24 Uses of Turing Machines

24.1 Introduction

We have previously covered the application of Turing Machine as a *recognizer* and *decider*. In this lecture we will discuss the uses of Turing Machine as an **enumerator** and a **function computer**. We will also give an introduction to **Universal Turing machines**.

24.2 Turing machine as an Enumerator

Simply put, an *Enumerator* is a Turing machine with an output printer. The Turing machine can use the printer as an output device to print strings. The following diagram gives a schematic of an enumerator :-

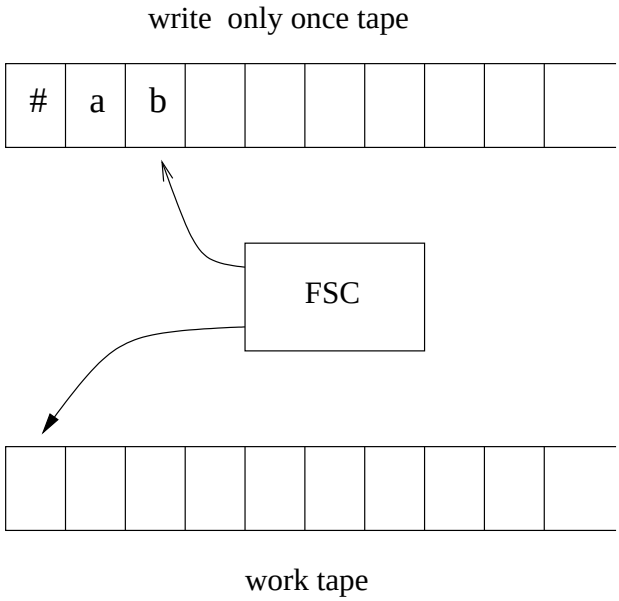

Figure 1: Schematic of an Enumerator

The language enumerated by an enumerator *E* is the collection of strings that it eventually prints out. *E* may generate the strings in any order, possibly with repetitions. We must also note that there is no input to an enumerator. Also, if the enumerating TM does not halt, it may print an infinite list of strings. A formal definition of an enumerator follows

24.2.1 Definition

An **enumerator** *E* is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{print})$ where

1. Q is the finite set of states;

2. Σ is the **print tape alphabet**, *not* containing the blank symbol $\sqcup$;
3. Γ is the **work tape alphabet**, where $\{\sqcup\} \in \Gamma$ and $\Sigma \subseteq \Gamma$;
4. $\delta : Q \times \Sigma \times \Gamma \mapsto Q \times \Sigma \times \{L, R\} \times \Gamma \times \{L, R\}$ is the transition function for the two tape device ;
5. $q_0 \in Q$ is the initial state ;
6. $q_{print} \in Q$ is the final state.

24.2.2 Example

If $L = \{a^n b^n : n \geq 0\}$ then the enumerator will output the following strings :-

$$\{\varepsilon, ab, a^2b^2, a^3b^3, ...\}$$

Theorem 1 *A language is Turing-recognizable if and only if some enumerator enumerates it.*

Proof of (IF): To show that if an enumerator *E* enumerates a language *L*, there is a Turing Machine *M* which recognizes *L*.

Given an input string x, we design M in the following way :

1. Run *E* to enumerate the next string y of L. Compare it with x
2. If $x = y$ accept x, else goto 1

Proof of (ONLY IF): To show that if a Turing machine M recognizes a language L, there is an enumerator for L.

Let $s_1, s_2, s_3, ...$ be a list of all possible strings in Σ^*. One can readily construct an enumerator E_s which generates such a sequence. An enumerator E for $L(M)$ works as follows:

Repeat the following for $i = 1, 2, 3, 4.....$

1. Run M for i steps on each input $s_1, s_2, s_3, ...s_i$;
2. If any computation is accepted, print out the corresponding s_j.

If M accepts a particular string s it eventually appears on the list generated by E_s, after which it is printed out.

24.3 Turing Machine as a function computer

Let us first take a look at what a partial function is -

24.3.1 Definition

A function f :A$\mapsto$B is called a partial function if $\forall\, a \in$ A, there is at most one element $f(a) \in$ B. Note that it is possible that a certain element of A may not be mapped to any element in B. An example of a partial function has been represented below

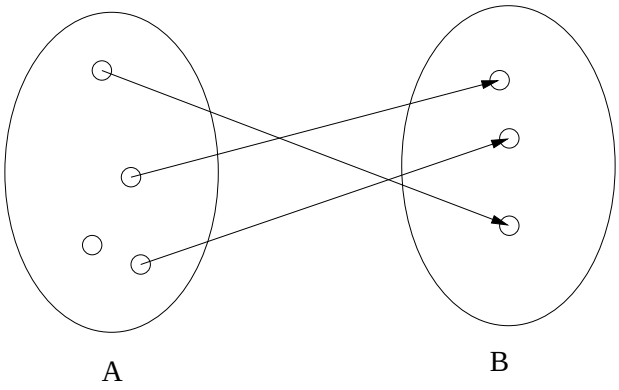

Figure 2: An example of a partial function

A partial function $f : (\Sigma^*)^n \longrightarrow \Sigma^*$ is said to be *Turing computable* if there exists a TM $M = (Q, \Sigma, \Gamma, \delta, q_0, \#, h)$, where $\# \in \Gamma$, $h =$ halt state; such that

$$q_0 x_1 \# x_2 \# x_3 .....x_n \longrightarrow yhz$$

where $yz, x_1, x_2, ...x_n \in \Sigma^*$ and $f(x_1, x_2, ...x_n) = yz$

Below is an example with the addition function $f : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$

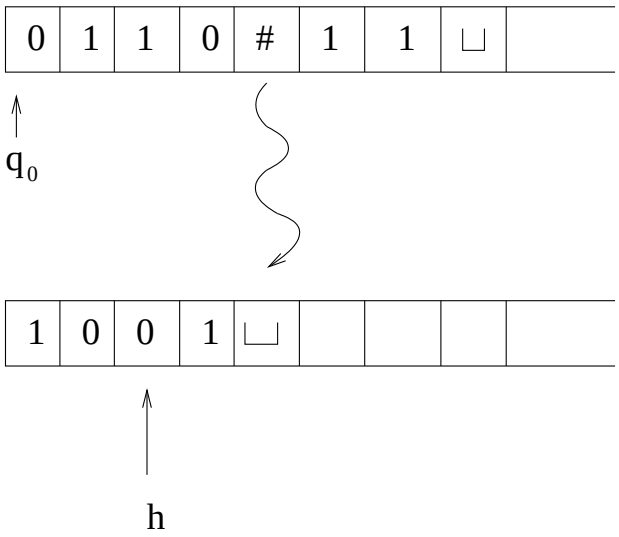

Figure 3: Addition function

Hence, a function $f : \mathbb{N}^n \rightarrow \mathbb{N}$ is computed by a TM if $F : (\Sigma^*)^n \rightarrow \Sigma^*$ is also computed by a TM where

$$F(\overline{n}_1, \overline{n}_2, ...\overline{n}_k) = \overline{f(n_1, n_2, n_3, ...n_k)}$$

Here $\overline{n}_i$ is the encoding of n_i in Σ. We now move on to *Universal Turing Machines*.

24.4 Universal Turing Machines

A **Universal Turing Machine** is one which can accept the description of another Turing machine as input and simulate the operation of that Turing machine. To construct a universal TM we need to devise an *encoding scheme* to serve all TM's. Let us first discuss a few notations-
Since we are going the encode the description of a TM, we need to standardize our notation of the state transition function δ. Note that δ can be represented in more than one ways as shown below-

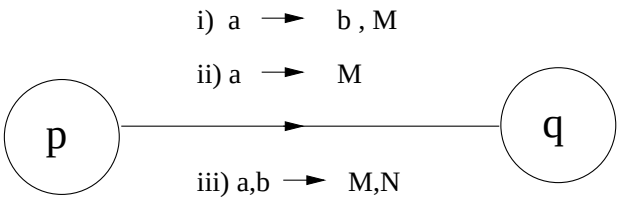

Figure 4: Different notations for δ

We propose the following representations for the above -

1. δ(p,a) = (q,b,M)
2. δ(p,a) = (q,a,M)
3. δ(p,a) = (q,a,M), δ(p,b) = (q,b,N)

Let us consider the following language -

$$L = \{x \in \{0,1\}^* : x = y0\}$$

The state transition diagram of a TM M which accepts L can be represented as below -

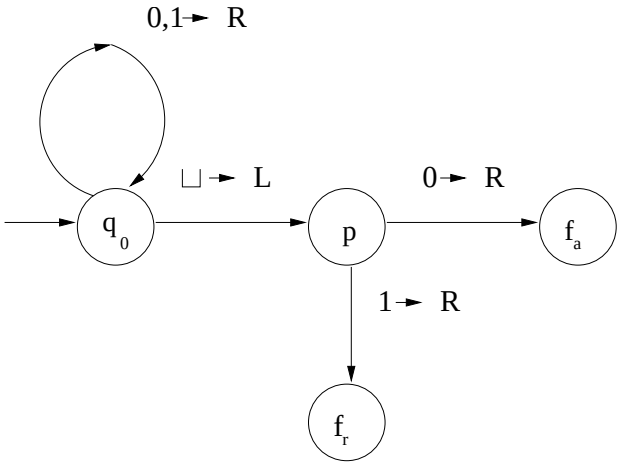

Figure 5: State transition diagram for M

We now give the formal description of M :

$$M = (\{q_0, p, f_a, f_r\}, \{0,1\}, \{0,1,\sqcup\}, \delta, q_0, f_a, f_r)$$

We define an encoding alphabet $\Delta = \{a, b\}$ and propose the following encoding scheme:

$$q_0 \rightarrow a$$

$$p \rightarrow aa$$

$$f_a \rightarrow aaa$$

$$f_r \rightarrow aaaa$$

$$0 \rightarrow a^5, 1 \rightarrow a^6, \sqcup \rightarrow a^7$$

$$L \rightarrow a^8, R \rightarrow a^9$$

To distinguish between two symbols of the same field we use the encoding string b and to distinguish between two different fields we use bb. Hence the state transition functions are encoded as

$$\delta(q_0, 0) \mapsto (p, 0, R) \longrightarrow aba^5baaba^5ba^9bb$$

Now the description of M can be encoded as :

$$(\{q_0, p, f_a, f_r\}, \{0, 1\}, \{0, 1, \sqcup\}, \delta, q_0, f_a, f_r) \rightarrow \underbrace{abaabaaabaaaabb}_{Q}\underbrace{a^5ba^6bb}_{\Sigma}\underbrace{a^5ba^6ba^7bb}_{\Gamma}.....$$

This is followed by the encoded strings of each of the state transition functions and finally q_0, f_a, f_r. A universal TM U expects a description of any TM M followed by an input string ω which is the encoded form of an input x to M. U operates on the entire input string in the same way as M operates on x ,*ie*, U accepts (or rejects) the input string only if M accepts (or rejects) x.