Comp487/587 - Turing Machines

1 Definition of a deterministic Turing machine

(Sections 1.2-1.4 in the book)

Formally a Deterministic Turing Machine is a tuple $M =< \Sigma, \Gamma, Q, \delta, q_0, q_A, q_R >$ where:

- Σ is the (finite) input alphabet.
- Γ is the (finite) machine's alphabet ($\Sigma \cup \{\triangleright, \not b\}$) where $\triangleright$ is a start symbol that appears only at the beginning of the tape and $\not b$ is the blank symbol.
- Q is a finite of states.
- q_0 a starting set.
- q_A a final accepting state.
- q_R a final rejecting state.
- $\delta : (Q \backslash \{q_A, q_R\}) \times \Gamma \to Q \times \Gamma \times \{R, L\}$ is the transition function. In a non final state the machine reads a symbol γ, and then replaces γ a with a (possibly) another symbol, moves the head Right or Left, and moves to a (possibly different) state.

The Turing machine is working on a tape that starts with $\triangleright$ and is infinite to the right. The machine starts working on a tape with $\triangleright$ and then the input.The $\triangleright$ is just for convenience and M is not allowed to write on $\triangleright$ or to write $\triangleright$ anywhere on the tape. The tape head is on the left most input letter.

2 Example: the problem Palindrome

Definition 1. *Given an alphabet Σ, a palindrome is a word $x \in \Sigma^*$ that can be read exactly the same from left to right and from right to left. As an example:* 0110, 1001001 *are palindromes,* 01100 *is not.*

Definition 2. *We define the problem PAL as follows:*

- *Input: a binary string x.*
- *Output:* 1 *if x is a palindrome and* 0 *otherwise.*

We now give a Turing Machine M that solves PAL. M stops at an accepting state if the given input is a palindrome, and stops at a reject state if the input is not a palindrome. For simplicity, M does not clean the tape and write 1/0 if it accepts/rejects.

Intuitively M works as follows:

- If in state q_0, M reads the blank symbol $\not b$, M accepts.
- If in state q_0, M reads a symbol $i \in 0, 1$ as the leftmost symbol, the machine remembers the i (by moving to a state $q_{found i}$), deletes the symbol and goes all the way to the left until finds $\not b$, then goes a single step to the right to the rightmost symbol.
- Then M switches to $q_{match i}$ and compares the rightmost symbol on the right. If it is not i, M rejects.
- Otherwise M deletes the last symbol, goes all the way to the left (in state q_{rev}), until reads $\not b$, goes one step left to the leftmost symbol and repeats back to state q_0.

Formally M is defined as follows. $M =< \Sigma, \Gamma, Q, \delta, q_0, q_A, q_R >$ where:

- $Q = \{q_0, q_A, q_R, q_{found0}, q_{match0}, q_{found1}, q_{match1}, q_{rev}\}$
- $\Sigma = \{0, 1\}$.
- $\Gamma = \{0, 1, b, start\}$

We define δ as follows:

- $\delta(q_0, \not b) = (q_A, 1, L)$.
- $\delta(q_0, 0) = (q_{found0}, \not b, R)$
- $\delta(q_0, 1) = (q_{found1}, \not b, R)$

- $\delta(q_{found0}, 1) = (q_{found0}, 1, R)$
- $\delta(q_{found0}, 0) = (q_{found0}, 0, R)$
- $\delta(q_{found0}, \not b) = (q_{match0}, \not b, L)$

- $\delta(q_{match0}, 1) = (q_R, \not b, L)$
- $\delta(q_{match0}, 0) = (q_{rev}, \not b, L)$
- $\delta(q_{match0}, \not b) = (q_{acc}, \not b, L)$

- $\delta(q_{rev}, 0) = (q_{rev}, 0, L)$
- $\delta(q_{rev}, 1) = (q_{rev}, 1, L)$
- $\delta(q_{rev}, \not b) = (q_0, \not b, R)$

- $\delta(q_{found1}, 1) = (q_{found1}, 1, R)$
- $\delta(q_{found1}, 0) = (q_{found1}, 0, R)$
- $\delta(q_{found1}, \not b) = (q_{match1}, \not b, L)$

- $\delta(q_{match1}, 0) = (q_R, \not b, L)$
- $\delta(q_{match1}, 1) = (q_{rev}, \not b, L)$
- $\delta(q_{match1}, \not b) = (q_A, \not b, L)$

Note: You are welcome to visit https://turingmachinesimulator.com/ where you can build, debug, and learn more about Turing machines. Note that definitions of TMs slightly deviate at different places; one however can easily reach from one simulation to another without excessive extra computation time.

3 Configuration

A *configuration* is a snapshot of the status of the TM. It includes the (finite) content of the tape, the location of the tape head, and the state the TM is at.

The *initial configuration* is where the TM is at state q_0, the leftmost cell is the start symbol. The content of the tape is then a finite number of cells (starting from the left) that contains input alphabet from Σ. the head is at the second to the left most cell, that is the first symbol of the input.

By executing the transition function, the machine moves from one configuration to another. We denote by $C \vdash C'$ the event that the configuration C' is reached from the configuration C by following the transition function δ *a single time*, i.e. if there is a single rule in δ that leads from C to C'.

Definition 3. *A computation of a TM M is a (possibly infinite) sequence of configuration $C_0, C_1, \cdots$ such that: (1) C_0 is the initial configuration; (2) for every $i \leq 0$ $C_i \vdash C_{i+1}$; (3) if the sequence is finite then the state in the last configuration is either q_A or q_R.*

Definition 4. *The language of a Turing machine M is denoted by $L(M)$ and is:*
$L(M) = \{x \in \Sigma^* | \text{the computation of } M \text{ on the input } x \text{ is finite and ends in } q_A\}$.

By defining configurations we can now formally define the running time of a TM on an input.

Definition 5. *The running time of M on input x is the number of configurations in a computation of M on input x. For an infinite computation we define the running time to be ∞.*

Definition 6. *Let $t : \mathbb{N} \to \mathbb{N} \cup \infty$ be some function. We say that M runs in time $t(n)$ if for every $n \in \mathbb{N}$ and for every input x of length n, the running time of M on x is at most $t(n)$ steps.*