

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 the first node and mark it.
2. Repeat the following steps until no new nodes are marked.
 3. For each node G_i , mark it if

it is attached by an edge to
a node that is already marked

4. Scan all nodes of G to determine
whether they are all marked.

If they are accept; otherwise,
reject. //

Intuitive notion
of
algorithms



Turing Machine
algorithms

Decidable problems related
to regular languages.

We choose to represent various
computational problems by languages.

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

Decidable problems on DFAs

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

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

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

Theorem 4.1

A_{DFA} is decidable.

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

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

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

Using a similar argument we can show ANFA and A_{REG} is decidable.

Theorem 4.4

E_{DFA} is decidable

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

$T = " \text{on } \langle A \rangle, \text{ where } A \text{ is a DFA:}$

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 a state that is already marked.
 4. If no accept state is marked, accept, otherwise, reject

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

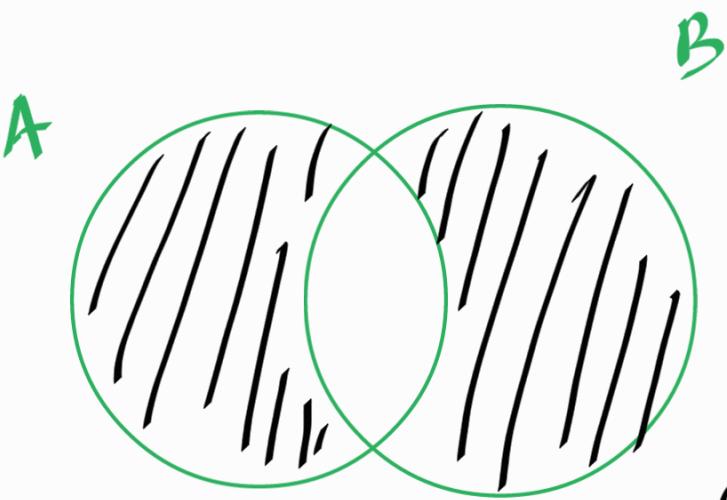
How about following TM K?

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

1. Feed all w into A and B.

If one accepts, accept; otherwise
reject

Is this a decider?



$$C = (A \cup B) - (A \cap B)$$

$$C = (A \cap \bar{B}) \cup (\bar{A} \cap B)$$

If $A=B$ if $C=\emptyset$

$$L(C) = L(A \cap \bar{B}) \cup L(\bar{A} \cap B)$$

$$L(A) = L(B) \text{ iff } L(C) = \emptyset$$

Regular languages are closed under \cup, \cap operations.

Correct TM K is as follows

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

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.

