Decidability

From here onwards we will use turing machines. But our focus is on algorithms.

Turing machine merely serves as a precise model for the definition of algorithm.

We will no longer consider low-level turing machines with states, tapes and transition functions.

When we describe a turing machine, there are three ways to do so.

(i) formal description

(ii) Implementation description
(we describe how turing machine moves its head and the way it stores data.

(iii) high-level description
we use English to describe an algorithm.

Input to a turing machine is always a string.

If we want to input an object to a turing machine, we have to encode the object as a string.

Example: Let A be the language consisting of all strings representing undirected graphs that are connected.

$$A = \{ \langle G \rangle \mid G \text{ is connected undirected graph} \}$$

we can describe a turing machine M that decides A using following high-level description.

M = "on input $\langle G \rangle$, the encoding of a graph G:

1. Select a node and mark it.
2. Repeat the following steps until no new nodes are marked:
   3. for each node in G, mark it if it is attached by an edge to an already marked node.
4. Scan all nodes of G to determine whether they are all marked.

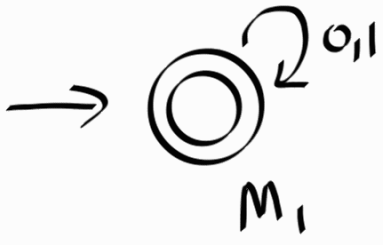
If they are all marked, accept, Otherwise, reject.

Intuitive notion of algorithms $\equiv$ Turing Machine algorithms

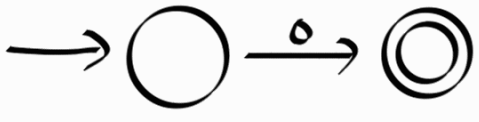
Decidable problems related to regular languages.

We choose to represent various computational problems by languages.

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



M_1



M_2

$L(M_2) = \{0\}$

$\langle M_1, \text{"01"} \rangle \in A_{DFA}$

$\langle M_1, \text{"00001"} \rangle \notin A_{DFA}$

$\langle M_2, \text{"0"} \rangle \in A_{DFA}$

$\langle M_2, \text{"001"} \rangle \notin A_{DFA}$

Decidable problems on DFAs

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

$E_{DFA} = \{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$

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

Theorem 4.1

A_{DFA} is decidable.

we construct a TM M that decides A_{DFA}.

$M =$ "on $\langle B, w\rangle$, where B is a DFA and w is a string:

1. Simulate B on w.
2. If simulation ends on an accept state, accept. If it ends on a non-accepting state, reject.

Using a similar argument we can show A_{NFA} and A_{REX} is decidable.

Theorem 4.4

E_{DFA} is decidable

We construct a turing machine T that decides E_{DFA}

T = "on <A>, where A is a DFA:

1. Mark the start state of A.
2. Repeat until no new states are marked:
   3. Mark any state that has a transition coming into it from a state that is already marked.
4. If no accept states are marked, accept. Otherwise, reject.

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

How about following TM K?

K = on "$\langle A, B\rangle$" where A, B are DFAs:

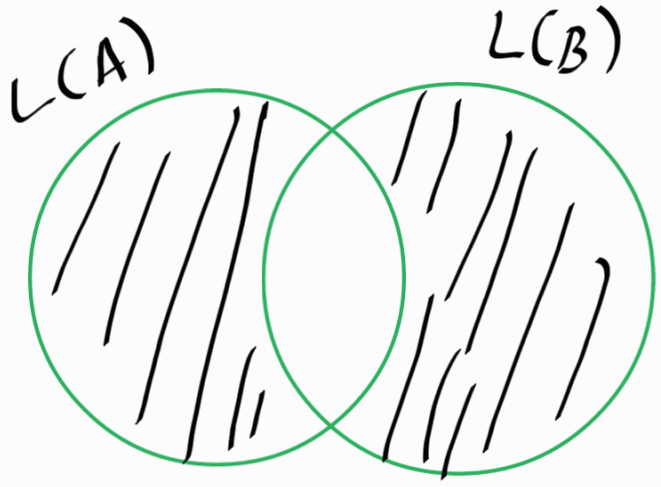
1. Feed all w into A and B.
   If one accepts and other one reject, reject.

Is this a decider?

Not a decider for EQ_{DFA}

Is this a recoghizer?

Not a recoghizer for EQ_{DFA}.



$$L(C) = \left(L(A) \cap \overline{L(B)}\right) \cup \left(\overline{L(A)} \cap L(B)\right)$$

$$L(A) = L(B) \Rightarrow L(C) = \phi$$

$$L(C) = \phi \Rightarrow L(A) = L(B)$$

$$L(A) = L(B) \Leftrightarrow L(C) = \phi$$

Regular operations are closed under $\cup, \cap, \overline{\ }$ operations.

Correct TM K is as follows

K = "on $\langle A, B \rangle$, where A, B are DFAs,

1. Construct DFA C as mentioned earlier.
2. Run TM T from Theorem 4.4 on input $\langle C \rangle$.
3. If T accepts, accept; otherwise reject.