Decidable Languages

Lecture 30
Section 4.1

Robb T. Koether

Hampden-Sydney College

Wed, Nov 5, 2008
Outline

1. Homework Review
2. Decidable Languages
3. Decidable Problems Concerning DFAs and NFAs
4. Decidable Problems Concerning CFGs and PDAs
5. Assignment
Exercise 3.19, page 162.

Show that every infinite Turing-recognizable language has an infinite decidable subset.
Solution

- Let $L$ be an infinite Turing-recognizable language.
- Then it has an enumerator $E$ that enumerates $L$, but not in lexicographical order.
Homework Review

Solution

- Design an enumerator $E'$ as follows.
  - Start up $E$, letting it write strings on Tape 2.
  - Copy the first such string onto Tape 1.
  - Whenever a new string is written on Tape 2, compare it to the last one copied onto Tape 1.
  - If it is greater in lexicographical order, then copy it onto Tape 1.
  - If not, then ignore it and let $E$ continue.
Solution

- Because $L$ is infinite, $E$ will forever be writing strings on Tape 2 that are greater than any string copied onto Tape 1 so far.

- Therefore, an infinite number of strings will be copied onto Tape 1.

- Define the language $L'$ to be the set of strings copied onto Tape 1.

- Obviously, $L' \subseteq L$.

- Furthermore, $L'$ is decidable because it has an enumerator $E'$ that enumerates its members in lexicographical order.
We have seen that

\[ \{ \langle G \rangle \mid G \text{ has an Euler circuit} \} \]

is a decidable language.

That is, we can create a Turing machine that will read \( \langle G \rangle \) and give a yes or no answer every time.
In a similar way, any decision problem $P$ can be represented as a language:

$$L_P = \{ \langle I \rangle \mid I \text{ is an instance of problem } P \text{ for which the answer is yes} \}.$$
The Acceptance Problem for DFAs

Definition (The Acceptance Problem for DFAs)

Given a DFA $M$ and a string $w$, does $M$ accept $w$?

- The language is

$$A_{DFA} = \{ \langle M, w \rangle \mid \text{DFA } M \text{ accepts string } w \}.$$
The Acceptance Problem for DFAs

- To decide the problem, we build a Turing machine $M'$ that simulates $M$ on input $w$.
- Then we run $M'$ on input $\langle M, w \rangle$.
- After a finite number of steps, $M'$ halts in either an accept state or a reject state.
- If $M'$ halts in an accept state, then $M$ accepts $w$.
- If $M'$ halts in a reject state, then $M$ rejects $w$. 
Decidable Languages

The Acceptance Problem for DFAs

Theorem

\[ A_{DFA} \text{ is decidable.} \]
The Acceptance Problem for NFAs

Definition (The Acceptance Problem for NFAs)

Given an NFA $M$ and a string $w$, does $M$ accept $w$?

- The language is

$$A_{NFA} = \{ \langle M, w \rangle \mid \text{NFA } M \text{ accepts string } w \}.$$
The Acceptance Problem for NFAs

- The strategy is to convert NFA $M$ to a DFA $D$ and then run the previous algorithm on $\langle D, w \rangle$.
- This is an example of a reduction of one problem to another problem (from $A_{\text{NFA}}$ to $A_{\text{DFA}}$).
The Acceptance Problem for NFAs

Theorem

\[ A_{NFA} \text{ is decidable.} \]
The Emptiness Problem for DFAs

**Definition (The Emptiness Problem for DFAs)**

Given a DFA $M$, is the language of $M$ empty. That is, does $M$ reject every word in $\Sigma^*$?

- The language is

$$E_{\text{DFA}} = \{ \langle M \rangle \mid L(M) = \emptyset \}.$$
The Emptiness Problem for DFAs

- The strategy is to do a breadth-first search of the state diagram for an accept state, starting from the start state.
The Emptiness Problem for DFAs

- If the start state is an accept state, then reject $\langle M \rangle$.
- If not, then mark the start state as inspected.
- Then inspect every state that is reachable in one transition from the start state.
- If any is an of them is an accept state, then reject $\langle M \rangle$.
- If not, then mark them as inspected.
- Continue in the same manner with the states that are reachable from those states in one transition and that have not been marked.
The Emptiness Problem for DFAs

- This procedure will eventually terminate when it can reach only states that are already marked.
- If no marked state is an accept state, then accept $\langle M \rangle$. 
The Emptiness Problem for DFAs

**Theorem**

$E_{DFA}$ is decidable.
**The Equivalence Problem for DFAs**

**Definition (The Equivalence Problem for DFAs)**

Given two DFAs $A$ and $B$, do they have the same language? That is, does $L(A) = L(B)$?

The language is

$$EQ_{DFA} = \{ \langle A, B \rangle \mid L(A) = L(B) \}.$$
The Equivalence Problem for DFAs

- The strategy is to follow the algorithm to build the DFA $M$ whose language is

\[
\left( L(A) \cap \overline{L(B)} \right) \cup \left( \overline{L(A)} \cap L(B) \right).
\]
Then solve the Emptiness Problem for $M$.

- If $L(M) = \emptyset$, then $L(A) = L(B)$.
- If $L(M) \neq \emptyset$, then $L(A) \neq L(B)$.

This is a second example of a reduction (from $EQ_{DFA}$ to $E_{DFA}$).
The Equivalence Problem for DFAs

Theorem

$EQ_{DFA}$ is decidable.
The Derivation Problem for CFGs

Definition

Given a CFG \( G \) and a word \( w \), can \( w \) be derived from \( G \)? That is, is \( w \in L(G) \)?

- The language is

\[
D_{\text{CFG}} = \{ \langle G, w \rangle \mid w \in L(G) \}.
\]
The Derivation Problem for CFGs

- The strategy is first to convert the grammar $G$ to an equivalent grammar $G'$ in Chomsky Normal Form.
- If the string $w$ has length $n$, then it is in $L(G)$ if and only if it can be derived from $G'$ in exactly $2n - 1$ steps from $G$.
- In fact, the first $n - 1$ steps must use rules of the form $A \rightarrow BC$ and the last $n$ steps must use rules of the form $A \rightarrow a$. 
The Derivation Problem for CFGs

- The Turing machine systematically tests every possible such derivation.
- There are only a finitely many such derivations, so the process must terminate.
- If $w$ is derived, then accept $\langle G, w \rangle$.
- If $w$ is not derived, then reject $\langle G, w \rangle$. 
The Derivation Problem for CFGs

Theorem

\( D_{CFG} \) is decidable.
The Acceptance Problem for PDAs

Definition
Given a PDA $M$ and a string $w$, does $M$ accept $w$?

- The language is

$$A_{PDA} = \{ \langle M, w \rangle \mid \text{PDA } M \text{ accepts string } w \}.$$
The Acceptance Problem for PDAs

- The strategy is to apply the algorithm that converts a PDA $M$ to an equivalent context-free grammar $G$.
- Then solve the Derivation Problem for $\langle G, w \rangle$.
- This is a third example of a reduction (from $A_{PDA}$ to $D_{CFG}$).
The Acceptance Problem for PDAs

Theorem

$A_{PDA}$ is decidable.
Homework

- Read Section 4.1, pages 165 - 173.
- Exercises 1, 2, 3, 4, pages 182 - 183.
- Problems 9, 10, 11, 12, 13 page 183.