CSE547T Class 10

Jeremy Buhler

February 20, 2017

1 Second Try at Modeling a Computer

- Our first attempt to build a computer used only finite memory, independent of the input size.
- This model of computation turned out to be very limited!
- We're going to try again with a new model that removes the "finite memory" limitation.
- More specifically, the computer is allowed to use as much memory as it needs for any particular input to carry out its computation.
- The new model will act much more like a "real" (but unbounded) computer.

2 Turing Machines

- The memory of our machine is a semi-infinite *tape*.
- The tape is divided into cells.
- Each cell may contain a symbol.
- Our machine has a "read-write head" whose *position* at any time is a particular tape cell.

- During each move, the machine can read symbol at current tape cell, write a new symbol, and move the head left or right by one cell.
- For convenience, we also give the machine finite control state.

Definition time!

- A *Turing machine* is a 5-tuple $(Q, \Sigma, \Gamma, q_0, \delta)$
- Q is a finite set of states (control)
- Σ is the input alphabet
- Γ is the *tape alphabet*
- Γ always contains at least Σ (and maybe more)
- There is also a unique "blank" tape symbol Δ in Γ. Δ is never a member of Σ.
- q_0 is start state for control
- $\delta : Q \times \Gamma \to (Q \cup \{h_a, h_r\}) \times \Gamma \times \{\text{left}, \text{right}, \text{stay}\}$ is transition function

How does a TM operate?

- For input $x \in \Sigma^*$, TM begins in following configuration:
  - state q_0
  - head in leftmost cell
  - tape contents are Δ in leftmost cell, followed by x in next $|x|$ cells, followed by infinitely many Δs

- If a TM is in state q, with symbol $a \in \Gamma$ under the head, and

$$\delta(q, a) = (q', b, \text{dir}),$$

- then the TM will transition to state q', overwrite current tape cell with symbol b, and move the head one cell in direction *dir* (will not move if *dir* is "stay").
- If TM transitions to state h_a or h_r, the computation ends, and we say that the TM has *halted*.
- If TM halts in state h_a, it *accepts*; else, it *rejects*

Here's a (rather stupid) example.

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{c, d\}$
- $\Gamma = \Sigma$
- $\delta(q_0, \Delta) = (q_2, \Delta, R)$

- $\forall a \in \Sigma$, $\delta(q_1, a) = (q_2, a, R)$
- $\forall a \in \Sigma$, $\delta(q_2, a) = (q_1, a, R)$
- $\delta(q_1, \Delta) = (h_r, \Delta, S)$
- $\delta(q_2, \Delta) = (h_a, \Delta, S)$
- Consider behavior on input "cddcd".

- TM accepts if length of input is even; otherwise, it rejects.

A couple of important points:

- So far, TMs are deterministic!
- A TM can run forever without accepting or rejecting (e.g. infinite loop)
- (Finite automata could not do this!)
- If a TM attempts to move off left end of tape, it *crashes* (equivalent to reject)
- If a TM has no legal move at any point, it crashes (equivalent to reject)
- $L(M)$, the language accepted by a TM M, is set of all strings x such that, when started with input x, M halts and accepts.

3 Basic TM Ops

- Talking about complex TMs can get tiring very quickly – their δs can be quite complex
- It's convenient to be a little "hand-wavy" when describing how a TM works, rather than writing down δ explicitly
- To make sure this is OK, let's enumerate some things that a TM can trivially do.
- **finite movement**: a TM can move a fixed number k of cells left or right, do something, then move back.
- Implement by a series of states in which the TM always moves left / right without touching its tape.

- **marking**: a TM can create a "mark" $\cdot$ on any tape cell.
- For every $c \in \Gamma$, define a new tape symbol $\dot{c}$.
- To mark a cell containing c, TM overwrites cell with $\dot{c}$.
- We can design a TM that makes any bounded number of distinct marks.
- **seeking**: a TM can repeatedly move left or right until it finds a cell containing a symbol c
- If TM is in state q and wants to move right until encountering c, can define moves for all $a \in \Gamma - \{c\}$:
$$\delta(q, a) = (q, a, R)$$
- **Implication**: a TM can always return to a specific marked cell later. For example, mark starting cell so that we can always "reset" head to left side.
- **cell memory**: a TM can remember the contents of a cell in its finite control.
- Augment the states of the TM to have form $\langle q, a \rangle$, where q is a control state and a is a tape symbol.
- When a TM wants to store the symbol under head in its memory (overwriting previous symbol), it can perform the transition
$$\delta(\langle q, a \rangle, b) = (\langle q, b \rangle, b, S)$$
- Generalizes to remembering any bounded number of symbols.

Here's a little example using these ideas.

- Let $L = \{ss \mid s \in \Sigma^*\}$
- **Claim**: there is a Turing machine accepting L.
- **Pf**: define a TM M with following behavior on input x.
- *Phase 0*
  1. If cell to right of starting cell is blank (x is ε), accept.
  2. Else determine whether x has even length, using above proc
  3. If x has odd length, reject; otherwise, return to starting cell and goto Phase 1.
- *Phase 1*
  1. move right, then mark first input symbol with mark m_0.
  2. seek right to end of input (cell before next blank), then mark last input symbol with mark m_1.
  3. seek left to m_0 (start of input) and goto Phase 2.
- *Phase 2*: **loop** on following steps

1. if right neighbor of current cell is marked with m_1, goto Phase 3
2. otherwise, seek right to m_1
3. remove m_1 from current cell and mark its left neighbor with m_1
4. seek left to m_0
5. remove m_0 from current cell and mark its right neighbor with m_0
6. move right one, to place head over new cell marked with m_0.

- *Observe*: when this loop terminates, mark m_1 will be on first symbol of second half of x!
- *Phase 3*

1. remove mark m_0 from current cell.
2. reset to leftmost cell.
3. move right one to first input symbol, and mark it with m_0
4. **loop** as follows to compare first and second halves of x:
   (a) remember symbol a under tape head.
   (b) seek right to m_1.
   (c) if cell with mark m_1 is blank, **accept** (moved off right end of x).
   (d) else if char with mark m_1 is not a, **reject**.
   (e) otherwise, move m_1 one posn to right, then seek left to m_0.
   (f) move m_0 one posn to right and leave head there.

Conclusion: TMs are powerful enough to accept non-regular languages!