CSCI 2670
Introduction to Theory of Computing

October 19, 2004
Agenda

• Last week
  - Variants of Turing machines
  - Definition of algorithm

• This week
  - Chapter 4
    • Decidable and undecidable languages
    • The halting problem
Announcements

• Homework due next Tuesday (10/26)
  - 4.2, 4.5, 4.7, 4.11, 4.14, 4.20
Decidable languages

• A language is decidable if some Turing machine decides it
  - Every string in $\Sigma^*$ is either accepted or rejected
• Not all languages can be decided by a Turing machine
Some decidable languages

- $A_{\text{DFA}} = \{ <B,w> \mid B \text{ is a DFA that accepts input string } w \}$
- $A_{\text{NFA}} = \{ <B,w> \mid B \text{ is an NFA that accepts input string } w \}$
- $A_{\text{REX}} = \{ <R,w> \mid R \text{ is a regular expression that generates string } w \}$
- $E_{\text{DFA}} = \{ <A> \mid A \text{ is a DFA and } L(A) = \emptyset \}$
One more decidable regular language

- \( EQ_{\text{DFA}} = \{ \langle A, B \rangle \mid A \text{ and } B \text{ are DFA's and } L(A) = L(B) \} \)

**Theorem:** \( EQ_{\text{DFA}} \) is a decidable language

**Proof:** Consider the following language

\[ (L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B)) \]
One more decidable regular language

\[(L(A) \cap L(B)) \cup (\overline{L(A)} \cap \overline{L(B)})\]
One more decidable regular language

\[ \text{EQ}_{\text{DFA}} = \{ <A, B> \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B) \} \]

**Theorem:** \( \text{EQ}_{\text{DFA}} \) is a decidable language

**Proof:** Consider DFA \( C \) that accepts
\[
L(C) = (L(A) \cap L(B)) \cup (L(A) \cap L(B))
\]

How do we know such a DFA exists?

If \( L(C) = \emptyset \), then \( L(A) = L(B) \)
Question

• How would we show that the following language is decidable?

\[ \text{ALL}_{\text{DFA}} = \{ <A> \mid A \text{ is a DFA that recognizes } \Sigma^* \} \]
Deciders and CFG’s

• Consider the following language
  \[ A_{CFG} = \{ <G, w> \mid G \text{ is a CFG that generates string } w \} \]

• Is \( A_{CFG} \) decidable?
  - Problem: How can we get a TM to simulate a CFG?
    • Must be certain CFG tries a finite number of steps!
  - Solution: Use Chomsky normal form
Chomsky normal form review

• All rules are of the form
  \[ A \rightarrow BC \]
  \[ A \rightarrow a \]
• Where \( A, B, \) and \( C \) are any variables
  (\( B \) and \( C \) cannot be the start variable)
  \[ S \rightarrow \epsilon \]
• is the only \( \epsilon \) rule, where \( S \) is the start variable
How many steps to generate $w$?

- If $|w| = 0$
  - 1 step
- If $|w| = n > 0$?
  - $2n - 1$ steps
TM simulating \(A_{\text{CFG}}\)

1. Convert \(G\) into Chomsky normal form
2. If \(|w| = 0\)
   - If there is an \(S \rightarrow \varepsilon\) rule, accept
   - Otherwise, reject
3. List all derivations with \(2|w|-1\) steps
   - If any generate \(w\), accept
   - Otherwise, reject
Empty CFG’s

• Consider the following language
  \( E_{\text{CFG}} = \{<G> \mid G \text{ is a CFG and } L(G) = \emptyset\} \)

• Theorem: \( E_{\text{CFG}} \) is decidable

• Can we use the TM in \( A_{\text{CFG}} \) to prove this?
  
  – No. There are infinitely many possible strings in \( \Sigma^* \)
  
  – Instead, we need to check if there is any way to get from the start variable to some string of terminals
Work backwards

1. **Mark all terminals**

2. **Repeat until no new variables are marked**
   - Mark any variable $A$ if $G$ has a rule $A \rightarrow U_1U_2...U_k$ where $U_1, U_2, ..., U_k$ are all marked

- If $S$ is marked, reject
- Otherwise, accept
What about $\text{EQ}_{\text{CFG}}$?

- Recall for $\text{EQ}_{\text{DFA}}$, we considered $(L(A) \cap L(B)) \cup (L(A) \cap L(B))$
- Will this work for CFG's?
  - No. CFG's are not closed under complementation or intersection
- $E_{\text{CFG}}$ is not a decidable language!
Decidability of CFL’s

Theorem: Every context-free language $L$ is decidable

Proof: For each $w$, we need to decide whether or not $w$ is in $L$. Let $G$ be a CFG for $L$. This problem boils down to $A_{CFG}$, which we showed is decidable.

Question: Why don’t we just make a TM that simulates a PDA?
Relationship of classes of languages

- Regular
- Context-free
- Decidable
- Turing-recognizable