ITS64304 Theory of Computation

School of Computer Science
Taylor’s University Lakeside Campus

Lecture 4: Turing Machines

Dr Raja..

Learning outcomes

At the end of this topic students should be able to:

- Identify a formal language for a given Turing machine,
- Construct a Turing machine for a given specification*

* Course Learning Outcome 2

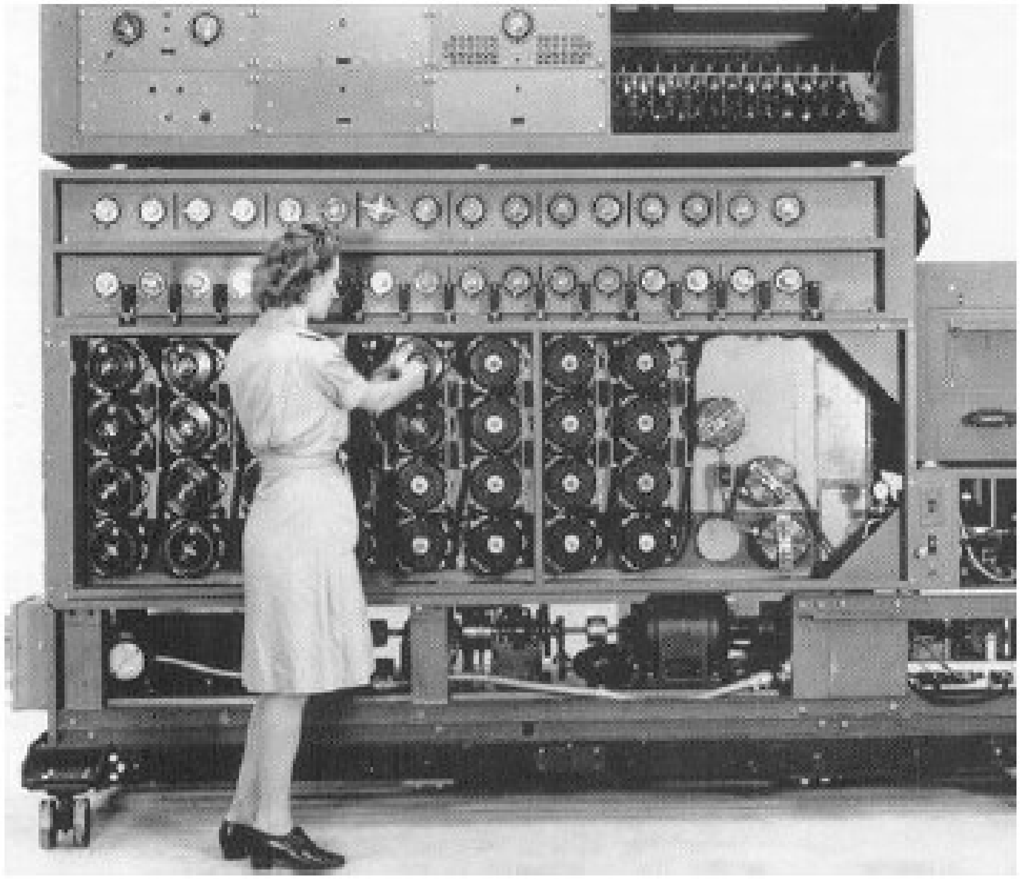
Turing Machines

- Invented by Alan Turing in 1936
- Formal model of computation
- Model of stored-program computer
- Similar concept to a modern computer
- Encode data and programs as sequence of symbols
- Unbounded memory using an infinite tape
- Consists: finite control, a tape and a head
  - An instruction (transition) table and current state
  - An infinite tape where symbols can be written
  - Head – reading *or writing on tape*

- Turing Machine (TM) = FSA + Tape
- TM = PDA + 2 stacks

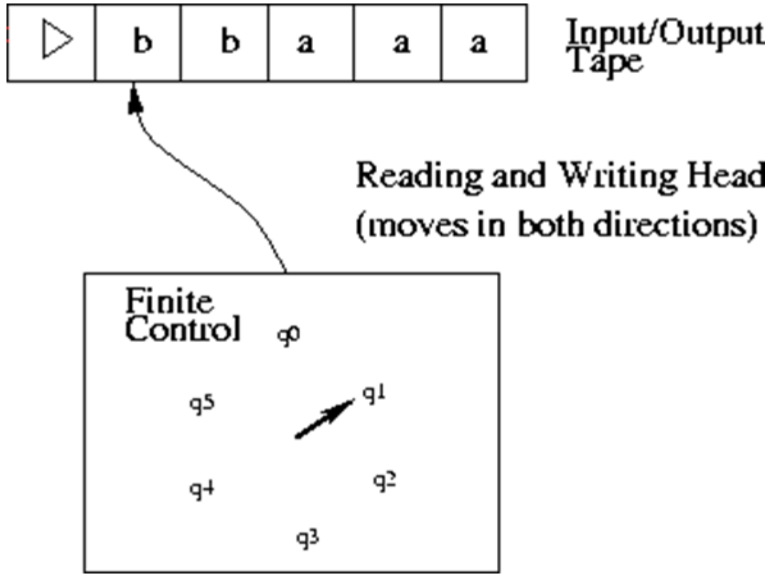
A real Turing Machine

Bombe
1940'
(Used to decrypt Enigma encrypted messages during WWII)


Source: http://en.wikipedia.org/wiki/Bombe

Turing Machines

- (same) Input as a string on a tape
- (same) Internal register for distinct internal states
- (same) Instructions for changing to a new state, depending on the input
- (same) A distinguished initial state
- (same) A number of final states
- (NEW) Output as a string on the tape



Turing Machines

- The computation sequence is
  - Change to a new state (specified by the current state and input symbol)
  - Write a new symbol on the tape
  - Move the tape head 1 place left or right

    - read → write → move → change state

- Tape is bounded to the left (by ▷), but is infinite to the right.
- Initially tape head is pointing at ▷
- Input w is a string in Σ^*, BwBBB...

Turing Machines

Definition 9: A Turing machine is a quintuple $M = (Q, \Sigma, \Gamma, \delta, q_0)$ where

- Q is a finite set of states
- Γ is the tape alphabet, which must include the symbol *B* for a blank space
- Σ is the input alphabet, which is a subset of $\Gamma - \{B\}$
- q_0 is the initial state
- δ , the transition function is a function from $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$.
  - $\delta(q_i, x) = (q_j, y, d), d \in \{L,R\}$

Transition

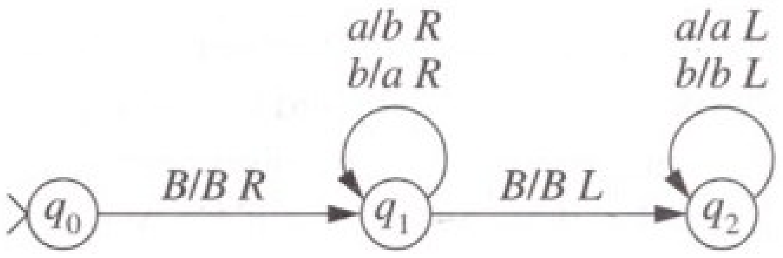
- Transition depends on
  - Current state
  - Current symbol under tape head
- Transition is 3 actions
  - Change state
  - Write symbol on tape
  - Move the tape head
- Computation **halts**
  - if it encounters a state and symbol for which no transition is defined
- Computation **terminates abnormally**
  - when a transition specifies a move to the left of the boundary of the tape
- **TM here is deterministic** as there is a maximum of one transition specified for every combination of the current state and input symbol

Example: TM to interchange *a*’s and *b*’s

- Q = {q_0, q_1, q_2}, S = {a, b}, Γ = {a, b, B}
- The tabular representation of the transition function of a **standard Turing machine** with input alphabet *{a, b}* is given in the table below.

δ	B	a	b
q_0	q_1, B, R		
q_1	q_2, B, L	q_1, b, R	q_1, a, R
q_2		q_2, a, L	q_2, b, L

- Transition from state q_0 moves the tape head to position one to read the input.
- Transitions in state q_1 reads the input string and interchange the symbols *a* and *b.*
- Transitions in q_2 return the machine to the initial position.



Turing Machines as Language Acceptors

- TMs can be designed to accept languages and to compute functions.
- TM as language acceptors
  - A computation accepts or rejects the input string
- Acceptance is defined by the final state of the computation.
  - similar to the technique used by FSA and PDA to accept strings.
  - Unlike FSA and PDA, a Turing machine need not read the entire input string to accept the string.

Turing Machines as Language Acceptors

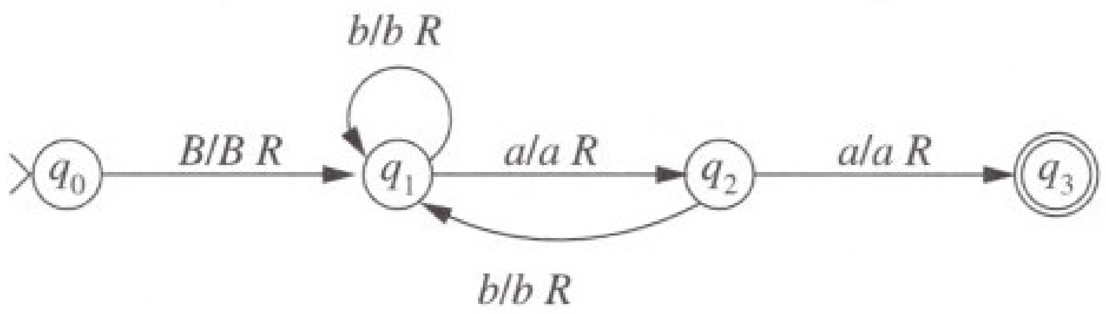
- TM augmented with final states is a sextuple
  $(Q, \Sigma, \Gamma, \delta, q_0, F)$, $F \subseteq Q$, F is a set of final states

- A string $w \in \Sigma^*$ is accepted by final state if the computation of M with input w halts in a final state.
  i.e. $L(T) = \{w \mid w \in \Sigma^*$ and w is accepted by T$\}$

- Computation terminates abnormally
  - TM rejects the input regardless of the state in which the machine halts.

Example

- Machine for (a ∪ b)*aa(a ∪ b)* assuming ∑ = {a, b}



- Computation for input *aabb*
  - $q_0BaabbB \to Bq_1aabbB \to Baq_2abbB \to Baaq_3bbB$

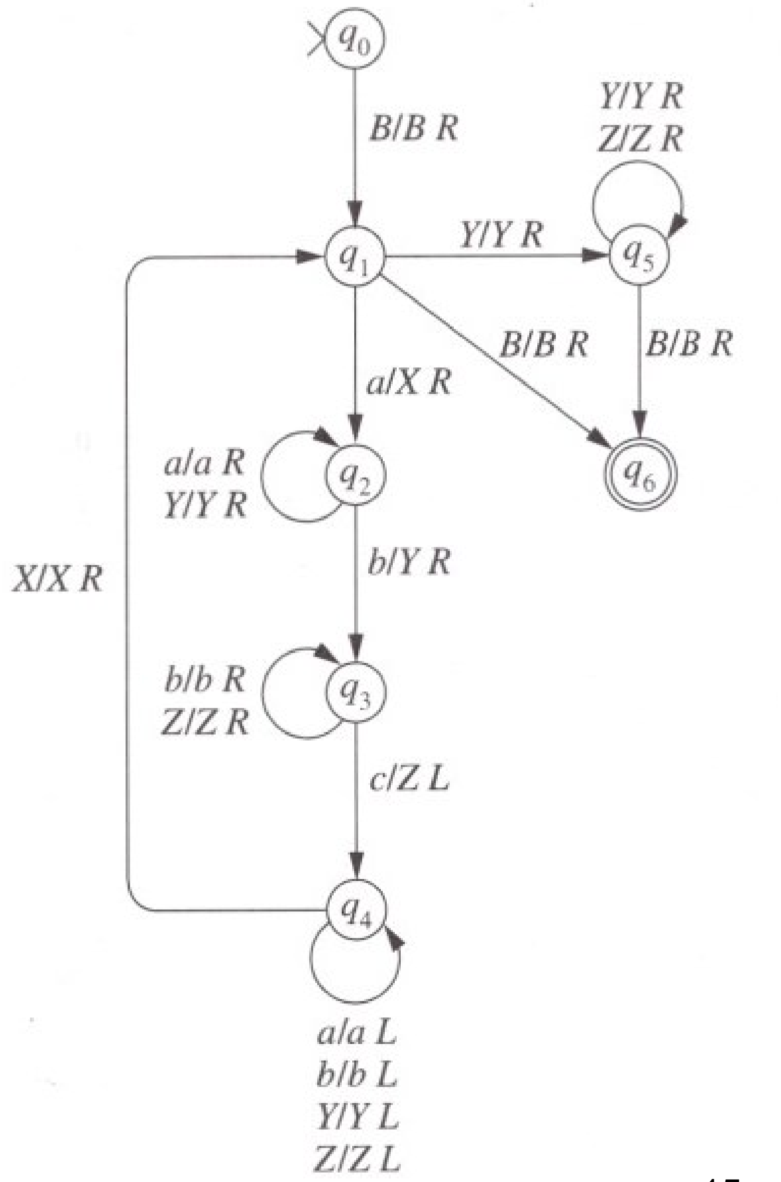
- Successful computation terminates when a substring aa is encountered
- All other computations halt on reading 1st blank following the input

Turing Machines as Language Acceptors

- A language accepted by TM is called recursively enumerable language
- Three possible outcome for TM computation:
  - halt and accept input string
  - halt and reject
  - may not halt at all
- A language accepted by TM that halts for all input is recursive.
- Hence the language given in previous slide is recursive

Example

- L = {$a^i b^i c^i$ | i >= 0}
- Given TM accepts aabbcc
  - Halts in q6
- Does not match
  - bca – halts in q1
  - abb – halts in q3



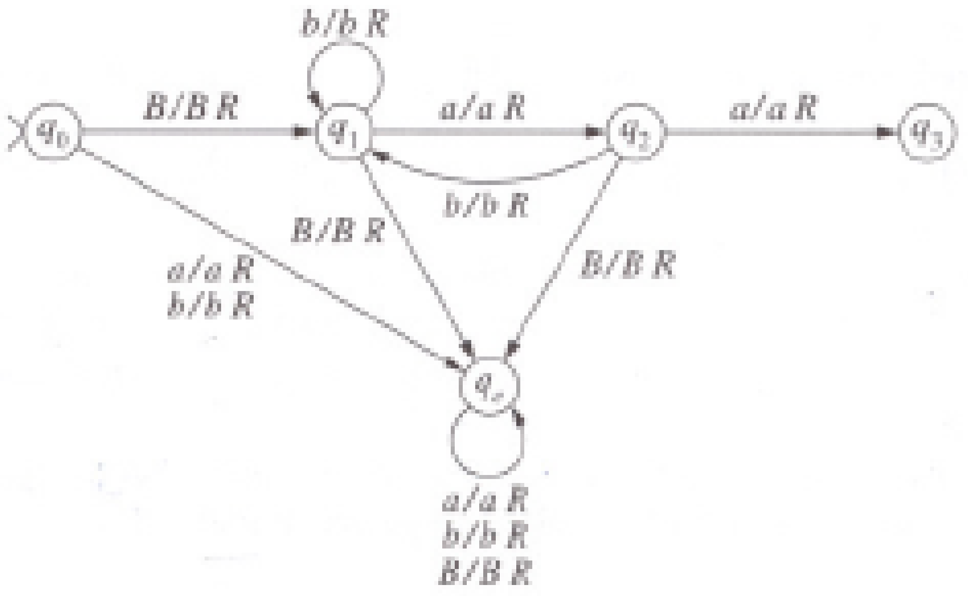
Acceptance by halting

- In TM designed to accept by halting
  - input string is accepted if the computation initiated with the string halts.
  - computation terminates abnormally – rejects.

- Let M = (Q, ∑, Γ , δ, q_0) be a TM.
- A string $w \in \sum^*$ is accepted by halting if the computation of M with input w halts (normally).

Acceptance by halting - Example

- Machine for (a u b)*aa(a u b)* by halting.



- A computation enters q_e when the entire input has been read and no *aa* found.

Conclusion

- The language L is accepted by a TM that accepts by final state
- The language L is accepted by a TM that accepts by halting
- Every language L accepted by a TM by halting is accepted by a TM by final state
- meaning that machines designed in this manner accept the same family of languages as those accepted by standard TM

Read your lesson materials

Go thru’ Tutorials on TM, prepare for it…..