Automata Theory - Lecture 9

Models of Computation – Turing Machines (TMs)

Lecturer: Martha Gichuki

Lecture learning outcomes

At the end of the lecture you will be able to:

(i) Describe Turing Machines.

(ii) Describe Turing Machines using State Diagrams

(iii) Formally describe various Turing machines

Extensions of Finite Automata

The family of languages accepted by the Finite Automata (FA) is called the family of regular languages. More powerful automata can accept more complicated languages. Such automata include: -

(i) Pushdown Automata (PDA)

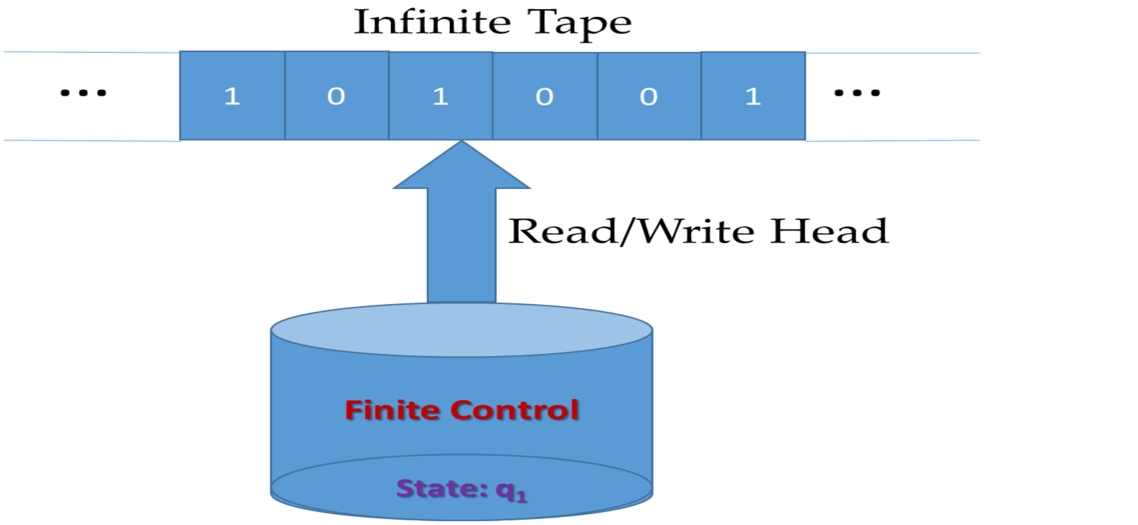
Such machines are identical to Deterministic Finite Automata (DFAs) and Non-Deterministic Finite Automata (NFAs), except that they additionally carry memory in form of a stack. Their transition function will now also depend on the symbol(s) on top of the stack, and will specify how the stack is to be changed at each transition. Non-deterministic PDAs accept the context-free languages.

(ii) Turing Machines (TMs)

These are the most powerful computational machines. They possess an infinite memory in form of a tape and a read/write head which can read and change the tape. The read/write head moves in either direction along the tape.

Which language does a Turing Machine (TMs) recognize/decide? Turing machines are equivalent to algorithms and are the theoretical basis for modern computers. Turing machines decide recurve languages and recognize the recursively enumerable languages.

Turing Machines have the essential feature of unrestricted access to limited memory, distinguishing them from weaker modes like Finite Automata (FA) and Push Down Stack machines (PDA).



Source: Introduction to the theory of computation (3rd ed.), Michael, S. Boston, Cengage Learning. ISBN-13: 978-1133187790, (2012). Page 166, 169

FORMAL DEFINITION OF A TURING MACHINE

A Turing Machine is ***a 7-tuple* ($Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}$),** where

(i) Q is a finite set of states

(ii) Σ is the input alphabet – blank symbol (∪) not included

(iii) Γ is the tape alphabet

(iv) δ: Q X $\Gamma \rightarrow$ Q X Γ X {L, R} is the transition function

(v) $q_0 \in$ Q is the start state

(vi) $q_{accept, \in}$ Q is the Accept state, and

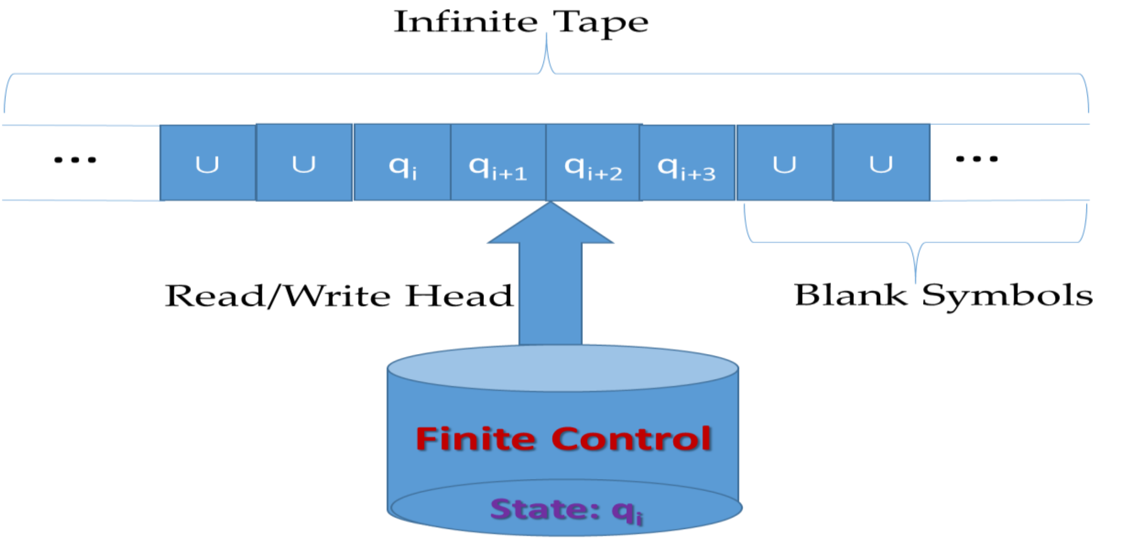
(vii) $q_{reject} \in$ Q is the Reject state.

The Transition Function

- δ: Q X $\Gamma \rightarrow$ Q X Γ X {L, R} is the transition function whereby while the machine is in State Q, reads a symbol from the tape alphabet, it moves to another state Q, replaces the symbol with another one and the Read/Write head moves either to the Left (L) or to the Right (R)

- Even if the Input alphabet is exhausted, the tape has infinite blank symbols along the tape

Turing Machine Configuration



Source: Introduction to the theory of computation (3rd ed.), Michael, S. Boston, Cengage Learning. ISBN-13: 978-1133187790, (2012). Page 166, 169

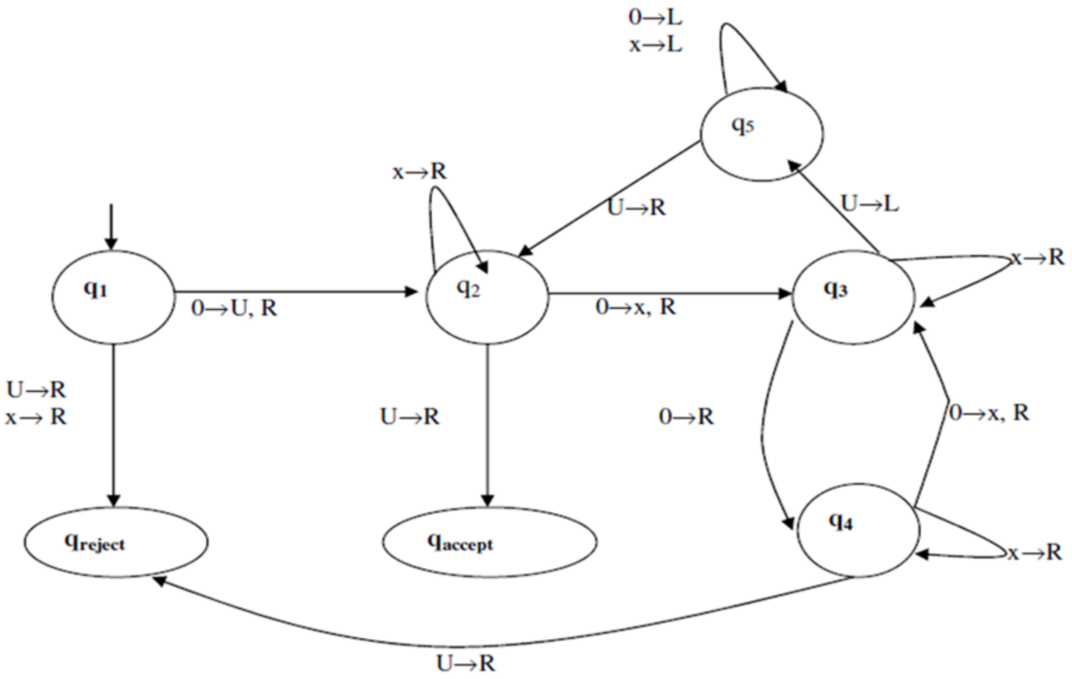
<u>Example 1:</u>

Given a Turing machine M_6, which recognizes the language consisting of all strings of 0s whose length is a power of 2? It decides the language A = $\{0^{2n}$ | n>=0}. E.g. 2^0, 2^1, 2^2, 2^3 …$2^{n.}$ {0, 00, 0000, and 00000000 …}

The formal definition of $M_6 = (Q, \sum, \Gamma, \delta, q_1, q_{accept}, q_{reject})$ ***is given as follows: -***

(i) Q = {q1, q2, q3, q4, q5, qaccept, qreject};
(ii) $\sum = \{0\}$;
(iii) $\Gamma = \{0, x, U\}$;
(iv) δ the transition function
(v) q_1 = Start State;
(vi) q_{accept} = Accept State and

(vii) q_{reject} = Reject State. The machine is described using the state diagram below: -



Turing Machine M1

Source: Introduction to the theory of computation (3rd ed.), Michael, S. Boston, Cengage Learning. ISBN-13: 978-1133187790, (2012). Page 172.

a) Show that the language 00 is accepted by machine M_6

$q_1$00⊔⊔⊔…
⊔$q_2$0⊔⊔⊔…
⊔ xq_3⊔⊔⊔…
⊔ q_5x⊔⊔⊔…
⊔ q_5⊔x⊔⊔⊔…
⊔ q_2x⊔⊔⊔…
⊔ x q_2⊔⊔⊔…
⊔ x⊔ q_{accept}

a) Show that the language 000 is not accepted by machine M_6

$q_1$000⊔⊔⊔…
⊔$q_2$00⊔⊔⊔…
⊔ x$q_3$0⊔⊔⊔…
⊔x0q_4⊔⊔⊔…
⊔x0⊔ q_{reject} ⊔⊔…

b) Show that the language 0000 is accepted by machine M_6

$q_1$0000⊔⊔⊔…
⊔$q_2$000⊔⊔⊔…
⊔ x$q_3$00⊔⊔⊔…
⊔ x0$q_4$0⊔⊔⊔…
⊔x0x q_3⊔⊔⊔…

⊔x0 q_5 x ⊔⊔⊔…
⊔x $q_5$0x ⊔⊔⊔…
⊔ q_5 x0x ⊔⊔⊔…
q_5 ⊔x0x ⊔⊔⊔…
⊔ q_2 x0x ⊔⊔⊔…
⊔ x q_2 0x ⊔⊔⊔…
⊔ xx q_3 x ⊔⊔⊔…
⊔ xx x q_3 ⊔⊔⊔…
⊔ xx q_5x ⊔⊔⊔…
⊔ x q_5 xx ⊔⊔⊔…
⊔ q_5 xx x ⊔⊔⊔…
q_5 ⊔ xx x ⊔⊔⊔…
⊔ q2 xx x ⊔⊔⊔…
⊔ xq2 x x ⊔⊔⊔…
⊔ xx q2 x ⊔⊔⊔…
⊔xx xq2 ⊔⊔⊔…
⊔ xx x ⊔q_{accept}

Assignment:

Show whether the following strings are accepted or rejected

i). 0

$uq_{1}0uu...$

$uuq_{2}uu...$

$uuuq_{accept}$

ii). 00000

$uq_{1}00000uu...$

$uuq_{2}0000uu...$

$uuxq_{3}000uu...$

$uux0q_{4}00uu...$

$uuxoxq_{3}0uu...$

$uux0x0q_{4}uu...$

$uux0x0uq_{reject}$

iii). 000000

$uq_{1}000000uu...$

$uuq_{2}00000uu...$

$uuxq_{3}0000uu...$

$uux0q_{4}000uu...$

$uuxoxq_{3}00uu...$

$uux0x0q_{4}0uu...$

$uux0x0xq_{3}uu...$

$uux0x0q_{5}xuu...$

$uux0xq_{5}0xuu...$

$uux0q_{5}x0xuu...$

$uuxq_{5}0x0xuu...$

$uuq_{5}x0x0xuu...$

$uq_{5}ux0x0xuu...$

$uuq_{2}x0x0xuu...$

$uuxq_{2}0x0xuu...$

$uuxxq_{3}x0xuu...$

$uuxxxq_{3}0xuu...$

$uuxxx0q_{4}xuu...$

$uuxxx0xq_{4}uu...$

$uuxxx0xuq_{reject}$

Variants of Turing Machines: -

(i) Multi-tape Turing Machine

- Has several tapes, each with its own head for reading and writing.
- As the name suggests, a multi-tape Turing machine has several tapes (k) for reading and writing information, instead of only one.
- The first tape is initialized with the input, and the rest are initially empty.
- The transition function is defined as:

$$\delta : Q\times\Gamma^k\rightarrow Q\times\Gamma^k\times\{L,R\}^k$$

- We can transition based on all k current input symbols, write a symbol to each of the k tapes, and move each of the k tapes either left or right. However, multi-tape Turing machines are no more capable than regular ones.

(ii) Enumerator

- This is a Turing machine with a printer attached to it.
- At the beginning of the tape, there is no input and instead of having accept or reject strings, the enumerator prints out a set of strings and this becomes the language of the enumerator.
- This string on the tape can be printed any time as decided by the enumerator

(iii) Non-Deterministic Turing Machine

- At any point, the machine may proceed according to several possibilities
- At each step in the computation, the machine can take one of several actions
- The transition function is defined as:

$$\delta: Q\times\Gamma\rightarrow P\ (Q\times\Gamma\times \{L, R\})$$

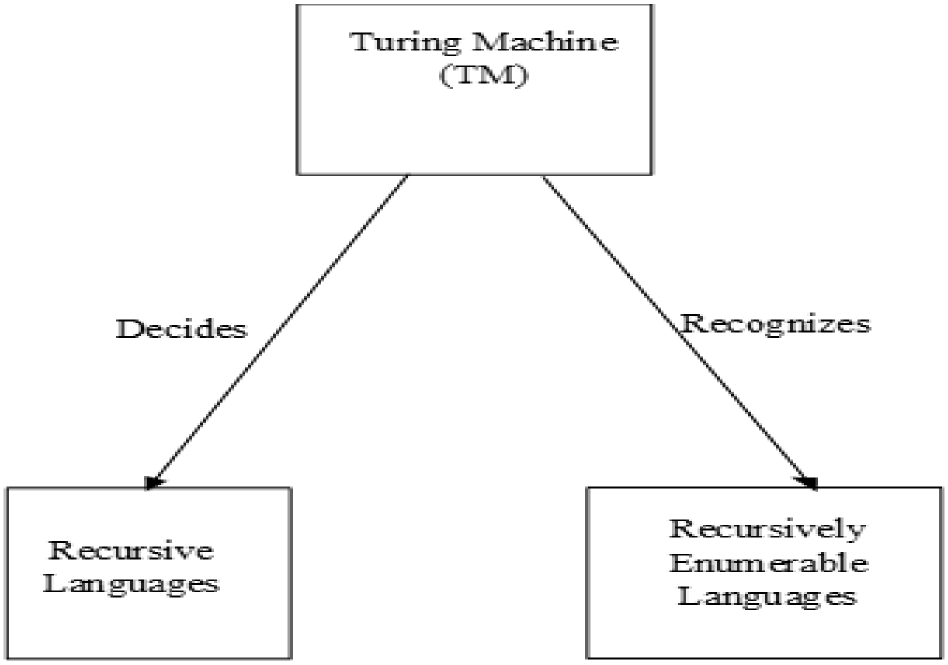
- Similar to the NFA, the computation of a non-deterministic TM is a tree, where each branch is a possible action.
- If any branch of the computation results in an
- accept state, the machine accepts the input.
- However, Non-deterministic Turing machines are no more powerful than deterministic ones.

Equivalence with Other Models

- Researchers have proposed several models of computation with time with all of them exhibiting the characteristic of unrestricted access to unlimited memory.
- Some of these models are similar to the Turing machines, while others are different e.g. the lambda calculus by Alonzo Church.
- This points at the fact that computational power depends on the computer model in use.
- Whatever can be handled by a powerful super-computer can also be handled by a basic Turing machine.

Summary

- ✓ At the heart of every machine definition is the transition Function
- ✓ Finite Automata can either be DFA or NFA, the same applies to Turing machines and PDAs
- ✓ There are other extensions of Finite Automata including the Push Down Automata (PDA) and the Turing Machine (TM) among others.
- ✓ Finite Automata recognize Regular Languages
- ✓ Push Down Automata recognize Context Free Languages
- ✓ Turing machines decide recurve languages and recognize the recursively enumerable languages.



References

1 Rowan G. & John T., (2009), *Discrete Mathematics: Proofs, Structures and Applications*, CRC Press, ISBN: 9781439812808.

2 W. D. Wallis (2003), *A Beginners Guide to Discrete Mathematics*, Springer Science & Business Media, ISBN: 978-0817642693.

3 Introduction to the theory of computation (3rd ed.), Michael, S. Boston, Cengage Learning. ISBN-13: 978-1133187790, (2012).

4 Introduction to languages and the theory of computation (3rd ed.), Martin, J., New York: McGraw-Hill. ISBN-13: 978-0072322002, (2002)