CSE211-Formal Languages and Automata Theory

U3L2 – Introduction to Turing Machines

Dr. P. Saravanan
School of Computing
SASTRA Deemed University

Agenda

- Recap of CSG and LBA
- Computational models
- Why Turing Machines?
- What is Turing Machine?
- How does Turing machine work?

Natures of computational model

- *Predicate calculus* --- declarative
- *Partial-recursive functions* --- computational (a programming-language-like notion)
- *Turing machine* --- computational (computer-like)

(invented by Alan Turing several years before true computers were invented)

Equivalence of *maximal* computational models

- *All maximal computational models*
  - *compute the same functions*

    (or)
  - *recognize the same languages,*

having the same power of computation.

Why Turing Machines?

- The study of decidability provides guidance to programmers about what they might or might not be able to accomplish through programming
- Previous problems are dealt with programs
- But *not* all problems can be solved by programs
- We need a simple model to deal with other decision problems (like grammar ambiguity problems)
- The *Turing machine* is one of such models, whose configuration is easy to describe, but whose function is the most versatile

Why Turing Machines?

- Why not deal with C programs or something like that?
- Answer: You can, but it is easier to prove things about TM's, because they are so simple
  - And yet they are as powerful as any computer.
    - More so, in fact, since they have infinite memory.

Then Why Not Finite-State Machines to Model Computers?

- In principle, you could, but it is not instructive.
- Programming models don't build in a limit on memory.
- In practice, you can go to Fry's and buy another disk.
- But finite automata vital at the chip level (model-checking).

Hypothesis

All computations done by a modern computer can be done by a Turing machine.

- (a hypothesis which is not proved but believed so far!)

Turing-Machine Formal def.

- A Turing machine (TM) is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ where
  1. Q: A finite set of *states*
  2. Σ: An *input alphabet*
  3. Γ: A *tape alphabet* (with Σ being a *subset* of it).
  4. δ: A *transition function,* $\delta(q, X) = (p, Y, D)$
  5. q_0 : A *start state*
  6. B: A *blank symbol* (B, in $\Gamma - \Sigma$, typically).
     1. All tape except for the input is blank initially.
  7. F: A set of *final states* ($F \subseteq Q$, typically).

Conventions

- a, b, … are input symbols.
- …, X, Y, Z are tape symbols.
- …, w, x, y, z are strings of input symbols.
- α, β,… are strings of tape symbols.

The Transition Function

- δ: a transition function $\delta(q, X) = (p, Y, D)$ where
- Takes two arguments:
  1. A state q, in Q.
  2. A tape symbol X in Γ.
- δ(q, Z) is either undefined or a triple of the form (p, Y, D).
  - p is a state.
  - Y is the new tape symbol.
  - D is a *direction*, L or R.

Actions of the Turing Machines

- If $\delta(q, X) = (p, Y, D)$ then, in state q, scanning Z under its tape head, the TM:
  1. Changes the state to p.
  2. Replaces X by Y on the tape.
  3. Moves the head one square in direction D.
     - D = L: move left; D = R; move right.

Example: Turing Machine

- This TM scans its input right, looking for a 1.
- If it finds one, it changes it to a 0, goes to final state f, and halts.
- If it reaches a blank, it changes it to a 1 and moves left.

Example: Turing Machine – (2)

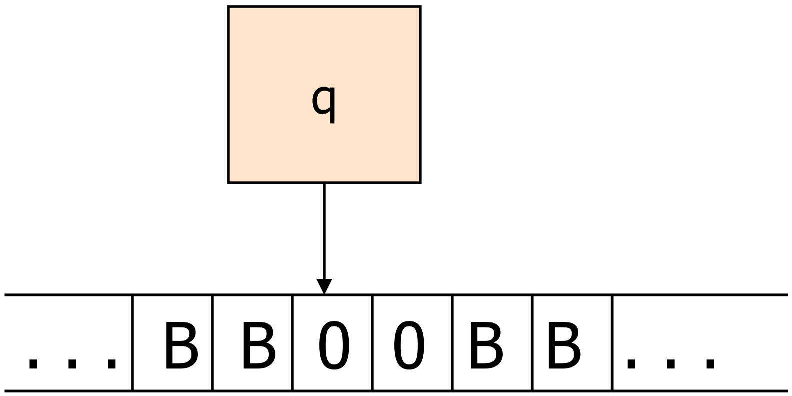
- States = {q (start), f (final)}.
- Input symbols = {0, 1}.
- Tape symbols = {0, 1, B}.
- $\delta(q, 0) = (q, 0, R)$.
- $\delta(q, 1) = (f, 0, R)$.
- $\delta(q, B) = (q, 1, L)$.

Simulation of TM

$\delta(q, 0) = (q, 0, R)$

$\delta(q, 1) = (f, 0, R)$

$\delta(q, B) = (q, 1, L)$

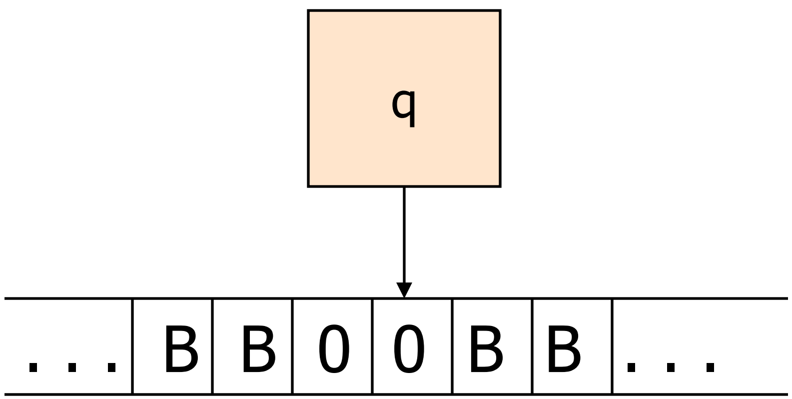


Simulation of TM

$\delta(q, 0) = (q, 0, R)$

$\delta(q, 1) = (f, 0, R)$

$\delta(q, B) = (q, 1, L)$

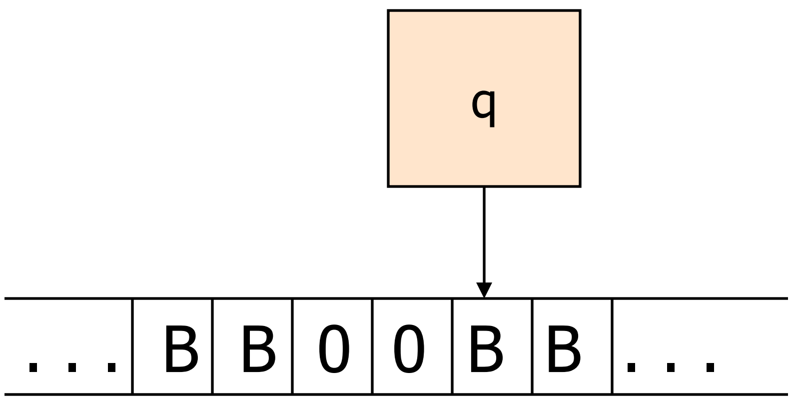


Simulation of TM

$\delta(q, 0) = (q, 0, R)$

$\delta(q, 1) = (f, 0, R)$

$\delta(q, B) = (q, 1, L)$

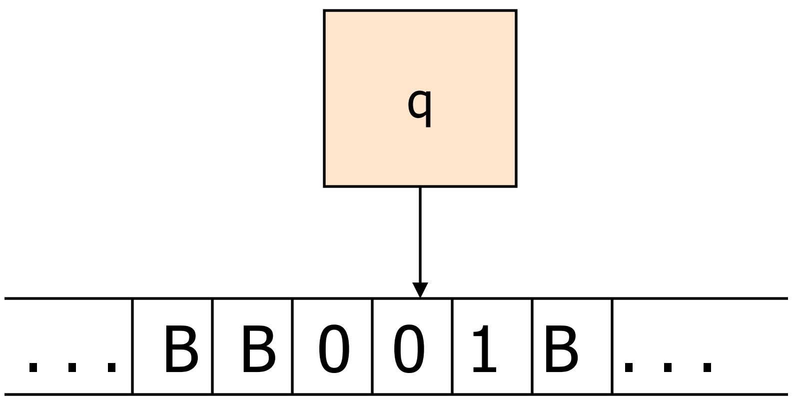


Simulation of TM

$\delta(q, 0) = (q, 0, R)$

$\delta(q, 1) = (f, 0, R)$

$\delta(q, B) = (q, 1, L)$

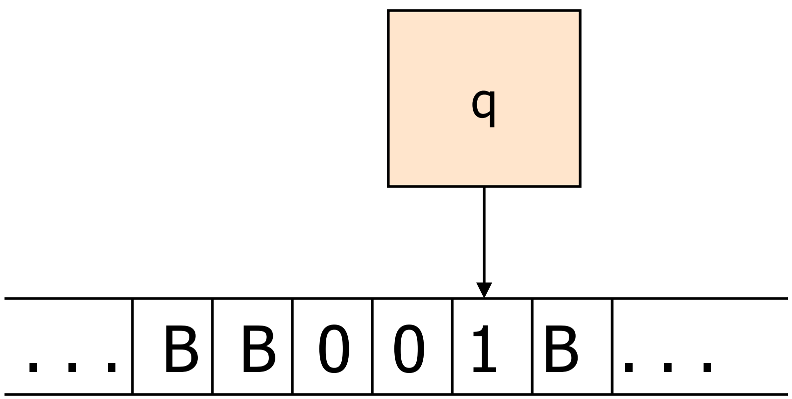


Simulation of TM

$\delta(q, 0) = (q, 0, R)$

$\delta(q, 1) = (f, 0, R)$

$\delta(q, B) = (q, 1, L)$

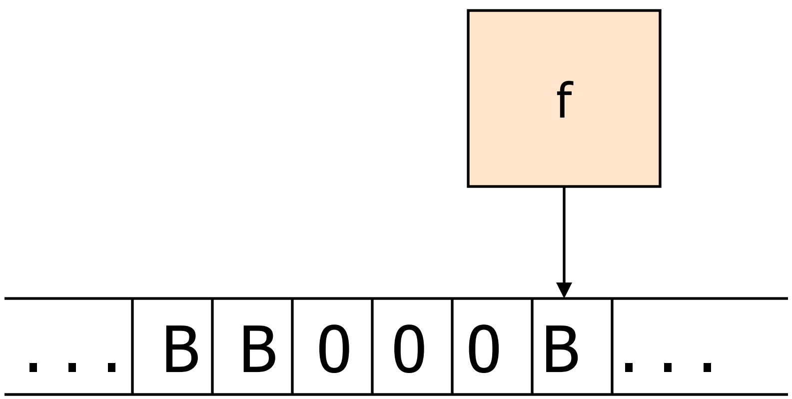


Simulation of TM

$\delta(q, 0) = (q, 0, R)$

$\delta(q, 1) = (f, 0, R)$

$\delta(q, B) = (q, 1, L)$



No move is possible.
The TM halts and accepts.

Summary

- Computational models
- Why Turing Machines?
- What is Turing Machine?
- How does Turing machine work?

References

- John E. Hopcroft, Rajeev Motwani and Jeffrey D. Ullman, *Introduction to Automata Theory*, Languages, and Computation, Pearson, 3rd Edition, 2011.
- Peter Linz, An Introduction to Formal Languages and Automata, Jones and Bartle Learning International, United Kingdom, 6th Edition, 2016.

Next Class: **Unit III**

Instantaneous Descriptions of Turing Machines

Thank you.