Variants of TMs

1. TMs with stay-put
2. Multitape TMs
3. Nondeterministic TMs (NTMs)
4. TM with output (enumerators)
Enumerator

Turing Machine with Output

Sipser

Figure 3.20
Schematic of an enumerator
Enumerator

\((Q, \Sigma, \Gamma, \delta, q_{\text{start}}, q_{\text{halt}})\)

\(\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\} \times (\Sigma \times \{R\}) \cup \{(\varepsilon, S)\}\)
A language is Turing recognizable iff some enumerator enumerates it
Given an enumerator E design a TM M that recognizes the language enumerated by E

M =

“On input \( w \)
1. Run E. Every time E writes a string on its printer tape compare it with \( w \).
2. If they match, accept. Else continue running E.”

Decider or Recognizer?
Given a TM M design an enumerator E that enumerates the language recognized by M

E =
1. Run M on strings $s_1, s_2, \ldots$
2. If M accepts any $s_n$
   print $s_n$.

list $\Sigma^*$ in short lex order

$S_1, S_2, \ldots$
Given a TM $M$ design an enumerator $E$ that enumerates the language recognized by $M$

E =

“For $i = 1, 2, \ldots$
1. Run $M$ on strings $s_1, s_2, \ldots, s_i$
2. If $M$ accepts any $s_n, 1 \leq n \leq i$ print $s_n$.”
Given a TM M design an enumerator E that enumerates the language recognized by M

\[ E = \]

“For \( i = 1, 2, \ldots \)

1. Run M on strings \( s_1, s_2, \ldots, s_i \) for \( i \) steps
2. If M accepts any \( s_n \) \( 1 \leq n \leq i \) print \( s_n \).”

the same string may appear multiple times!
A = \{G: G is connected undirected graph\}

But G is not a string!!!
A = \{ \langle G \rangle : G \text{ is connected undirected graph} \}

This is called an **encoding**

Typically for graphs \( G = (V, E) \)
A = \{ \langle G \rangle : G \text{ is connected undirected graph} \}

Example:
Is
\langle G \rangle = (\{v_1, v_2, v_3, v_4\}, \{\{v_1, v_2\}, \{v_3, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}\})
in A?
A = \{ \langle G \rangle : G \text{ is connected undirected graph} \}

M =

“On input \langle G \rangle

1. Mark an arbitrary vertex
2. Repeat until no new vertices are marked
   • For each vertex
     • Mark it if it is connected by an edge to a marked vertex
3. If all vertices are marked
   then accept else reject

Decider or Recognizer?
A_{\text{DFA}} = \{ \langle D, w \rangle : D \text{ is a DFA that accepts input } w \}
4.1 Answer all parts for the following DFA $M$ and give reasons for your answers.

a. Is $\langle M, 0100 \rangle \in A_{DFA}$?
b. Is $\langle M, 011 \rangle \in A_{DFA}$?
c. Is $\langle M \rangle \in A_{DFA}$?
d. Is $\langle M, 0100 \rangle \in A_{REX}$?
e. Is $\langle M \rangle \in E_{DFA}$?
f. Is $\langle M, M \rangle \in E_{Q_{DFA}}$?
\[ A_{\text{DFA}} = \{ \langle D, w \rangle : D \text{ is a DFA that accepts input } w \} \]

\[ M_{\text{DFA}} = \]

"On input \( \langle D, w \rangle \)

1. Simulate \( D \) on \( w \)
2. If \( D \) ends in an accept state, \textbf{accept}
   else (if it ends in any other state) \textbf{reject}"

\[ L(M_{\text{DFA}}) = A_{\text{DFA}} \]

Decider or Recognizer?
\( A_{NFA} = \{ \langle N, w \rangle : N \text{ is an NFA that accepts input } w \} \)

\( M_{NFA} = \)

“On input \( \langle N, w \rangle \)
1. Convert \( N \) to an equivalent DFA \( D \)
2. Run \( M_{DFA} \) on \( \langle D, w \rangle \) and do what it does:
  accept if it accepts
  reject if it rejects”

\( L(M_{NFA}) = A_{NFA} \)

similarly for \( A_{REX} \)