Week5 friday

We are ready to introduce a formal model that will capture a notion of general purpose computation.

- Similar to DFA, NFA, PDA: input will be an arbitrary string over a fixed alphabet.
- Different from NFA, PDA: machine is deterministic.
- Different from DFA, NFA, PDA: read-write head can move both to the left and to the right, and can extend to the right past the original input.
- Similar to DFA, NFA, PDA: transition function drives computation one step at a time by moving within a finite set of states, always starting at designated start state.
- Different from DFA, NFA, PDA: the special states for rejecting and accepting take effect immediately.

(See more details: Sipser p. 166)

Formally: a Turing machine is $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ where δ is the **transition function**

$$\delta: Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$$

The **computation** of M on a string w over Σ is:

- Read/write head starts at leftmost position on tape.
- Input string is written on |w|-many leftmost cells of tape, rest of the tape cells have the blank symbol. **Tape alphabet** is Γ with $\bot \in \Gamma$ and $\Sigma \subseteq \Gamma$. The blank symbol $\bot \notin \Sigma$.
- Given current state of machine and current symbol being read at the tape head, the machine transitions to next state, writes a symbol to the current position of the tape head (overwriting existing symbol), and moves the tape head L or R (if possible).
- Computation ends if and when machine enters either the accept or the reject state. This is called halting. Note: $q_{accept} \neq q_{reject}$.

The language recognized by the Turing machine M, is $L(M) = \{w \in \Sigma^* \mid w \text{ is accepted by } M\}$, which is defined as

 $\{w \in \Sigma^* \mid \text{computation of } M \text{ on } w \text{ halts after entering the accept state}\}$



The language recognized by this machine is ...

Describing Turing machines (Sipser p. 185) To define a Turing machine, we could give a

- Formal definition: the 7-tuple of parameters including set of states, input alphabet, tape alphabet, transition function, start state, accept state, and reject state; or,
- Implementation-level definition: English prose that describes the Turing machine head movements relative to contents of tape, and conditions for accepting / rejecting based on those contents.
- **High-level description**: description of algorithm (precise sequence of instructions), without implementation details of machine. As part of this description, can "call" and run another TM as a subroutine.

Fix $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, \bot\}$ for the Turing machines with the following state diagrams:



Example of string accepted: Example of string rejected:

Implementation-level description

High-level description



Example of string accepted: Example of string rejected:

Implementation-level description

 ${\bf High\text{-}level\ description}$



Example	of	string	accepted
Example	of	string	rejected:

Implementation-level description

High-level description



Example of string accepted: Example of string rejected:

 $Implementation-level\ description$

High-level description