Turing Machines

A more powerful computation model than a PDA?
Turing Machines

Some history:
- introduced by Alan Turing in 1936
- models a “human computer”
  (human writes/rewrites symbols on a sheet of paper, the human’s state of mind changes based on what s/he has seen)
- a reasonable model for real computers

Church–Turing Thesis:
For any problem L (given by a language) there exists an algorithm iff there exists a Turing machine which terminates on every input.
Example: the airline problem - given are airports and available flights, is it possible to get from a place A to a place B?

\[ A \Rightarrow F \# A \Rightarrow C \# E \Rightarrow A \# F \Rightarrow E \# D \Rightarrow B \# B \Rightarrow C \# A \Rightarrow D \# A \# B \]

\[ \Sigma = \{ A, \ldots, Z, \# \} \]

\[ L = \{ \text{strings of the above form for which there is a path from A to B} \} \]
Verbal explanation:

- The tape is infinite to the right and it initially contains the input string (the rest of the tape contains blanks - △).

- The TM starts in an initial state, reading the first symbol on the tape.

- The head can move left, right, or stay at its current position.

- The TM has two special states, h_a (accept) and h_r (reject).

- If the head moves to the left of the first symbol, this automatically means the change of state to h_r (reject).

- The transition is specified by a state and a tape symbol (to which the head points). It returns a new state, new tape symbol (to rewrite the original), and a head-move (L/R/S).
Example: As a warm-up, give a Turing machine for $a^* b^* c^*$

Simplified transition diagram:
we do not have to draw transitions leading to $h_r$. 

accepts only $a^* b^+ c^*$ (fix it)
A Turing machine (TM) is a 5-tuple \((Q, \Sigma, \Gamma, q_0, \delta)\) where:

- \(Q\) is a finite set of states not containing \(h_a, h_r\) (the two halting states)
- \(\Sigma\) is a finite alphabet (input symbols)
- \(\Gamma\) is a finite alphabet (tape symbols) such that \(\Sigma \subseteq \Gamma\) and \(\Gamma\) does not contain \(\triangle\) (the blank symbol)
- \(q_0 \in Q\) is the initial state
- \(\delta : Q \times (\Gamma \cup \{\triangle\}) \rightarrow (Q \cup \{h_a, h_r\}) \times (\Gamma \cup \{\triangle\}) \times \{L,R,S\}\) is a partial function defining the transitions.
Let $T = (Q, \Sigma, \Gamma, q_0, \delta)$ be a TM.

A configuration of $T$ is $(q, \alpha \beta)$ where $q \in Q$, $\alpha, \beta \in (\Sigma \cup \{\delta\})^*$.

Tape content until the last non-$\delta$ symbol.

The initial configuration is $(q_0, \delta x)$.

We use $\vdash_T$ to say that $T$ can get from one configuration to another configuration using a single transition. We use $\vdash_T^*$ to say that $T$ can get from one configuration to another configuration using a sequence of transitions.
Let $T = (Q, \Sigma, \Gamma, q_0, \delta)$ be a TM and $x \in \Sigma^*$. We say that $x$ is accepted by $T$ if

$$(q_0, \Delta x) \xrightarrow{\star} (h, \alpha \beta)$$

for some $\alpha, \beta \in (\Gamma \cup \{\Delta\})^*$, $f \in \Gamma \cup \{\Delta\}$.

The language accepted by $T$, denoted $L(T)$, is the set of all strings in $\Sigma^*$ that are accepted by $T$.

A string $x$ can be rejected in two ways: either the computation of $T$ on $x$ ends in the state $h_r$, or the computation of $T$ on $x$ gets into an infinite loop.

A language accepted by a TM is called recursively enumerable. A language for which there is a TM which never goes to an infinite loop is called recursive.
Example: Give a TM accepting \( \{ a^k b^k c^k \mid k \geq 0 \} \).
Let $T = (Q, \Sigma, \Gamma, q_0, \delta)$ be a TM and let $f$ be a total function from $\Sigma^*$ to $\Gamma^*$. We say that $T$ computes $f$ if for every $x \in \Sigma^*$,

$(q_0, \Delta x) \vdash_T^* (h_a, \Delta f(x))$.

**Example:** give a TM that computes the function $f(1^n) = 1^{2n}$
Possible attempts to make Turing machines stronger:

- 2-way infinite tape
- several heads, several tapes
- random access (the head can jump to any position)
- nondeterminism
- etc.

**Note:** All of the above changes can be simulated by a TM.
A stronger machine than a TM?

Example: How to simulate a 2-way infinite tape using a regular TM?

1. Simulate $M_2$, if attempts to fall overboard, then shift the tape content to the right.

2. Hold the tape.

$\Gamma'_2 = \{ \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \mid y_1, y_2 \in \Gamma_1 \cup \{ \text{a} \} \} \cup \Sigma$

$\delta'_2(q_{\text{bot } 1}, \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}) = (p_{\text{bot } 1}, \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, d)$

where $\delta'_2(q, y_2) = (p_1, y_2, d)$

1. Include transitions which at the beginning create "double-decker" symbols
2. Then do these transitions + similar (for the top of the tape, corresponding to $q_{\text{top } 1}$, plus what happens at the very beginning of the tape?)}
The definition is the same as Turing machines, except that the transition function goes from $Q \times (\Gamma \cup \{\triangle\})$ to subsets of $(Q \cup \{h_a, h_r\}) \times (\Gamma \cup \{\triangle\}) \times \{R, L, S\}$.

**Thm:** Let $T_1$ be an NTM. Then there exist a TM $T_2$ such that $L(T_1) = L(T_2)$. 

STOP when a conf. contains $h_a$