Turing Machines

CptS 317 ACT II:

Computability Theory

Complexity Theory

Computability Theory

Automata Theory

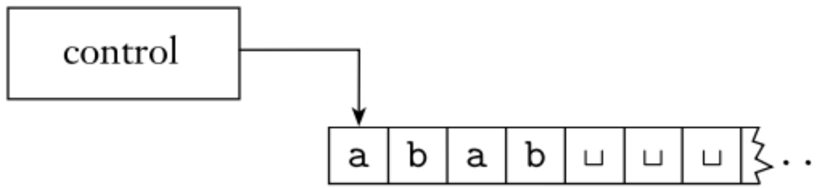
Models of computing devices we have seen so far

- Finite Automata
  - Good for devices with small amount of memory
- Push Down Automata
  - Good for devices with unlimited memory usable in LIFO (stack) manner
- Too restricted to serve as models of general-purpose computers

Turing Machine

- Much more powerful model
- First proposed by Alan Turing in 1936
- Similar to FA, but with an unlimited and unrestricted memory
- Can do everything that a real computer can do
- Yet, even a TM cannot solve certain problems
  - These problems are beyond the theoretical limits of computation

Turing Machine -- schematic



- TM uses an infinite tape as its unlimited memory
- Has a tape head that can read and write symbols and move around the tape
- Initially, the tape contains only the input string and is blank everywhere else
- Stores information by writing on the tape
- To read information, the machine can move its head back over it
- Continues computing until it decides to produce an output
- Outputs accept and reject by entering designated states
- If it doesn't enter an accepting or rejecting state, it goes forever

Differences between FA and TM

- A TM can both write on the tape and read from it
- The read-write head can move both to the left and to the right
- The tape is infinite
- The special states for rejecting and accepting take effect immediately

Informal description of a TM

- Example: consider designing a TM M_1 for testing membership in the language

$B = \{w\#w \mid w \in \{0,1\}^*\}$

- Want M_1 to accept if the input is a member of B and to reject otherwise

Informal description of a TM

- Strategy: zig-zag to the corresponding places on the two sides of the # symbol and determine whether they match
- Place marks on the top to keep track of which places correspond
- We design M_1 to work in this way
  - Makes multiple passes over the input string
  - On each pass it matches one of the characters on each side of the # symbol
  - To keep track of checked symbols, M_1 crosses off each symbol as it is examined
  - If it crosses off all symbols, that means everything matched successfully, and M_1 goes to accept state
  - If it discovers mismatch, it enters reject state

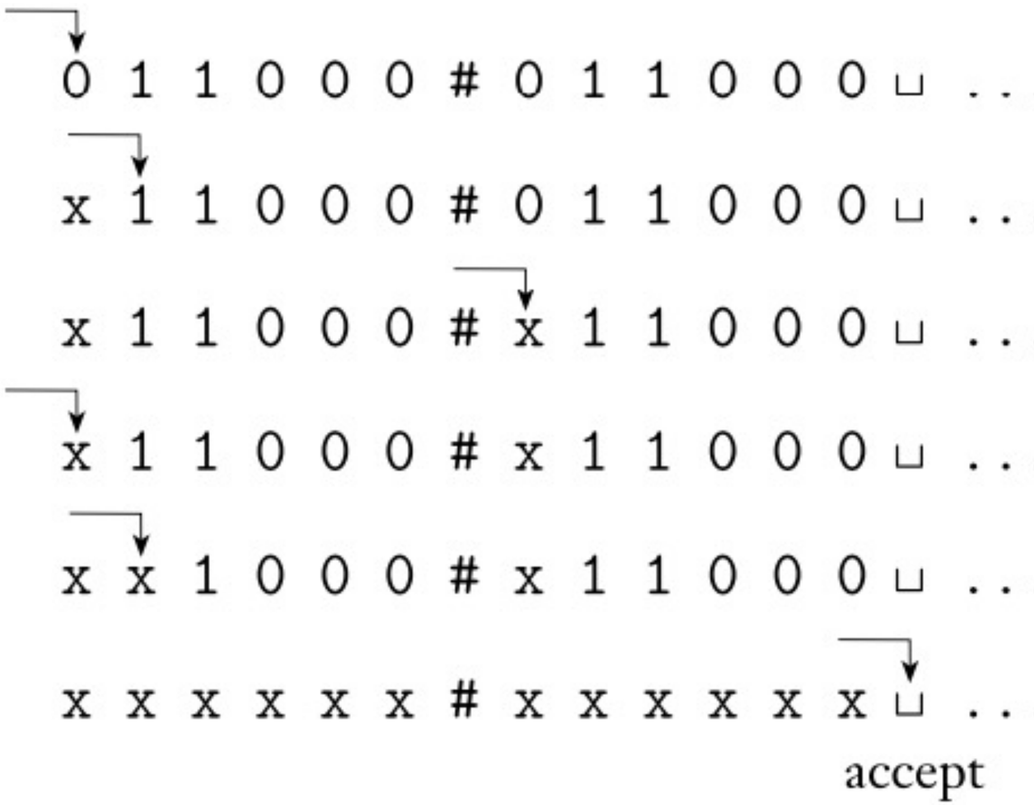
In summary M_1's algorithm..

M_1 = "*on input string w:*

1. *Zig Zag across the tape to corresponding positions on either side of # to check whether the inner positions contain the same symbol. If they don't, or if no # is found, reject. Cross off symbols as they are checked to keep track of which symbols correspond.*

2. *When all symbols to the left of # have been crossed off, check for any remaining symbols on the right of #. If any symbols remain, reject; otherwise accept."*

Snapshot of M_1 computing on input 011000#011000



Formal definition of TM

- Transition function δ

$$Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$$

That is, when the machine is in a certain state q and the head is over a tape square containing a symbol a, and if

$$\delta(q, a) = (r, b, L),$$

the machine writes the symbol b replacing a, and goes to state r.

The third component is either L or R and indicates whether the head moves to the left or right after writing.

Formal definition of TM

$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where

1. Q is the set of states
2. Σ is the input alphabet not containing the *blank symbol* $\sqcup$
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
4. $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$ is the transition function
5. $q_0 \in Q$ is the start state
6. $q_{accept} \in Q$ is the accept state
7. $q_{reject} \in Q$ is the reject state, where $q_{reject} \neq q_{accept}$

How a TM M computes

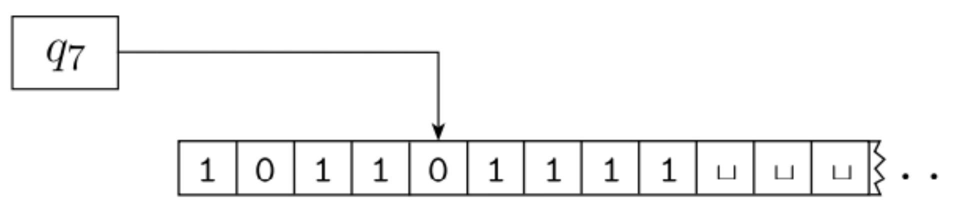
- Initially, M receives the input $w = w_1w_2\ldots w_n \in \Sigma^*$ on the leftmost n squares of the tape. Rest of tape is blank.
- The head starts on the leftmost square of the tape.
- Computation proceeds according to the rules specified by the transition function.
- If M ever tries to move its head to the left off the left-hand end of the tape, the head stays in the same place for that move, even though the transition function indicates L.
- Computation continues until it enters either the accept or reject states, at which point it halts.
- If neither occurs, M goes forever.

Configuration of a TM

- As a TM computes, changes occur in the
  - Current state,
  - Current tape content, and
  - Current head location.
- A setting of these three items is called a configuration of the TM.
- Configurations are represented in a special way.

Configuration of a TM

- For a state q and two strings u and v over the tape alphabet T, we write uqv for a configuration where the
  - Current state is q
  - Current tape content is uv, and
  - Current head location is the *first symbol* in v
  - The tape contains only blanks following the *last symbol* of v
- Example: $1011q_701111$ represents the configuration where the tape is 101101111, the current state is q_7, and the head is on the second 0.



Formalization of how TM computes

- We say that configuration C_1 yields configuration C_2 if the TM can legally go from C_1 to C_2 in a single step.
- Suppose that we have a,b and c in T, u and v in T*, and states q_i and q_j.
- In that case, uaq_ibv and uq_jacv are two configurations.
- We say that uaq_ibv *yields* uq_jacv if
  in the transition function $\delta(q_i,b) = (q_j,c,L)$
- We say that uaq_ibv *yields* $uacq_jv$ if
  $\delta(q_i,b) = (q_j,c,R)$

Formalization of how TM computes

- The **start configuration** of M on input w is the configuration q_0w
- In an **accepting configuration**, the state of the configuration is q_{accept}
- In a **rejecting configuration**, the state of the configuration is q_{reject}
- Accepting and rejecting configurations are **halting configurations**
- A TM M **accepts** input w if a sequence of configurations $C_1, C_2, \ldots, C_k$ exists, where

1. C_1 is the start configuration of M on input w,
2. Each C_i yields C_{i+1}, and
3. C_k is an accepting configuration

Turing recognizable and Turing decidable languages

- The collection of strings that M accepts is the **language of M**, or the **language recognized by M**, denoted by L(M)
- A language is called **Turing-recognizable** if some Turing machine recognizes it
  - Aka Recursively enumerable language
- When we start a TM on an input, three outcomes are possible:
  - accept
  - reject
  - loop (does not halt)
- A TM M can fail to accept an input by entering the q_{reject} state and rejecting, or by looping.
- Sometimes distinguishing a machine that is looping from one that is merely taking a long time is difficult.
- For this reason, we may prefer TMs that halt on all inputs; such machines never loop. These machines are called **deciders**.
- A language is called **Turing-decidable** if some language decides it.
  - Aka recursive language

Language of Turing Machines