Today's learning goals

- Trace the computation of a Turing machine on given input
- Describe the language recognized by a Turing machine
- Determine if a Turing machine is a decider
- Give an implementation-level description of a Turing machine
Turing machine computation

- **Read/write head** starts at leftmost position on tape
- **Input string** written on leftmost squares of tape, rest is blank
- **Computation** proceeds according to transition function:
  - Given current state of machine, and current symbol being read
  - the machine
    - transitions to new state
    - writes a symbol to its current position (overwriting existing symbol)
    - moves the tape head L or R (if possible)

- **Computation ends if and when** machine enters either the **accept** or the **reject** state.

$L(M) = \{ w | \text{ computation of } M \text{ on } w \text{ halts after entering the } \text{accept} \text{ state} \} = \{ w | w \text{ is accepted by } M \}$
Language of a TM

\[ L(M) = \{ w \mid M \text{ accepts } w \} \]

Which of the following is not always true?

A. If \( w \) is in \( L(M) \) then the computation of \( M \) on \( w \) halts and accepts.

X B. If the computation of \( M \) on \( w \) halts and rejects, then \( w \) is not in \( L(M) \).

X C. If \( w \) is not in \( L(M) \) then the computation of \( M \) on \( w \) halts and rejects.

E ?? Typ: \( M \) does not accept \( w \)

i.e. comp. of \( M \) on \( w \) does not enter \( q_{acc} \)

i.e. comp. of \( M \) on \( w \) either enters \( q_{ rej} \) or loops
At the start of the computation of $M$ on $w = w_1 \ldots w_n$

If $\delta((q_0, a)) = (q', b, R)$ then

If $\delta((q, a)) = (q', b, L)$ then
Language of a Turing machine

$L(M) = \{ w \mid \text{computation of } M \text{ on } w \text{ halts after entering the accept state}\}$

i.e. $L(M) = \{ w \mid w \text{ is accepted by } M\}$

Comparing TMs and PDAs, which of the following is true:

A. Both TMs and PDAs may accept a string before reading all of it.
B. A TM may only read symbols, whereas a PDA may write to its stack.
C. Both TMs and PDAs must read the string from left to right.
D. States in a PDA must be either accepting or rejecting, but in a TM may be neither.
E. I don't know.
\( \delta : Q \times \Pi \rightarrow Q \times \Pi \delta (+) \)

well-defined function

\( (q_{acc}, 0) \in Q \times \Pi \)
To think about

• Given a DFA, how would you simulate it with a TM?
• Given an NFA, how would you simulate it with a TM?
• Given a PDA, how would you simulate it with a TM?
Configuration

To trace DFAs: enough to list states.
To trace NFAs: tree of possible current states (incl. spontaneous moves)
To trace PDAs: tree of possible computations incl. state + stack

- Current state
- Current tape contents up to (finite) point after which all blank
- Current location of read/write head

current state is q
current tape contents are uv (and then all blanks)
current head location is first symbol of v
Special configurations

For input string w

- **Starting** configuration \( q_0 \ w \)
- Accepting configuration \( u \ q_{\text{acc}} \ v \)
- Rejecting configuration \( u \ q_{\text{rej}} \ v \)

current state is \( q \)
current tape contents are \( uv \) (and then all blanks)
current head location is first symbol of \( v \)
Language of a TM

\[ L(M) = \{ w \mid M \text{ accepts } w \} \]

= \{ w \mid \text{there is a sequence of configurations of } M \]

where \( C_1 \) is start configuration of \( M \) on input \( w \),

each \( C_i \) yields \( C_{i+1} \) and \( C_k \) is accepting configuration

"The language of \( M \)"

"The language recognized by \( M \)"
An example

$L = \{ w#w | w \text{ is in } \{0,1\}^* \}$

We already know that $L$ is

• not regular
• not context-free

We will prove that $L$ is

the language of some Turing machine
Implementation-level description

$L = \{ \text{w#w} \mid \text{w is in \{0,1\}^*} \}$

Idea for Turing machine

• Zig-zag across tape to corresponding positions on either side of '#' to check whether these positions agree. If they do not, or if there is no '#', reject. If they do, cross them off.

• Once all symbols to the left of the '#' are crossed off, check for any un-crossed-off symbols to the right of '#': if there are any, reject; if there aren't, accept.
Implementation-level description

Zig-zag across tape to corresponding positions on either side of '#' to check whether these positions agree. If they do not, or if there is no '#', reject. If they do, cross them off.

Once all symbols to the left of the '#' are crossed off, check for any un-crossed-off symbols to the right of '#': if there are any, reject; if there aren't, accept.

State diagram
Formal definition

\[ \delta(q, a, b) = (q', x, R) \]

\[ \delta(q, \#) = (q, x, R) \]

All missing transitions have output \((q_{\text{reject}}, _, R)\)

### States

- \(Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_{\text{accept}}, q_{\text{reject}}\}\)
- \(\Sigma = \{0, 1, \#\}\)
- \(\Gamma = \{0, 1, \#, x, _\}\)

Fig 3.10 in Sipser
Computation on input 0#0?
Computation on input 0#?
Describing TMs

- **Formal definition**: set of states, input alphabet, tape alphabet, transition function, state state, accept state, reject state.

- **Implementation-level definition**: English prose to describe Turing machine head movements relative to contents of tape.

- **High-level description**: Description of algorithm, without implementation details of machine. As part of this description, can "call" and run another TM as a subroutine.
For next time

GroupHW4 due Saturday, May 12

For Friday, pre-class reading:
* Bottom of page 166 and top of page 167 (high-level and implementation level definitions of Turing machines)

* Terminology for describing Turing machines pages 184-185