Tutorial 4 - Turing machines

Gidon Rosalki

2025-04-23

1 Motivation

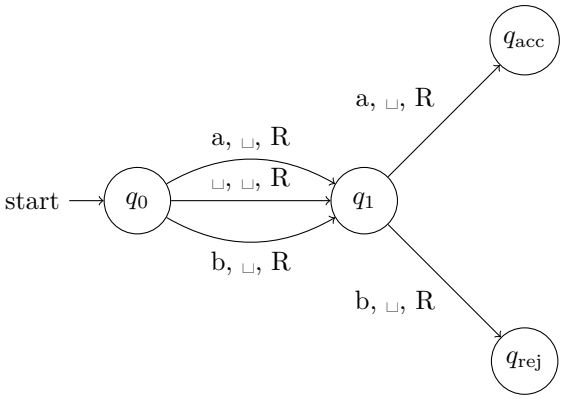
	DFA	Turing Machine
Reading the word	Once in order	Freely, any direction, as long as it likes
External memory	False	True

Table 1: TM vs DFA

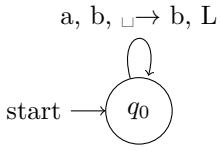
This makes TMs a more powerful computational model. The memory of a TM is a tape of cells that stretches infinitely in both directions. Each cell contains a letter. Initially the input is written on the tape, and every other cell contains a default blank symbol, denoted $\sqcup$. The machine's head initially points to the first letter of the input.

2 Examples

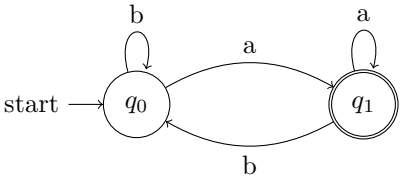
The following TM accepts if the second letter of the input is a, and deletes the first 2 letters of the input:

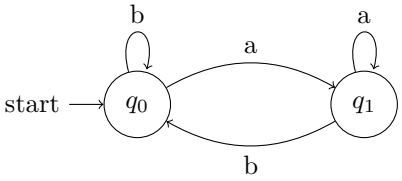


The following TM does not halt, but instead fills the tape with an endless string of bs, and overwrites the first letter of input with b.



The following DFA recognises the language $L = \{w \in \{a,b\}^* : w[-1] = a\}$:





3 Formal definition

Definition 3.1 (Turing maching)**.** *A Turing machine is comprised as follows:*

$$M = (\Sigma, \Gamma, \sqcup, Q, q_0, F, \delta)$$

Where

- Σ - *Finite set of input alphabet*
- Γ - *Finite set, working alphabet,* $\Sigma \subseteq \Gamma$
- $\sqcup$- *Space:* $\sqcup \in \Gamma \setminus \Sigma$
- W - *States (finite, non empty)*
- q_0 - *Starting state*
- F - *Final states*
- δ - *Transition function such that* $\delta : (Q \setminus F) \times \Gamma \to Q \times \Gamma \times \{R, L\}$

Definition 3.2 (Configuration)**.** *A* ***configuration*** *is made up of the current state, the state of the tape, and the position of the head. If the tape is in state* $x_1 \dots x_n$*, the machine is in state* q*, and the head is over* x_i*, then we represent the configuration as follows:*

$$x_1 \dots x_{i-1} q x_i x_{i+1} \dots x_n$$

Definition 3.3 (Run)**.** *A* ***run*** *of a TM* M *on* $w \in \Sigma^*$ *is a finite or infinite sequence of configurations* $c_1, \dots, c_T$ *(where when it is infinite,* $T = \infty$*), such that*

1. $c_1 = q_0 w_1 \dots w_n$*, where* $w = w_1 \dots w_n$
2. *For all* $0 \leq j < T$*, if*

   $$c_j = x_1 \dots x_{i-1} q x_i \dots x_n$$

   then the next configuration c_{j+1} *is the subsequent configuration, according to* δ*, which is to say:*

   - *If* $\delta(q, x_i) = (q', y, L)$ *then*

     $$c_{j+1} = x_1 \dots x_{i-2} q' x_{i-1} y_i x_{i+1} \dots x_n$$

   - *If* $\delta(q, x_i) = (q', y, R)$ *then*

     $$c_{j+1} = x_1 \dots x_{i-2} x_{i-1} y_i q' x_{i+1} \dots x_n$$

Definition 3.4 (Decidable TM)**.** *A Turing machine for a decision problem is a machine with final states*

$$F = \{q_{acc}, q_{rej}\}$$

where reaching q_{acc} *means the machine accepts, and reaching* q_{rej} *means the machine rejects.*

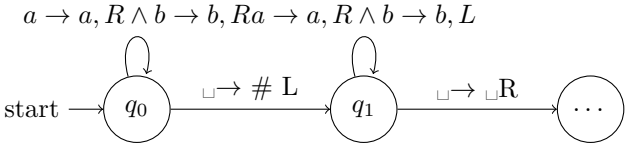
Definition 3.5 (Language)**.** *Given a deciding TM* M*,* $L(M)$ *is*

$$L(M) = \{w \in \Sigma^* : M accepts w\}$$

4 Useful procedures

4.1 Mark with hash

Suppose we want to mark the end of the input with a special symbol #. This can be useful for example when a TM uses the tape cells to the right of the input to store auxiliary data in its computation, and wants to separate the data from the input. The following TM (with the alphabet $\Sigma = \{a, b\}$) puts # at the end, and then returns to the beginning of the input:

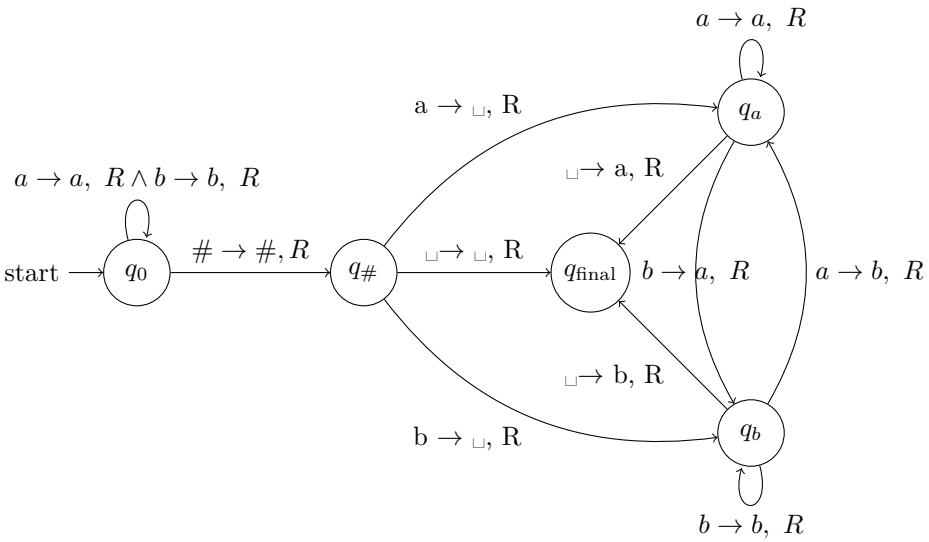


4.2 Shift right

This machine takes $x\#y$, and replaces it with $x\#\sqcup y$. It operates as follows:

1. Move the head to the beginning of y (ie, right after the # symbol.
2. Replace the first symbol of y with a blank $\sqcup$, move the head one cell to the right, and enter a state that remembers the symbol that was overwritten.
3. In each step:
   (a) Write the symbol remembered from the previous step
4. When a blank symbol is encountered (indicating the end of the word), terminate in a final state.

Let us explicitly construct the TM for $\Sigma = \{a, b\}$:



5 Designing Turing Machines

5.1 $a^n b^n$

Consider the language $L = \{a^n b^n : n \geq 0\}$. We shall describe the machine as follows:

1. We will put # at the beginning, and the end, of the input
2. We will go to the first letter that is not a space to the right of the leftmost #: If it is #, we accept, b we reject, and a we delete (replace with $\sqcup$) and continue.
3. Go to the first non blank symbol before the right #: If it is #, we reject, a we reject, and b we delete and go back to step 2.

5.2 Substring

Describe a Turing machine that decides whether a short string is a substring of a longer string. Formally, suppose the input is of the form $x\#y$; the machine should accept if x is a substring of y, and reject otherwise. We will use a technique where we "write" x above y, thus allowing us to compare the two in parallel. We will define the tape alphabet to include pairs of symbols stacked vertically:

$$\Gamma = \Sigma \cup (\Sigma \cup \{\sqcup\}) \times (\Sigma \cup \{\sqcup\})$$

The TM operates as follows:

1. For each symbol τ in y, we will replace it with $\overset{\sqcup}{\tau}$
2. Return to the beginning of x
3. For every σ in x, we will delete it, enter a state that remembers it, move right until reaching some cell $\overset{\sqcup}{\tau}$ (reject on $\sqcup$), replace it with $\overset{\sigma}{\tau}$, return to the beginning of what remains of x
4. Move right until reaching the first cell of the form $\overset{\sqcup}{\tau}$ (ie, the ending of x written above y)
5. Compare x and y symbol by symbol as follows:

(a) If the current cell is $\overset{\sigma}{\sigma}$, continue left

(b) If the current cell is $\overset{\sigma}{\tau}$ where $\sigma \neq \tau$, move to a special mismatch state, and continue left

6. If you reach $\overset{\sqcup}{\tau}$, while still in the match state, accept
7. Otherwise (i.e., if a mismatch occurred), shift the string x one cell to the right using the algorithm from the previous example. Be sure to keep the lower symbols (those of y) unchanged
8. Return to step 5, and repeat the comparison at the new position
9. If, when shifting x to the right, you reach $\sqcup$, reject.

The runtime is $O(|x|\,|y|)$, since we need to perform $O(|y|)$ comparison attempts, each taking $O(|x|)$. Additionally, the shifting takes $O(|x|)$, so we get an overall result of $O(|x|\,|y|)$.