The Acceptance Problem for NFA

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

**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.

---

The Acceptance Problem for Regular Exp.

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

**Theorem.** $A_{\text{REG}}$ 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.

---

Decidability Problems About Regular Languages

The Acceptance Problem for DFA

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

Here we assume a fixed encoding scheme for $B$ and $w$.

**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 $w$. 


**The Acceptance Problem for CFG**

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

**Theorem.** $A_{CFG}$ is decidable.

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

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 DFA**

Define $E_{DFA} = \{ (A) \mid A$ is a DFA that accepts no string $\}$.  

**Theorem.** $E_{DFA}$ 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 $EQ_{DFA} = \{ (A, B) \mid A$ and $B$ are DFA that 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 $L(B)$, 

$$ L(A) \cap \overline{L(B)} \cup \overline{L(A) \cap L(B)}, $$

and test the emptiness of $L(C)$.  

**The Emptiness Problem for CFG**

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

**Theorem.** $E_{CFG}$ 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 $(G, w)$ on $M$, and
2. accepts if $M$ accepts and rejects otherwise.