Theory of Computation Notes: Turing Machines

Arjun Chandrasekhar

1 Turing Machines

Here we will describe a new model of computation for recognizing languages and solving decision problems. Informally, a Turing Machine contains the following components:

1. An infinitely long tape split into infinitely many squares
2. A tape head that can move around the tape one square at a time. The tape head can read and write on the tape. It also can toggle between different states.

The machine performs computation as follows:

1. All of the input string w is placed onto the tape. The tape head starts on the first character.
2. At every step, the machine read the character on the current square.
3. Based on what character it reads, and what state it is in, the machine write a new character onto the tape square. The machine moves either left or right, and updates its state.
4. The machine continues reading, writing, and moving until it moves into an accept state or a reject state.

We now give a formal definition of a Turing Machine.

Definition 1.1. A **Turing Machine (TM)** is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R)$ where Q, Σ, Γ are finite sets and:

1. Q is the set of states.
2. Σ is the **input alphabet**. This alphabet should not contain the blank symbol $\sqcup$
3. Γ is the **tape alphabet**, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the **transition function.**
5. $q_s \in Q$ is the **start state.**
6. $q_A \in Q$ is the **accept state**.
7. $q_R \in Q$ is the **reject state**.

To define how a TM performs computation, we define the concept of a **configuration**. Informally, the configuration describes all of the tape contents, the state of the tape head, and the location of the tape head.

Definition 1.2. Formally, let $q \in Q$ be a state, and let $u, v \in \Gamma^*$ be strings from the tape alphabet. We write uqv for the configuration where the current state is q, the tape currently contains uv, and the tape head is located on the first character of v. The tape contains all blank symbols to the right of v. For example, the configuration $1011q_701111$ means the machine is in state q_7, the tape contains 101101111, and the tape head is on top of the second 0.

We say configuration C_1 **yields** C_2 if it is possible to go from C_1 to C_2 in exactly one step. Formally, let $a, b, c \in \Gamma$ be tape symbols, $u, v \in \Gamma^*$ be strings from the tape alphabet, and $q_i, q_j \in Q$ be states. Suppose $\delta(q_i, b) = (q_j, c, L)$. This means that if the machine is in state q_i and it reads b, it should write c, move left, and transition to state q_j. In this case we say the configuration uaq_ibv yields uq_jacv.

Similarly, suppose $\delta(q_i, b) = (q_j, c, R)$. This means that if the machine is in state q_i and it reads b, it should write c, move right, and transition to state q_j. In this case, we say the configuration uaq_ibv yields $uacq_jv$.

The **start** configuration is q_sw; this simply indicates that the tape contains the input string, and the head starts on the left-most symbol. Any configuration that contains q_A is an **accepting configuration**, and any configuration that contains q_R is a **rejecting configuration**. A **halting configuration** is any configuration that is either accepting or rejecting.

A TM **accepts** input w if a sequence of configurations $C_1, C_2, \ldots C_n$ existr where

1. C_1 is the start configuration of M on input w
2. C_i yields C_{i+1} for each i
3. C_n is an accepting configuration.

Definition 1.3. We say the **language of M**, denoted $L(M)$ is the set of strings that are accepted by M.

Remark. Some notes:

1. Unlike the automata models we have seen, a Turing machine does not read its input one by one. The input starts on the tape. The machine can go back and forth freely, and it can read characters more than once
2. The machine does not stop when it reads the last character; it stops when it enters the accept state. ***This means the machine is not necessarily guaranteed to stop running after a certain point.***

Definition 1.4. We say a machine M **recognizes** a language L if M accepts all strings $w \in L$. If $w \notin L$, then M does not accept w; this means M may either reject w or go into an infinite loop. Basically, M accepts w if and only if $w \in L$. Alternately, $L(M) = L$ if M recognizes L.

We say L is **Turing-recognizable** if some Turing machine recognizes it.

Definition 1.5. We say a machine M **decides** a language L if M accepts all strings $w \in L$ *and* M rejects all strings $w \notin L$. M is guaranteed to halt on all strings.

We say L is **Turing-decidable** or simply **decidable** if some some Turing machine decides it.

Remark. If L is decidable then L is recognizable; the machine M that decides L also recognizes L, because M accepts all strings $w \in L$ and does not accept (in fact, rejects) all strings $w \notin L$.

Example 1.1. We will show that the language $\Sigma^*1\Sigma^*$ is decidable. Here, we will give a formal description as well.

Intuitively, the machine scans the input and accepts as soon as it encounters a 1. If it reaches the end of the input without finding a 1, it rejects. The machine knows it has reached the end of the input once it reads a blank space.

1. $Q = \{q_0, q_A, q_R\}$
2. $\Sigma = \{0, 1\}$
3. $\Gamma = \{0, 1, \sqcup\}$
4. $\delta(q_0, 0) = (q_0, 0, R)$
   $\delta(q_0, 1) = (q_A, 1, R)$
   $\delta(q_0, \sqcup) = (q_R, \sqcup, L)$
5. $q_s = q_0$
6. $q_A = q_A$
7. $q_R = q_R$

Example 1.2. Here we describe a machine to decide $L = \{a^n b^n\}$. Rather than giving a formal description, we give a high level description.

1. First, scan the string from left to right to make sure the characters are in the right order. When the machine encounters its first b, it goes into a new state to remember this. Afterwards, if it encounters any a's, it automatically rejects.
2. If the characters are in the right order, it goes back to the start. It reads an a and replaces it with a' it. It then scans for the next b.
   (a) If it finds a b, it replaces it with b'. It then goes back to the start, and repeats the process.
   (b) If it never finds a b, reject
3. If we never find an a, make sure there are not any remaining b's. If there are, reject. Otherwise, accept.

1.1 Exercises

1. Show that the language of palindromes $L = \{w \in \{a, b\}^* | w = w_R\}$ is decidable. Try giving a formal description.

2. Show that the language $L = \{a^n b^n c^n | n \geq 0\}$ is Turing-decidable. Do not give a formal description, but give a clear tape-level description.