Computability Theory
Decidable Languages

Haniel Barbosa

THE UNIVERSITY OF IOWA
Readings for this lecture

Chapter 4 of [Sipser 1996], 3rd edition. Section 4.1.
We use languages to represent various computational problems because we have a terminology for dealing with languages.

We develop examples of languages that are decidable by algorithms.

**Definition (Decidability)**

A language is *decidable* if there is an algorithm (i.e. a Turing Machine decider) to recognize it.
Acceptance in DFAs as a Membership Problem

- Consider the acceptance problem for DFAs: *test whether a particular finite automaton accepts a given string*
  This can be expressed as a language: $A_{\text{DFA}}$

- $A_{\text{DFA}}$ contains the encodings of all DFAs together with strings the DFAs accept, i.e. $A_{\text{DFA}} = \{ \langle B, w \rangle \mid B \text{ is a DFA that accepts the string } w \}$

- Therefore testing whether DFA $B$ accepts $w$ is the same as testing whether $\langle B, w \rangle \in A_{\text{DFA}}$

- Other computational problems are formulated in terms of testing membership in a language

- To show that a computational problem is decidable is to show that the encoding of the problem is decidable
Decidability of acceptance problems for DFAs

**Theorem**

$A_{DFA}$ is a decidable language.

**Proof idea: construct a TM $M$ that decides $A_{DFA}$**

$M =$“On input string $\langle B, w \rangle$, where $B$ is a DFA and $w$ is a string:

1. Simulate $B$ on $w$

2. If the simulation ends in an accept state then accept; otherwise reject.”
Performing the simulation

▷ \( \langle B, w \rangle \) is a representation of a DFA \( B \) together with a string \( w \). One can represent \( B \) by a list of its five components: \((Q, \Sigma, \delta, q_0, F)\)

▷ When \( M \) receives an input it checks first whether this input represents a DFA \( B \) and a string \( w \), otherwise reject

▷ If the input is right, \( M \) keeps track of \( B \)'s current state and \( B \)'s current position in \( w \) by writing this information on its tape

▷ Initially the state of \( B \) is \( q_0 \) and \( B \)'s current position is the leftmost symbol of \( w \); the states and position are updated according to \( \delta \)

▷ When \( M \) finishes processing the last symbol of \( w \), \( M \) accepts if \( B \) is in a final state and rejects otherwise
Theorem

$$A_{NFA} = \{ \langle B, w \rangle \mid B \text{ is an NFA that accepts the string } w \}$$ is a decidable language

Proof idea: construct a TM $N$ that decides $A_{NFA}$.

By using the TM $M$ that decides $A_{DFA}$, $N$ first converts its input NFA to a DFA by the usual technique.

$N =$“On input string $\langle B, w \rangle$, where $B$ is an NFA and $w$ is a string:

1. Convert $B$ to a DFA $C$
2. Run $M$ on $\langle C, w \rangle$
3. If $M$ accepts, accept; otherwise reject.”

Note

Running $M$ in stage 2 means incorporating $M$ into the design of $N$ as a subprocedure.
Acceptance problem for regular expressions

We can similarly determine whether a regular expression generates a given string.

**Theorem**

\[ A_{\text{REX}} = \{ \langle R, w \rangle \mid R \text{ is a regular expression that generates the string } w \} \text{ is a decidable language} \]

What would be the proof idea?
Emptiness Problem

- Another kind of problems concerning FAs in the *emptiness testing*
  - Is the language of a DFA empty?

- Consider the language

\[ E_{\text{DFA}} = \{ \langle A \rangle \mid A \text{ is a DFA and } L(A) = \emptyset \} \]
Decidability of the emptiness problem for DFAs

**Theorem**

$E_{DFA}$ is a decidable language.

**Proof idea**

- A DFA accepts some string if and only if it is possible to reach a final state from the start state by applying its state transition function.
- To test this condition we can construct a TM $T$ that marks states of a DFA in a similar manner as testing whether a graph is connected.
Decidability of the emptiness problem for DFAs

Theorem

\( E_{\text{DFA}} \) is a decidable language.

Proof idea

- A DFA accepts some string if and only if it is possible to reach a final state from the start state by applying its state transition function.
- To test this condition we can construct a TM \( T \) that marks states of a DFA in a similar manner as testing whether a graph is connected.

\( T = \) “On input string \( \langle A \rangle \), where \( A \) is a DFA:

1. Mark the start state of \( A \)
2. Repeat until no new states get marked:
   3. Mark any state that has a transition coming into it from any state that is already marked.
4. If no final state is marked, accept, otherwise reject.”
For two DFAs $A$ and $B$, is $\mathcal{L}(A) = \mathcal{L}(B)$? This class of problems amounts to testing membership in the language

$$EQ_{DFA} = \{ \langle A, B \rangle \mid A, B \text{ are DFAs, and } \mathcal{L}(A) = \mathcal{L}(B) \}$$
Language Equality

For two DFAs $A$ and $B$, is $\mathcal{L}(A) = \mathcal{L}(B)$? This class of problems amounts to testing membership in the language

$$EQ_{DFA} = \{ \langle A, B \rangle \mid A, B \text{ are DFAs, and } \mathcal{L}(A) = \mathcal{L}(B) \}$$

Definition (Symmetric Difference)

Two languages $L_1$ and $L_2$ are equal if their symmetric difference is empty, i.e.

$$\left( L_1 \cap \overline{L_2} \right) \cup \left( \overline{L_1} \cap L_2 \right) = \emptyset$$
Language Equality

For two DFAs $A$ and $B$, is $\mathcal{L}(A) = \mathcal{L}(B)$? This class of problems amounts to testing membership in the language

$$EQ_{DFA} = \{\langle A, B \rangle \mid A, B \text{ are DFAs, and } \mathcal{L}(A) = \mathcal{L}(B)\}$$

**Definition (Symmetric Difference)**

Two languages $L_1$ and $L_2$ are equal if their symmetric difference is empty, i.e.

$$\left(L_1 \cap \overline{L_2}\right) \cup \left(L_1 \cap L_2\right) = \emptyset$$

Then deciding membership in $EQ_{DFA}$ can be done in terms of the symmetric difference of the languages of the two DFAs and the emptiness problem.
Decidability of the equality problem for DFAs

**Theorem**

$EQ_{DFA}$ is a decidable language.

**Proof idea**

- Construct a DFA $C$ from $A$ and $B$ such that $C$ accepts only strings accepted either by $A$ or by $B$ but not by both.

- If $L(A) = L(B)$, then $L(C) = \emptyset$

- To test this condition we can construct a TM $F$ in terms of the TM $T$ for deciding the emptiness problem.
Decidability of the equality problem for DFAs

**Theorem**

$E_{Q_{DFA}}$ is a decidable language.

**Proof idea**

- Construct a DFA $C$ from $A$ and $B$ such that $C$ accepts only strings accepted either by $A$ or by $B$ but not by both.

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

- To test this condition we can construct a TM $F$ in terms of the TM $T$ for deciding the emptiness problem.

$F =$ “On input string $\langle A, B \rangle$, where $A$ and $B$ are DFAs:

1. Construct DFA $C$ as described
2. Run TM $T$ on $\langle C \rangle$
3. If $T$ accepts, accept, otherwise reject.”