Decidability

Decidable Problems About Regular Languages

The Acceptance Problem for DFA

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

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

Theorem. A_{DFA} *is decidable.*

Proof A Turing machine can, given an input x, try to decode x into a DFA 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 NFA

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

Theorem. *A_{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_{DFA} on $\langle C, w\rangle$. If the machine accepts, then **accept**; otherwise **reject**.

∎

(Or we could use a 3-tape TM, as in simulating a NTM.)

The Acceptance Problem for Regular Exp.

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

Theorem. *A_{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_{DFA} on $\langle C, w\rangle$. If the machine accepts, then **accept**; otherwise **reject**.

∎

The Emptiness Problem for DFA

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

Theorem. E_{DFA} *is decidable.*

Proof Given an input x, try to decode x as a DFA A. 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

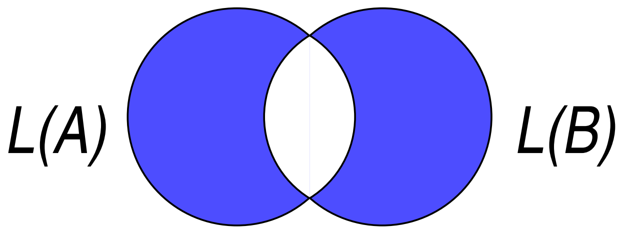
Define $EQ_{\mathrm{DFA}} = \{\langle A, B\rangle \mid A$ and B are DFA that accept the same language $\}$.

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 Acceptance Problem for CFG

Define $A_{\mathrm{CFG}} = \{\langle G, w\rangle \mid G$ 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:

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

Theorem. *E_{CFG} is decidable.*

Proof Given x, first try to decode it as a grammar G. If "successful" then test G's ability to generate terminal strings:

1. **Mark all the terminals**.
2. Repeat the following until no new symbols are marked:
   - Mark any variables A with **a production $A \to w$ such that all symbols in w are marked**.
3. **Accept** if the start symbol is not marked; **reject** otherwise.

∎

Context-Free Languages are Decidable

Theorem. *Every context-free language is decidable.*

Simulation of a PDA may not halt, so this wouldn't be a good approach.

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

1. runs M with input $\langle G, w \rangle$, and
2. **accepts** if M accepts and **rejects** otherwise.

$\blacksquare$