Decidable languages

Sipser 4.1 (pages 165–173)

Hierarchy of languages
Describing Turing machine input

• Input is a string over the alphabet $\Sigma$
• What if we want to encode an “object”?  
  – DFA, NFA, PDA, CFG, etc...
  – Use brackets shorthand to indicate that input is encoding of object
    • For example:
    • $<M>$ is the encoding of $M$, where $M$ is a DFA
    • Then, in the machine, simply refer to $M$ as a DFA

Problems concerning regular languages

• $A_{DFA} = \{<B, w>| \ B \text{ is a DFA that accepts input string } w\}$
• $A_{NFA} = \{<B, w>| \ B \text{ is a NFA that accepts input string } w\}$
• $A_{REX} = \{<R, w>| \ R \text{ is a regular expression that generates string } w\}$

• Are these languages decidable?
Deciding regular languages

• Theorem 4.1: $A_{DFA}$ is decidable.
• Proof.
  Let $M = "\text{On input } <B,w>, \text{ where } B \text{ is a DFA:}"
  1. Simulate $B$ on input $w$.
  2. If the simulation ends in an accept state, accept. Otherwise, reject.

What about guessing?

• Theorem 4.2: $A_{NFA}$ is decidable.
• Proof:
  Let $N = "\text{On input } <B,w>, \text{ where } B \text{ is an NFA:}"
  1. Convert NFA $B$ to an equivalent DFA $C$
     using the procedure given in Theorem 1.39
  2. Simulate TM $M$ of Theorem 4.1 on input $<C,w>$
  3. If $M$ accepts, accept. Otherwise, reject."
Deciding regular expressions

• Theorem 4.3: $A_{REX}$ is decidable.
• Proof:
  Let $P =$ "On input $<R,w>$, where $R$ is a regular expression:
  1. Convert regular expression $R$ to an equivalent NFA $A$ using the procedure given in Theorem 1.54
  2. Simulate TM $N$ of Theorem 4.1 on input $<A,w>$
  3. If $N$ accepts, accept. Otherwise, reject."

Can we test for emptiness?

• $E_{DFA} = \{<A> \mid A$ is a DFA and $L(A) = \emptyset\}$
• Theorem 4.4: $E_{DFA}$ is a decidable language.
• Proof:
  Let $T =$ "On input $<A>$, where $A$ is a DFA:
  1. Mark the start state of $A$.
  2. Repeat until no new states get marked:
     • Mark any state that has a transition coming into it from any state that is already marked.
  3. If no accept state is marked, accept; otherwise, reject."
Can we tell if two DFAs are equivalent?

- \( EQ_{\text{DFA}} = \{ <A, B> \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B) \} \)
- Theorem 4.5: \( EQ_{\text{DFA}} \) is a decidable language.
- Proof:
  Let \( F = \) “On input \( <A, B> \), where \( A \) and \( B \) are DFAs:
  1. Construct DFA \( C \) to recognize \( A \ XOR B \).
  2. Run TM \( T \) from Theorem 4.4 on input \( <C> \).
  3. If \( T \) accepts, accept. Otherwise, reject.”

What about context-free languages?

- \( A_{\text{CFG}} = \{ <G,w> \mid G \text{ is a CFG that generates } w \} \)
- Theorem 4.7: \( A_{\text{CFG}} \) is decidable.
- Proof:
  Let \( S = \) “On input \( <G,w> \), where \( G \) is a CFG:
  1. Convert \( G \) to an equivalent grammar in Chomsky normal form
  2. List all derivations with \( 2n-1 \) steps, where \( n \) is the length of \( w \)
  3. If any of these derivations generate \( w \), accept. Otherwise, reject.”
Emptiness... again

- $E_{CFG} = \{<G> \mid G \text{ is a CFG and } L(G) = \emptyset\}$
- Theorem 4.8: $E_{CFG}$ is decidable.
- Proof:
  Let $R = \text{"On input } <G>, \text{ where } G \text{ is a CFG:}\$
  1. Mark all terminal symbols in $G$.
  2. 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 $U_1, U_2, \ldots, U_k$ has already been marked
  3. If the start variable is not marked, accept; otherwise, reject.

Can we tell if two CFGs are equivalent?

- $EQ_{CFG} = \{<G, H> \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$
- Is $EQ_{DFA}$ a decidable language?
- Is something wrong with this proof:
  Let $W = \text{"On input } <G,H>, \text{ where } G \text{ and } H \text{ are CFGs:}\$
  1. Construct CFG $F$ to recognize $L(G) \ XOR \ L(H)$. 
  2. Run TM $R$ from Theorem 4.8 on input $<F>$. 
  3. If $R$ accepts, accept. Otherwise, reject."
But, the good news is...

- Theorem 4.9: Every context-free language is decidable.
- Proof:
  Let $G$ be a CFG for $A$. We design a TM $M_G$ that decides $A$ as follows.
  
  $M_G = \text{"On input } w:\n 1. \text{ Run TM } S \text{ from Theorem 4.7 on input } <G,w>.$
  2. \text{ If this machine accepts, } accept. \text{ Otherwise, } reject."