reachable from start state
    • If any Final State is marked, accept
    • If not, reject
  • Again will halt since only finite # of states in any DFA

• Define $EQ_{DFA} = \{\langle A, B \rangle | A, B \text{ both DFAs & } L(A) = L(B)\}$
  • “EQ” stands for Equivalent
  • I.e. the set of all pairs of DFAs that are equivalent
• (p. 196) Theorem 4.5 $EQ_{DFA}$ is decidable
  • Proof:
    • Construct a new DFA $C$ from $A$ and $B$ that
      • Accepts only those strings that are accepted by either $A$ or $B$, but not both
        • i.e. $L(C) = (L(A) \cap \text{not}(L(B))) \cup (\text{not}(L(A)) \cap L(B))$
        • Called Symmetric Difference
      • If $L(C)$ is empty then $A$ & $B$ gen same language
    • Then use machine from Theorem 4.4
• (p. 198) Decidable Problems re CFLs

• Define $A_{CFG} = \{<G,w>|G \text{ is a CFG that generates } w\}$

• (p. 198) Theorem 4.7 $A_{CFG}$ is a decidable language
  • If $G$ is in Chomsky Normal Form, any derivation of $w$ has $2n-1$ steps, where $|w|=n$
  • TM S
    • Convert $G$ to Chomsky
    • List all derivations with $2n-1$ steps
    • If any generate $w$, accept, else reject

• Define $E_{CFG} = \{<G>|G \text{ is a CFG & } L(G) = \emptyset\}$

• (p. 199) Theorem 4.8 $E_{CFG}$ is a decidable language
  • TM R
    • Mark all terminal symbols in $G$
    • Repeat until no new variables get marked
      • Mark any variable $A$ where $G$ has a rule $A \rightarrow U_1 U_2 \ldots U_k$
        and each symbol $U_i$ has already been marked
    • If start variable not marked, accept, else reject
• Define $\text{EQ}_{\text{CFG}} = \{<G,H>|G \text{ & } H \text{ are CFGs, } \& \text{ } L(G)=L(H)\}$

• Cannot use DFA approach because CFLs not closed under complement or intersection & this is NOT decidable

• (p. 200) Theorem 4.9 **Every CFL is decidable**

• Don’t want to try converting a PDA into an TM
  • Some branches of PDAs computation may go on forever, so TM can’t be a decider

• Proof: Let G be a CFG for A; TM $M_G$ is to decide A
  • Run TM $S$ on $<G,w>$
  • If it accepts, then accept, else reject

• Result: p.201 Fig. 4.10. Following are proper subsets of the next one
  • Regular languages
  • Context-Free languages
  • Decidable Languages
  • Turing-recognizable languages