CMPSC230
Lesson 12: Decidable Languages

Prof. John Wenskovitch
10/21/2014
Last Time

• Turing Machine Variants
  – Stay-Put Turing Machine
  – Multitape Turing Machine
  – Nondeterministic Turing Machine
  – Enumerator
Today

• Decidable Languages:
  – What is a decidable language?
  – What are some examples
Turing Machine Terminology

• A language **recognized** by a Turing machine is called **Turing-recognizable**.
• A language **decided** by a Turing machine is called **Turing-decidable**.
• A **recognizer** reads all strings in a language.
• A **decider** determines whether to accept or reject on all strings in a language.
Acceptance Problem for DFAs

• Will a given DFA accept a given input string?

• $A_{DFA}$ contains the encodings of all DFAs together with strings that those DFAs accept:

$$A_{DFA} = \{ \langle B, w \rangle | B \text{ is a DFA that accepts input string } w \}$$

• The problem of testing whether DFA $B$ accepts input string $w$ is the same as the problem of testing whether $\langle B, w \rangle$ is a member of $A_{DFA}$. 
Acceptance Problem for DFAs

• Is $A_{DFA}$ decidable? Present a TM that decides $A_{DFA}$.

• Proof:

  $M = \text{"On input } \langle B, w \rangle, \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." ■

• Implementation details: List the components of $B$ within $M$ as $(Q, \Sigma, \delta, q_0, F)$, update current state by keeping track of the position in $w$ and using $\delta$. Make the accept/reject decision when all symbols in $w$ have been read.
How About $A_{NFA}$?

- **Proof:**

  $N = \text{“On input } \langle B, w \rangle, \text{ where } B \text{ is an NFA and } w \text{ is a string:} \overline{\phantom{ }\phantom{}}$

  1. Convert NFA $B$ into an equivalent DFA $C$, using the procedure for the conversion given in Chapter 1.

  2. Run TM $M$ from the last slide on input $\langle C, w \rangle$ and do same.” □

- “and do same” = If TM $A$ accepts, then let TM $B$ accept; if TM $A$ rejects, then let TM $B$ reject.
How About $A_{REX}$?

• **Proof:**

\[ P = \text{“On input } \langle R, w \rangle, \text{ where } R \text{ is an regular expression and } w \text{ is a string:} \]

1. Convert regular expression $R$ into an equivalent NFA $A$, using the procedure for the conversion given in Chapter 1.

2. Run TM $N$ from the last slide on input $\langle A, w \rangle$ and do same.”

Emptiness Problem

• Determine whether or not a given finite state machine accepts any strings at all.

\[ E_{DFA} = \{ \langle A \rangle | \text{A is a DFA and } L(A) = \emptyset \} \]

• A DFA accepts a string iff reaching an accept state from the start state by traveling along the arrows of the DFA is possible.
Emptiness Problem for DFAs

• Is $E_{DFA}$ a decidable language?

• **Proof:**

  $T$ = “On input $\langle A \rangle$, 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.” ■
Equality Problem

• Determine whether or not two DFAs recognize the same language.

\[ E_{Q_{DFA}} = \{ A, B \} \]

• Symmetric Difference:

\[ L(C) = (L(A) \cap L(B)) \cup (L(A) \cap L(B)) \]
Equality Problem for DFAs

• Why is symmetric difference useful?
  – If \( L(C) = \emptyset \), then \( L(A) = L(B) \).

• Proof:

  \[ F = \text{“On input } \langle A, B \rangle, \text{ where } A \text{ and } B \text{ are DFAs:} \]
  
  1. Construct DFA \( C \) as described.
  2. Run TM \( T(E_{DFA}) \) on input \( \langle C \rangle \) and do same.” \]

\[ \blacksquare \]
So What About Context-Free Languages?

\[ A_{CFG} = \{ \langle G, w \rangle \mid G \text{ is a CFG that generates string } w \} \]

- We can’t just go through all possible derivations generated by \( G \), because it’s possible to try an infinite number of derivations.
  - This would be a recognizer, not a decider.
  - To make it a decider, we need to ensure that the algorithm only tries a finite number of derivations.
Acceptance Problem for CFGs

• **Proof:**

\[ S = \text{"On input } \langle G, w \rangle, \text{ where } G \text{ is a CFG and } w \text{ is a string:} \]

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 \( w \); except if \( n=0 \), then instead list all derivations with just one step.
3. If any of these derivations generate \( w \), accept; if not, reject."
So What About Context-Free Languages?

\[ E_{CFG} = \{ \langle G \rangle | \text{ } G \text{ is a CFG and } L(G) = \emptyset \} \]

• We can’t use the \( A_{CFG} \) machine, because to determine if a language is empty, the machine would have to test all possible \( w \)’s, one by one... but there are infinitely many \( w \)’s to try.

• Instead, to determine if the language of a grammar is empty, we need to test if the start variable can generate a string of terminals.
Emptiness Problem for CFGs

• Proof:

\[ R = \text{“On input } \langle G \rangle, \text{ where } G \text{ is a CFG:} \]
1. Mark all terminal symbols in \( G \).
2. Repeat until no new variables get marked:
   Mark any variable \( A \) where \( G \) has a rule \( A \mapsto U_1 U_2 \ldots U_k \), and each symbol on the right has already been marked.
3. If the start symbol is not marked, accept; otherwise, reject.” □
So What About Context-Free Languages?

\[ EQ_{CFG} = \{ \langle G, H \rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H) \} \]

- With finite state machines, we used the decision procedure for \( EQ_{DFA} \) to prove that \( EQ_{DFA} \) was decidable. So, same idea, right?
- Wrong! The class of context-free languages is not closed under complementation or intersection. This approach won’t work.
- Turns out, \( EQ_{CFG} \) is not decidable!
One More Theorem

**Theorem:** Every context-free language is decidable by a Turing machine.

**Proof:**

Let $G$ be a CFG for language $A$, and design a Turing Machine $M_G$ that decides $A$.

$$M_G = \text{"On input } w:\"$$

1. Run $A_{CFG}$ on input $\langle G, w \rangle$ and do same.” ■
One More Theorem

- Turing-recognizable
- Decidable
- Context-free
- Regular
Any Questions?

HOMEWORK (due 10/30)
4.1, 4.2, 4.3, 4.4