Coding Turing Machines – The Universal Turing Machine

Universal alphabet:

$A^* = \{a_1, a_2, a_3, \ldots\ldots\}$ ∞ set

Universal state set:

$Q^* = \{q_1, q_2, q_3, \ldots\ldots\}$ ∞ set

Given a TM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$

when $|Q| = r$ and $|\Gamma| = t$

Rename Q to $\{q_1, q_2, \ldots, q_r\}$ and

Γ to $\{a_1, a_2, \ldots, a_t\}$

Conclusion: Every TM M has an equivalent TM M' with $Q(M') \subseteq Q^*$ prefix and $\Gamma(M') \subseteq \Gamma^*$ prefix

Use binary numbers $q_7 = q_{111}$ $a_5 = a_{101}$

Code for a TM M (assume M is deterministic)

List of tuples of the kind $((q_i, a_j), (q_s, a_q, \})) $

where $\} \in \{R, L, S\}$

When $r = |Q(M)|$ and $t = |\Gamma(M)|$

we may assume $q_1 \sim q_0$ (initial), $q_{r-1} = q_{accept}$ and $q_r = q_{reject}$

and now we can write the code of M as

$\langle M \rangle = ((q_1, a_1), (q_{f_1}, a_{q_1}, R)), ((q_1, a_2), (-,-,-)), \ldots, ((q_{r-2}, a_t), (q_b, a_e, L))$

Similarly we can code strings $w \in \Gamma^*$:

If $w = a_7 a_1 a_2 a_3$ then $\langle w \rangle = (a_7)(a_1),(a_2),(a_3)$

So we can code all Turing Machines

using the alphabet

$\Sigma = \{ '(', ')', 'a', 'q', '0', '1', ',', 'R', 'L', 'S' \}$

Universal Turing machine U

Takes input of the form

$\langle M \rangle \langle w \rangle$

and uses 3 tapes to simulate M on w:

Initially:

tape 1 $\triangleright \langle w \rangle$

tape 2 $\triangleright \langle M \rangle$

tape 3 $\triangleright q_1$

One step of simulation:

Let q = state on tape 3

a = symbol under head on tape 1

Find entry for (q, a) on tape 2 and perform changes on tapes 1 and 3: If entry is $((q,a)(p,b,L))$ then

new symbol on tape 1 is b and head moves left

new state on tape 3 is p

If $p = q_{r-1}$, then U enters its accept state and stops

If $p = q_r$, the U enters its reject state and stops

Easy to see that:

U accepts/rejects $\langle M \rangle \langle w \rangle$

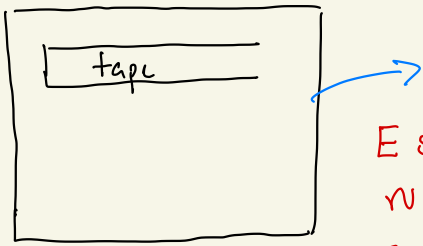
$\Updownarrow$

M accepts/rejects w

and the deterministic TM M loops on w if and only if U loops on $\langle M \rangle \langle w \rangle$

Enumerators

Turing machine with an output tape



E

first string printed

$w_1 \Delta w_2 \Delta w_3 \Delta \dots$

E starts on the empty string and runs forever

From time to time E prints a string w_i on its output tape

$L(E) = \{ w \mid w \text{ is printed by } E \}$ ← over ∞ time

L is enumerable if $L = L(E)$ for some enumerator

E.g. $L = \{ 0^{2^n} \mid n \geq 0 \}$ is enumerable

Theorem 3.21 L is enumerable

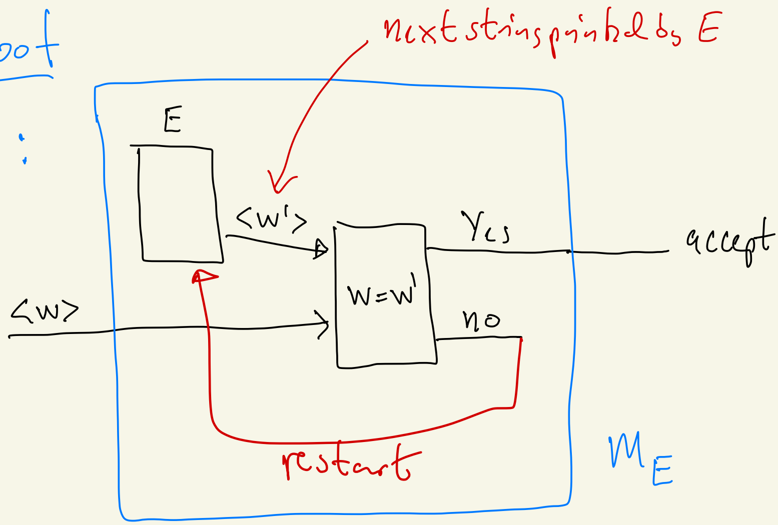
$\Updownarrow$

L is recognizable

($L = L(M)$ for some DTM M)

Proof

$\Downarrow$:



- On input $\langle w\rangle$ the TM M_E starts the enumerator E on the empty string
- whenever E prints a string w', M_E compares w' with w
- If $w = w'$, M_E accepts w

  Else it restarts E and waits for next string w' printed by E

Clearly $L(M_E) \subseteq L(E)$ since M_E can only accept w if it is printed by E

Also $L(E) \subseteq L(M_E)$ since

$w \in L(E) \Rightarrow E$ eventually prints w.

⇑ Let $L = L(M)$ for a DTM M

Let Σ be the input alphabet for M
and order the strings in Σ^* as

$s_1, s_2, s_3 \cdots s_k, s_{k+1} \cdots$

lexicographically

E_M: For $i = 1$ to ∞

round i { Simulate M for i steps on each of the strings $s_1, s_2 \ldots, s_i$ and print s_j if M accepts it in less than $i+1$ steps

$\# s_1 \# s_2 \# \cdots \# s_i \#$ ← strings listed lexicographically on one of E_M's tapes

Suppose M accepts w after p steps and $w = s_k$ (k'th string in lex order of Σ^*)

Then E_M will print w in every round n where $n \geq \max\{p, k\}$

So $w \in L(M) \Rightarrow E_M$ prints w (∞ many times)

and E_M only prints strings from $L(M)$

Hence $L(E_M) = L(M)$

Hilbert's 10'th problem

$D = \{\langle p \rangle \mid p$ is a polynomium with an integral root$\}$

ex of polynomium

$p(x,y,z) = 6x^2y^3z - 8xyz + 2y^2z^6$

$x = y = z = 1$ is a root as $p(1,1,1) = 6 - 8 + 2 = 0$

Hilbert asked for an algorithm to decide D

It turns out that no such algorithm exists !

simpler problem $D_1 = \{\langle p \rangle \mid p$ is a polynomium over one variable x which has an integral root$\}$

Both D an D_1 are Turing-recognizable as a NDTM can guess a solution when one exists.

A deterministic TM for D_1 can check for x in

$0, 1, -1, 2, -2, 3, -3 \dots$

This TM can be turned into a decider since

$p(x) = c_1x^n + c_2x^{n-1} + \dots + c_1x + c_0$ has a root

if and only if it has a root $x \in \{-(n+1)\cdot\frac{c_{max}}{|c_1|}, \dots, (n+1)\frac{c_{max}}{|c_1|}\}$

Hence the TM only needs to check values in this interval

Church-Turing thesis

$\exists$ algorithm for a decision problem L

$\Updownarrow$

$\exists$ DTM M which decides L

We can use encodings a la $\langle M \rangle$ and $\langle w \rangle$ for all decision problems

$A = \{\langle G \rangle \mid G \text{ is a connected graph}\}$

$\langle G \rangle = (v_1, v_2, \dots, v_n), (e_1), (e_2), \dots, (e_m)$

where $(e_i) = (u_i, w_i)$ for some pair $u_i, w_i \in \{v_1, v_2, \dots, v_n\}$ of distinct vertices

Given $\langle G \rangle$ a DTM can check whether $\langle w \rangle \in A$ by performing a Breath-First-Search from v_1