CSE547T Class 13

Jeremy Buhler

March 1, 2017

1 The Universal Turing Machine

- FAs can simulate other FAs.
- That is, given one machine M, we can define another M' that behaved "the same" as M.
- We've done this for TMs too, repeatedly.
- The catch: simulation has always been "manual" – we are *given* M, and we manually design M' from it.
- (Think of a TM or FA as equivalent to a program. Simulation entails our writing a new program that behaves like the first.)
- However, a FA is not smart enough to take an FA as *input* and evaluate its behavior on a string.

Turing machines are powerful enough to take other TMs as input and run them!

- Let $M = (Q, \Sigma, \Gamma, q_0, \delta)$ be a TM.
- We will define an encoding e mapping TMs and their inputs to strings in $\{0,1\}^*$.
- Define the following mapping from states $q_i \in Q$, symbols $a_i \in \Gamma$, and directions in $\{L, R, S\}$ to unary integers:

$$\begin{aligned}
[h_a] &= 0 \\
[h_r] &= 00 \\
[q_i] &= 0^{i+2} \\
[\Delta] &= 0 \\
[a_i] &= 0^{i+1} \\
[L] &= 0 \\
[R] &= 00 \\
[S] &= 000
\end{aligned}$$

- For each move $m =$ "$\delta(q, a) = (q', b, d)$", define encoding

$$e(m) = [q]1[a]1[q']1[b]1[d].$$

- Then, if δ consists of a list of moves $m_1 \ldots m_j$,

$$e(M) = 1e(m_1)1e(m_2)1 \ldots 1e(m_j)1.$$

- (q_0 is WLOG "000"; alphabet and list of states is implicit in move function)
- Moreover, for any string $w = a_1 a_2 \ldots a_n$ define

$$e(w) = 1[a_1]1[a_2]1 \ldots 1[a_n]1.$$

So what?

- We will define a *universal TM* M_u that can take an encoded TM M and string w as input and execute M on w.
- M_u has three tapes: its *input tape*, a *work tape*, and a *state tape*
- On reading input x, M_u checks whether x has the form $e(M)e(w)$, for a TM M and a string w.
- (Checking is simply counting that we have right number of numeric fields for M, followed by 11, followed by encoded string w.)
- If x does not have right form, reject.
- Otherwise, M_u simulates M on w as follows.
- First, write $[q_0]$ to state tape (start state)
- Then, M_u prepares its work tape by writing $e(\Delta \cdot w)$ and moving work tape head to start of leftmost symbol.
- Finally, M_u simulates M on x by the following loop:
  1. Sequentially check the moves of M encoded on the input tape.
  2. If we find a move $\delta(q, a) = (q', b, d)$, such that
     - state on state tape is $[q]$,
     - symbol starting at work tape head posn is $[a]$,

     then execute the move as follows:
     - erase state tape and replace contents with $[q']$
     - erase encoded symbol under head on work tape and replace with $[b]$
     - (Exercise: may need to move rest of work tape forward or backward so as not to leave a "hole" if $[b] \neq [a]$).
     - move head on work tape one complete symbol in direction d.
     - (if $d \neq S$, find the next "1" in dir d and go one more step)
     - If we move off left end of work tape, crash (reject).
     - If we move past last symbol encoded on work tape, write new $e(\Delta)$.
     - If state tape contains $[h_a]$ (resp. $[h_r]$), accept (resp. reject).
  3. If no move matching current state and symbol is found, crash (reject).

- (When M accepts or rejects, M_u can erase its input tape and copy work tape back to input tape, so that its output is the encoded *output* of M_u. Needed only if we want to look at output of M.)
- Note that, when input is a proper TM and string encoding, M_u accepts precisely when M does and rejects precisely when M does.

2 Non-RE and Non-Recursive Languages

So, what can computers do?

- **Lemma**: there exists a non-RE language.
- **Pf**: Consider the language

$$\text{NSA} = \{e(M) \mid M \text{ is a TM and } e(M) \notin L(M)\}.$$

- NSA is the language of *non-self-accepting* TMs.
- Suppose we could find a TM M_{NSA} that accepts the language NSA.
- If $e(M_{NSA}) \in$ NSA, then M_{NSA} must accept $e(M_{NSA})$.
- But then M_{NSA} accepts its own encoding, and so $e(M_{NSA}) \notin$ NSA. Contradiction!
- OK, so it must be that $e(M_{NSA}) \notin$ NSA, and so M_{NSA} fails to accept $e(M_{NSA})$.
- But then M_{NSA} does not accept its own encoding, and so $e(M_{NSA}) \in L$. Contradiction!
- Conclude that M_{NSA} cannot exist, and so NSA is not RE.

Apparently, computers cannot test certain self-referential properties!

- **Lemma**: there exists an RE but non-recursive language.
- **Pf**: Consider the language

$$\text{SA} = \{e(M) \mid M \text{ a TM and } e(M) \in L(M)\}$$

- SA is the language of *self-accepting* TMs.
- First, we claim that SA is RE.
- Build a TM M_{SA} accepting SA as follows.
- On input x, M_{SA} first checks that x encodes a TM, that is, $x = e(M)$ for some M. If not, M_{SA} rejects x.
- Then, M_{SA} runs universal TM M_u on input $e(M)e(e(M))$.
- M_{SA} accepts x iff M_u accepts its input, that is, iff M accepts $e(M)$.
- Conclude that SA is RE.

That proves RE-ness, but why isn't SA recursive?

- Suppose to the contrary that SA is recursive.
- Then $\overline{\text{SA}}$ is RE (in fact, recursive).
- Suppose we had a TM M_{CSA} accepting $\overline{\text{SA}}$.
- Now $x \in \overline{\text{SA}}$ iff either x does not encode a TM *or* $x = e(M)$, where M is a non-self-accepting TM.
- Consider following TM M':
- On input x, M' first checks if $x = e(M)$ for some TM M. If not, M' rejects x.
- Otherwise, M' simulates M_{CSA} on x and returns the latter's result.
- **Obs**: M' accepts x iff x encodes a TM *and* $x \in \overline{\text{SA}}$, i.e. iff x encodes a non-self-accepting TM.
- Conclude that M' accepts NSA, which is impossible because NSA is not RE.
- Conclude that SA must not be recursive. QED

Self-acceptance for TMs is an *undecidable problem*: the language of TMs that self-accept cannot be decided by any TM!