Decidability
Decidable Problems About Regular Languages

The Acceptance Problem for DFA

Define $A_{\text{DFA}}$ to be:

$$\{\langle B, w \rangle \mid B \text{ is a DFA that accepts input string } w\}.$$ 

**Theorem.** $A_{\text{DFA}}$ is decidable.

**Proof** A Turing machine can, given an input $x$, try to decode $x$ into an NFA $B$ and a string $w$. If the decoding is successful then it can test whether $B$ accepts $w$ by **simulating** $B$ on $w$. 

CSC527 Chapter 4, Part 1 © 2012 Mitsunori Ogihara
How This Can Be Done

- After checking the legitimacy of encoding, our Turing machine writes on its second tape the input $w$ (as an encoded form).
- Our machine starts simulating $M$, using the second tape as the tape of $M$ by looking up information about $M$’s action in the first tape and using a tape symbol encoding scheme consistent with the input $x$.

When $M$ terminates, our machine terminates accordingly.
Define $A_{\text{NFA}}$ to be:

$$\{ \langle B, w \rangle \mid B \text{ is an NFA that accepts input string } w \}.$$ 

**Theorem.** $A_{\text{NFA}}$ is decidable.

**Proof**  Given an input $x$, try to decode $x$ into an NFA $B$ and a string $w$. If “successful” then:

1. Convert $B$ to a DFA $C$.
2. Run the machine for $A_{\text{DFA}}$ on $\langle C, w \rangle$. If the machine accepts, then accept; otherwise reject.
Define $A_{\text{REX}}$ to be:

$$\{\langle R, w \rangle \mid R \text{ is a regular expression that produces } w \}.$$

**Theorem.** $A_{\text{REX}}$ is decidable.

**Proof**  Given an input $x$, try to decode $x$ into a regular expression $R$ and a string $w$. If “successful” then:

1. Convert $R$ to a DFA $C$.
2. Run the machine for $A_{\text{DFA}}$ on $\langle C, w \rangle$. If the machine accepts, then accept; otherwise reject.
The Emptiness Problem for DFA

Define $E_{\text{DFA}} = \{ \langle A \rangle \mid A \text{ is a DFA that accepts no string} \}$.

**Theorem.** $E_{\text{DFA}}$ is decidable.

**Proof**  Given an input $x$, try to decode a DFA $A$ out of $x$. If “successful” then:

1. **Mark the start state** of $A$.
2. **Repeat until no new states are marked:**
   - Mark any unmarked state that has a transition from a marked state
3. Accept if **no final state is marked**; reject otherwise.
The Equivalence Problem for DFA

Define $EQ_{DFA}$ to be:
$$\{ \langle A, B \rangle \mid A \text{ and } B \text{ are DFA and accept the same language } \}.$$  

**Theorem.** $EQ_{DFA}$ is decidable.

**Proof**  Given a string $x$, try to decode $x$ into a pair of DFAs $A$ and $B$. If “successful” then construct a DFA $C$ that accepts the **symmetric difference** of $L(A)$ and $L(B)$,
$$\bigl( L(A) \cap \overline{L(B)} \bigr) \cup \bigl( \overline{L(A)} \cap L(B) \bigr),$$
and test the emptiness of $L(C)$.
The Acceptance Problem for CFG

Define \( A_{\text{CFG}} = \{ \langle G, w \rangle \mid G \text{ is a CFG that generates } w \} \).

**Theorem.** \( A_{\text{CFG}} \) is decidable.

**Proof** Given an input \( x \), try to decode \( x \) into a CFG \( G \) and a string \( w \). If “successful” then:

1. Convert \( G \) to an equivalent Chomsky normal form grammar \( G' \).
2. List all derivations with \( 2n - 1 \) steps, where \( n = |w| \).
3. If any of the listed derivations generate \( w \), then accept; otherwise, reject.
The Emptiness Problem for CFG

Define \( E_{\text{CFG}} = \{ \langle G \rangle \mid G \text{ is a CFG such that } L(G) = \emptyset \} \).

**Theorem.** \( E_{\text{CFG}} \) is decidable.

**Proof**  Given \( x \), first try to decode a grammar \( G \) out of it. If “pass” then test the ability of generating terminal strings:

1. **Mark all the terminals.**
2. Repeat the following until no new symbols are marked:
   - Mark any variables \( A \) with a **production** \( A \rightarrow w \) such that all symbols in \( w \) are marked.
3. **Accept** if the start symbol is marked; **reject** otherwise.
Theorem. Every context-free language is decidable.

Simulation of a PDA may not halt.

Proof Use the machine $M$ for $A_{CFG}$. Let $G$ be a fixed CFG. The machine for $L(G)$, on input $w$,

1. run $\langle G, w \rangle$ on $M$, and
2. accepts if $M$ accepts and rejects otherwise.