CSE547T Class 11

Jeremy Buhler

February 22, 2017

1 Some Less Trivial Simulations

Regular TMs can be a pain to work with. Here are two ways to increase their expressiveness.

- Having just one tape is annoying
- It would be nice if a TM had *several* tapes
- Avoids having to track multiple pieces of (arbitrarily large) information on one tape
- **Defn**: a *k-tape TM* M has k tapes $t_1 \ldots t_k$, each with an independently movable head.
- By convention, input is always written to tape 1, and other tapes start blank.
- The formal defn of a multitape TM is as for a regular TM, except that its move function has form

$$\delta : Q \times \Gamma^k \rightarrow (Q \cup \{h_a, h_r\}) \times \Gamma^k \times \{L, R, S\}^k$$

- In one move, the TM reads the symbol under each of its k tape heads, then
  1. performs a single state transition
  2. writes a new symbol under each tape head
  3. moves each tape head by zero or one tape cells
- If *any* head moves off the left end of its tape, the TM crashes.
- (Note that, as a special case, we can think of a multitape TM that updates just one tape on each move.)

Is a multitape TM more powerful than a regular TM? Not obvious.

- **Claim**: every multitape TM can be simulated by a single-tape TM that accepts the same language.
- **Construction**: Let M be a 2-tape TM. (construction will extend easily to more than two tapes)
- Construct a single-tape TM M' as follows.

- *Idea*: every tape cell of M' stores contents of corresponding cells of *both* tapes of M
- Moreover, tape will store the *head positions* of both heads of M, so that we can track them.
- Let h_1, h_2 be two marks – the *head posns* of tapes 1 and 2.
- If M has tape alphabet Γ, then M' has tape alphabet

$$\Gamma' = (\Gamma \cup \{(c, h_1) \mid c \in \Gamma\}) \times (\Gamma \cup \{(c, h_2) \mid c \in \Gamma\})$$

- Moreover, M' tracks state of M in its finite control
- M' simulates a generic move of M as follows
- To begin, move head of M' to left end of tape (using standard marking trick)
- First, M' seeks right to find first head posn h_1
- M' caches the symbol a_1 under h_1, then returns to left end of tape
- Second, M' seeks right to find second head posn h_2
- M' caches the symbol a_2 under h_2, then returns to left end of tape
- Suppose $\delta(q, a_1, a_2) = (q', b_1, b_2, d_1, d_2)$.
- To update tape 1, M' seeks right to first head posn and replaces (a_1, h_1) by b_1, while leaving the other part of the symbol intact
- M' then moves one step in direction d_1 and marks the first part of the current cell's symbol with h_1
- Finally, M' then moves back to the left end of the tape and repeats the update for h_2, b_2, and d_2
- If M has not accepted or rejected, M' continues with next move of M.

2 Nondeterminism

That was pretty easy. What about something more powerful?

- **Defn**: a *nondeterministic TM* (NTM) is a Turing machine with a nondeterministic move function

$$\delta : Q \times \Gamma \to 2^{(Q \cup \{h_a, h_r\}) \times \Gamma \times \{L, R, S\}}$$

- Given current state and tape symbol, an NTM chooses next move from allowed set nondeterministically.

- An NTM accepts an input x iff it can accept x by a computation using *some* set of nondeterministic choices.
- (When does an NTM reject an input? If every possible computation rejects it. If no computation accepts and some computation runs forever, then the NTM runs forever.)

Nondeterminism does not make finite automata more powerful. What about TMs?

- **Claim**: every NTM M can be simulated by a deterministic TM M' that accepts the same language.
- *Idea*: M' could successively simulate M with each possible set of nondeterministic choices.
- *Problem*: there are infinitely many such sets of choices, since they can be arbitrarily long
- Also, even a finite set of choices could get M into an infinite loop, so we'd never finish the simulation!

We will need two tricks: interleaving and enumeration!

- To avoid the run-forever problem, we use *iterative deepening*.
- WLOG, suppose that for M, each $\delta(q, a)$ w/defined transition has exactly k possible moves. Number these moves $0 \ldots k-1$ in some order.
- A computation of M can be described by a k-ary tree:

- At each tree node, M chooses one of k possible moves.
- Leaves correspond to computations that halt.
- M' first explores all paths in tree up to depth 1
- If any such path accepts, we are done.
- Otherwise, explore all paths up to depth 2, then 3, and so on.
- (We effectively interleave all possible computations of M)
- *Challenge*: how to enumerate all paths of depth k?

- Any set of m nondeterministic moves can be described by a base-k integer $c_1 c_2 \ldots c_m$, specifying which choice is taken for each move.
- To enumerate all possible computations of m moves, suffices to enumerate all base-k integers with m digits
- Can a TM count in base k? Sure!

Construction: M' is a 3-tape TM.

- Tape 1 is "work tape" that contains current tape state of M
- Tape 2 stores a copy of *initial* tape state for M
- Tape 3 enumerates current set of nondeterministic choices as a base-k integer
- M' first copies input to tape 2 (up to first blank)
- M' then runs following outer loop for each $i > 0$
  1. Initialize tape 3 to "0^i" (one more 0 than previous loop iter)
  2. Run following inner loop until tape 3 contains $\{k-1\}^i$ (all cells up to first blank contain $k-1$)
     (a) *Erase* tape 1 completely and copy tape 2 to tape 1 (exercise)
     (b) Simulate M on tape 1 for as many moves as there are digits on tape 3 (move tape 3's head right one step each time, until we see a Δ)
     (c) For jth move, simulation makes choice corresponding to jth digit on tape 3
     (d) If M accepts, accept the input.
     (e) Otherwise, if i moves have elapsed without acceptance, increment number on tape 3 and repeat loop.