Figure 1: A finite automaton $M_1$ (on left) and a finite automaton $M_2$ (on right).

1. Define:

$$A_{NFA} = \{ < B, w > \mid B \text{ is an NFA that accepts input string } w \},$$

$$EQ_{DFA} = \{ < B, C > \mid B, C \text{ are DFAs and } L(B) = L(C) \}. $$

Recall that DFA (respectively, NFA) stands for deterministic (respectively, nondeterministic) finite automaton. Answer the following questions (for $M_1$ and $M_2$ given in the above figure) and give reasons for your answers.

(a) Is $< M_1, babb > \in A_{NFA}$?  Yes. $M_1$ accepts $babb$.

(b) Is $< M_1, abba > \in A_{NFA}$?  No. $M_1$ does not accept $abba$.

(c) Is $< M_2, abb > \in A_{NFA}$?  Yes. $M_2$ accepts $abb$.

(d) Is $< M_2, aabbb > \in A_{NFA}$?  Yes. $M_2$ accepts $aabbb$.

(e) Is $< M_1, M_2 > \in EQ_{DFA}$?  No. $M_2$ is not a DFA.

(f) Is $< M_1, M_1 > \in EQ_{DFA}$?  Yes. $L(M_1) = L(M_1)$.

(g) Is $< M_2, M_2 > \in EQ_{DFA}$?  No. $M_2$ is not a DFA.
2. (a) Consider the language

\[ C = \{ < M, w > \mid M \text{ is a DFA, and some string of } L(M) \text{ contains string } w \text{ as a substring} \} \]

Is the language \( C \) decidable or undecidable? Prove your answer.

**Decider for \( C \):**

\[ P = " \text{On input } < M, w > " \]

1. Construct a DFA \( A \) for \( L(M) \cap \Sigma^* w \Sigma^* \), where \( \Sigma \) is the alphabet of \( M \). This is possible because regular languages are closed under intersection.
2. Decide whether \( L(A) = \emptyset \) (DFA emptiness is decidable)
   - If yes, reject. If no accept.

(b) Consider the language

\[ D = \{ < M > \mid M \text{ is a Turing machine and there exists a DFA } A \text{ such that } L(M) = L(A) \} \]

Is the language \( D \) decidable or undecidable? Prove your answer.

\( D \) is undecidable. Two justifications (only one needed).

i) \( D \) is a non-trivial semantic property. Undecidability follows from Rice's theorem.

ii) \( D \) encodes the problem of deciding whether the language recognized by a TM is regular. This is shown in the text to be undecidable (in Theorem 5.3)
3. Give an implementation-level description of a deterministic one tape Turing machine that decides the following language $A$ over the alphabet $\Sigma = \{c,d\}$. The number of occurrences of symbol $c$ (respectively, $d$) in a string $w$ is denoted $|w|_c$ (respectively, $|w|_d$).

$$A = \{ w \in \Sigma^* \mid |w|_c \leq |w|_d \leq 2 \cdot |w|_c \}.$$ 

That is, in strings of $A$ the number of occurrences of $d$ is at least the number of occurrences of $c$ and at most 2 times the number of occurrences of $c$.

$$P = \text{On input } w \in \Sigma^*: \text{unmarked} \text{, unmarked}$$

1. Repeat the following as long as both $c$'s and $d$'s remain:
   - scan the tape and mark one $c$ and one $d$.
2. If after 1. there remain $c$'s on the tape reject.
   Else continue from 3.
3. Unmark all c's and d's.
4. Repeat the following as long as both unmarked c's and unmarked d's remain:
   - scan the tape and mark two $d$'s and one $c$.
5. If after the last scan unmarked $d$'s remain,
   reject.
   Else accept /* this includes the case where the last scan could mark only one $d$ */
4. Let
\[ \text{TWO}^\text{TM} = \{ <M> \mid M \text{ is a deterministic Turing machine and } L(M) \text{ consists of exactly two strings} \} .\]

**Without using Rice’s theorem show that** \( \text{TWO}^\text{TM} \) **is undecidable.**

We reduce \( \text{ATM} \) to \( \text{TWO}^\text{TM} \). Suppose \( \text{TM} P \) decides \( \text{TWO}^\text{TM} \). Construct decide \( Q \) for \( \text{ATM} \):

\[ Q = " \text{On input } <M, w> \]

1. Construct decide \( \text{TM} N \):

\[ N = " \text{On input } x:\]

1. If \( x = a \) or \( x = aa \) (\( a \in \Sigma \)), accept.
2. Simulate \( M \) on \( w \). If \( M \) accepts, accept.
   If \( M \) rejects, reject."

2. Run \( P \) on \( <N> \). If \( P \) accepts, reject.
   If \( P \) rejects, accept "

**Explanation why this works:**

- If \( <M, w> \notin \text{ATM} \), then \( L(N) = \{a, aa\} \) and \( <N> \notin \text{TWO}^\text{TM} \).
- If \( <M, w> \in \text{ATM} \), then \( L(N) = \Sigma^* \) and \( <N> \notin \text{TWO}^\text{TM} \).
5. Define

\[ E_{CFG} = \{ \langle G \rangle \mid G \text{ is a context-free grammar and } L(G) = \emptyset \} \].

Answer the following questions and justify your answers. ("\( \leq_m \)" denotes mapping reducibility.) If you answer "yes", please give an example of the language and the corresponding mapping reduction.

(a) Does there exist a decidable language \( B \) such that \( E_{CFG} \leq_m B \)?

Yes. We can use \( B = E_{CFG} \) and the identity function as mapping reduction.

(b) Does there exist an undecidable language \( C \) such that \( C \leq_m E_{CFG} \)?

No. Since \( E_{CFG} \) is decidable, \( C \leq_m E_{CFG} \) implies that \( C \) is decidable.

(c) Does there exist an undecidable language \( D \) such that \( E_{CFG} \leq_m D \)?

Yes. Choose \( D = A_{TM} \). Let \( M_o \) be a TM such that \( b \in L(M_o) \) and \( bb \notin L(M_o) \). Define function \( A \):

\[
A(\langle G \rangle) = \langle M_o, b \rangle \text{ if } G \text{ is a CFG and } L(G) = \emptyset \\
A(\langle G \rangle) = \langle M_o, bb \rangle \text{ if } G \text{ is a CFG and } L(G) \neq \emptyset \\
A(x) = \langle M_o, bb \rangle \text{ if } x \text{ does not encode a CFG.}
\]

\( A \) is compatible because \( E_{CFG} \) is decidable. Function \( A \) mapping reduces \( E_{CFG} \) to \( A_{TM} \).