Decidable Languages
From the Church-Turing Thesis, we can use TMs tell if a problem is computable:
- Can represent problems as languages
- Formulate problems in terms of testing membership in a language
  - If the language is decidable, the problem is decidable
  - Can simulate all previous topics in TMs

Example: acceptance problem
- Test whether a DFA accepts a given string
- Can be expressed as a language, $A_{DFA}$
  - $A_{DFA} = \{ <B,w> | B \text{ is a DFA that accepts input string } w \}$
- Problem of testing whether a DFA $B$ accepts an input $w$ is the same as testing whether $<B,w>$ is a member of the language $A_{DFA}$. 
Example: Deterministic FA

- Present a TM $M$ that decides $A_{DFA}$.

- (Implementation Description) $M = \text{“On input } <B,w>, \text{ where } B \text{ is a DFA and } w \text{ is a string:}"
  
  1. Simulate $B$ on input $w$.
  2. If the simulation ends in an accept state, accept.
     - If it ends in a nonaccepting state, reject.

- TM $M$ exists, $A_{DFA}$ is decidable
  
  - It is possible to test whether a DFA will accept a given string
Example: Nondeterministic FA

- $A_{\text{NFA}} = \{<B, w> \mid B \text{ is an NFA that accepts input string } w\}$
- Present a TM $N$ that decides $A_{\text{NFA}}$.
  - May make use of TM $M$ from previous example

- $N = \text{"On input } <C, w>, \text{ where } C \text{ is an NFA and } w \text{ is a string:"
  - 1. Convert NFA $C$ to DFA $B$
  - 2. Run TM $M$ on $<B, w>$.
  - 3. If $M$ accepts, accept.
    - Otherwise, reject.

- TM $N$ exists, $A_{\text{NFA}}$ is decidable
  - It is possible to test whether a NFA will accept a given string
Example: Regular Expressions

- $A_{\text{REX}} = \{<R,w> \mid R \text{ is a regular expression that generates string } w\}$
- Present a TM $P$ that decides $A_{\text{REX}}$.
  - May make use of TM $N$ from previous example

$P =$ “On input $<R,w>$, where $R$ is a regular expression and $w$ is a string:
  - 1. Convert RE $R$ to NFA $C$
  - 2. Run TM $N$ on $<C,w>$.
  - 3. If $N$ accepts, accept.
    - Otherwise, reject.

- TM $P$ exists, $A_{\text{REX}}$ is decidable
  - It is possible to test whether a regular expression generates a specific string
Example: Emptiness Testing

- All previous problems tested whether an FA accepts a particular string.
  - It is sometimes important to test if an FA accepts anything at all.

\[ E_{\text{DFA}} = \{ <B> \mid B \text{ is a DFA and } L(B) = \emptyset \} \]
- Present a TM \( T \) that decides \( E_{\text{DFA}} \).

\( T = \text{“On input } <B>, \text{ where } B \text{ is a DFA:} \)
- 1. Mark the start state of \( B \).
- 2. Mark any states that can be directly transitioned from a currently marked state
  - Repeat until no new states are marked.
- 3. If no marked states are accept states, accept.
  - Otherwise, reject.

- TM \( T \) exists, \( E_{\text{DFA}} \) is decidable
  - It is possible to test whether a DFA accepts no strings
  - Test whether DFA’s language is empty
Example: Equivalence Testing

- \( \text{EQ}_{\text{DFA}} = \{ <B_1, B_2> \mid B_1, B_2 \text{ are DFAs and } L(B_1) = L(B_2) \} \)
  - Present a TM \( F \) that decides \( \text{EQ}_{\text{DFA}} \).

- Create a DFA \( C \) which recognizes strings that are accepted by either \( B_1 \) or \( B_2 \) but not both.
  - \( L(C) = \) symmetric difference
  - For \( L(B_1) = L(B_2) \), \( L(C) \) must be empty

- \( F = "\text{On input } <B_1, B_2>, \text{ where } B_1, B_2 \text{ are DFAs:} \)
  - 1. Construct DFA \( C \) as described.
  - 2. Run TM \( T \) for emptiness testing
  - 3. If \( T \) accepts, accept.
    - Otherwise, reject.

- TM \( F \) exists, \( \text{EQ}_{\text{DFA}} \) is decidable
  - It is possible to test whether two DFAs are equivalent
Example: Context Free Grammars

- $A_{CFG} = \{ <G, w> \mid G \text{ is a CFG that generates string } w \}$
  - Present a TM $S$ that decides $A_{CFG}$.

- Systematically produce derivations of $G$ until one matches $w$
  - May never halt if correct derivation is never encountered
    - TM will be a recognizer but not a decider
  - If rules are put into Chomsky normal form, $G$ is guaranteed to produce string of the correct length within $2n-1$ steps
    - $n = \text{length of } w$
    - Only need to check all derivations with $2n-1$ steps
      - Finite number of derivations, halting is now guaranteed
Example: Context Free Grammars

- S = “On input < G,w >, where G is a CFG and w is a string:
  1. Convert G to equivalent grammar in Chomsky normal form.
  2. List all derivations with 2n-1 steps
  3. If any derivation generates w, accept
     - Otherwise, reject

- TM S exists, $A_{CFG}$ is decidable
  - It is possible to test if a CFG generates a particular string
Example: CFG Emptiness

- $E_{\text{CFG}} = \{ <G> \mid G \text{ is a CFG and } L(G) = \emptyset \}$
  - Present a TM $R$ that decides $E_{\text{CFG}}$.

- $R = \text{"On input } <G>, \text{ where } G \text{ is a CFG:} $
  - 1. Mark all terminal symbols in $G$
  - 2. Mark any variables that can be substituted with all marked symbols
    - Repeat until no new variables get marked
  - 3. If start variable is not marked, accept.
    - Otherwise, reject.

- TM $R$ exists, $E_{\text{CFG}}$ is decidable
  - It is possible to test if a CFG generates any strings
**Figure 4.10**
The relationship among classes of languages
Example: Every CFL is Decidable

- \( \text{EQ}_{\text{CFG}} = \{ <G_1,G_2> \mid G_1, G_2 \text{ are CFGs and } L(G_1) = L(G_2) \} \)
  - Present a TM \( Q \) that decides \( \text{EQ}_{\text{CFG}} \).

- \( Q = \text{"On input } < G >, \text{ where } G \text{ is a CFG:} \)
  1. Mark all terminal symbols in \( G \)
  2. Mark any variables that can be substituted with all marked symbols
    - Repeat until no new variables get marked
  3. If start variable is not marked, accept.
    - Otherwise, reject.

- TM \( Q \) exists, \( \text{E}_{\text{CFG}} \) is decidable
  - It is possible to test if a CFG generates any strings