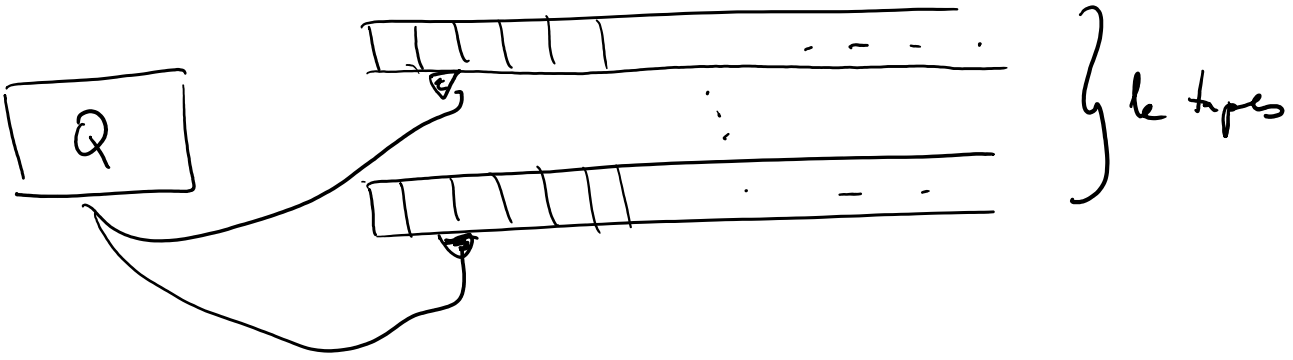
Universal Turing Machine

8 January, 2021 14:19

Turing Machines



multiple tapes, one-way infinite

Q ... set of states
Γ ... work tape alphabet } finite sets
$\Sigma \subseteq \Gamma$ ... input alphabet
$\sqcup \in \Gamma$ ... blank symbol
$q_{init}, q_{acc}, q_{rej}$ ... initial, accepting, rejecting state

$$\delta : Q \times \Gamma^k \longrightarrow Q \times (\Gamma \times \{L, R\})^k$$

... transition fcn.
# of tapes ... k

each TM can be described by a string encoding

$$Q, \delta, \Gamma, \Sigma, q_{init}, q_{acc}, q_{rej}$$

e.g. $(\{000, 001, 010, \ldots 101\}, \{00, 01, 10, 11\}, \ldots,$

Γ of size 5 Γ of size 4

Q of size 5 Γ of size 4

$\{(000, 01, 010, 10, L), \ldots 5)$

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

Over alphabet $\{0, 1, \{, \}, (,), ,\} = \Sigma$

W.L.O.G $\Sigma' = \{0, 1\}$

each TM can be described by a string from $\{0,1\}^*$.

each string w from $\{0,1\}^*$ corresponds to an integer with binary representation $1w$.

→ each TM can be assigned a number which describes it.

- single TM corresponds to infinitely many descriptions (numbers), by blowing up the encoding of Q, or Γ, or adding unused states to Q, etc.
- each integer n (>0) corresponds to some binary string w, $n = 1w$. If w corresponds to a valid encoding of a TM, define M_n to be that TM. If w is not a valid description define M_n as a TM which always rejects

$(\mathcal{L}_{int} = \mathcal{L}_{REG})$.

→ easy to check which is the case.

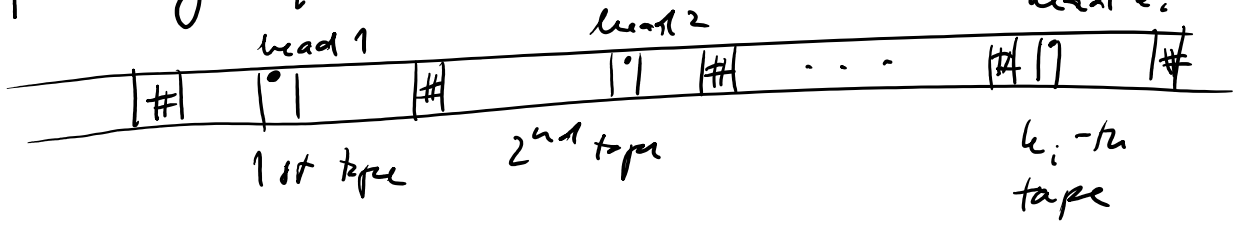
TM's: $M_1, M_2, \ldots M_n \ldots$ Enumeration of all TM's. Each TM appears infinitely often.

Universal Turing Machine

U ... TM that receives as its input pair (i, x), where i is an integer encoded in binary & x is a binary string, interprets i as a description of a TM M_i, x as an input encoded in alphabet of M_i & simulates M_i on x.

Representing k_i-tapes of M_i on the tape of U

1)

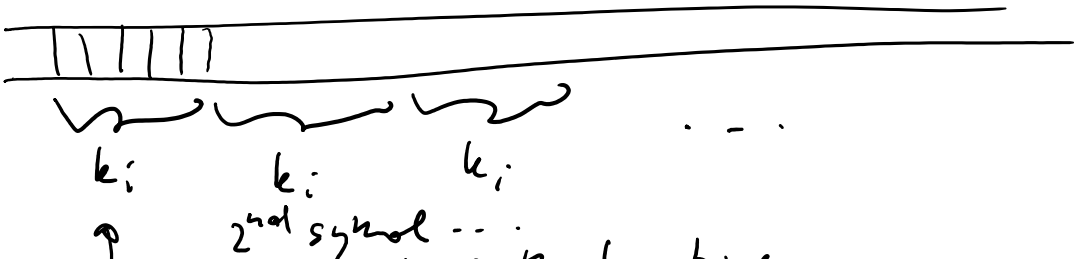


Content of each simulated tape is a block on U's work tape. To simulate one step of M_i, needs to collect info about symbols read by heads on all k_i tapes, determine what step to take and update the content of simulated steps. The update might

require shifting content of the tapes to make room for another cell on simulated tapes.

→ the l^{th} step of the simulator can take time $\leq c_i k_i l$, where c_i is a constant that depends on i but nothing else.

2)



1st symbol of each of the k_i tapes
2nd symbol ...

→ don't need to shift the tapes to make room for new cells

still, l^{th} step takes time $\leq c_i k_i l$.

⇒ in 1) & 2) simulates for T steps of M_i takes time $O(T^2)$

→ improvement to $O(T \lg T)$:

3) heads of all simulated tapes will be in the same posts of U. → will be shifting the "whole" simulated tapes on the tape of U!

Single simulated tape alg: (two-way infinite)

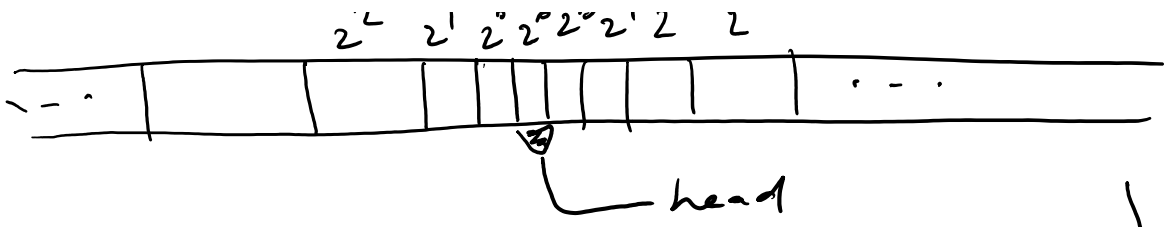
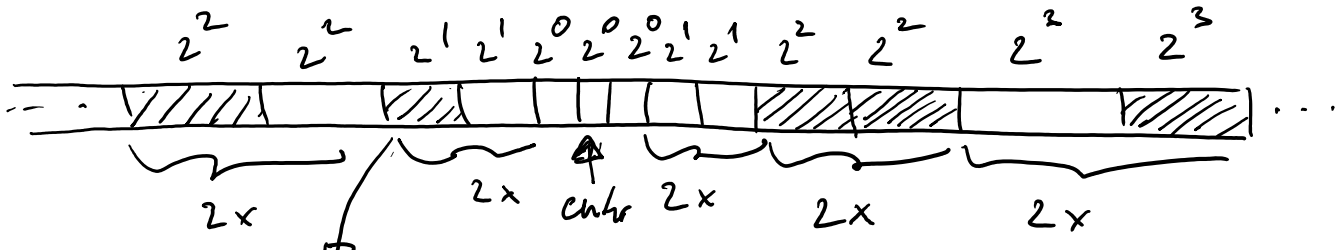
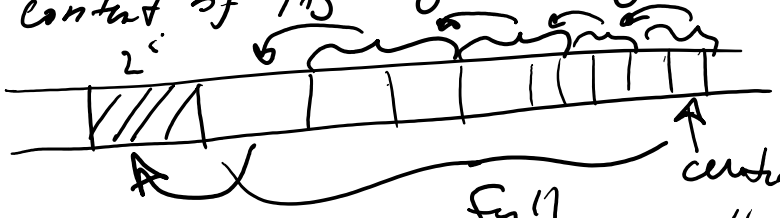


illustration picture, not quite accurate — on tape of $\mathcal{U}$



some blocks might be empty

Case 1: need to shift symbol from center to the left. → Find the first empty block to the left, say its size is 2^i.
If it's the left block, move to it the content of its right neighbor of size 2^i

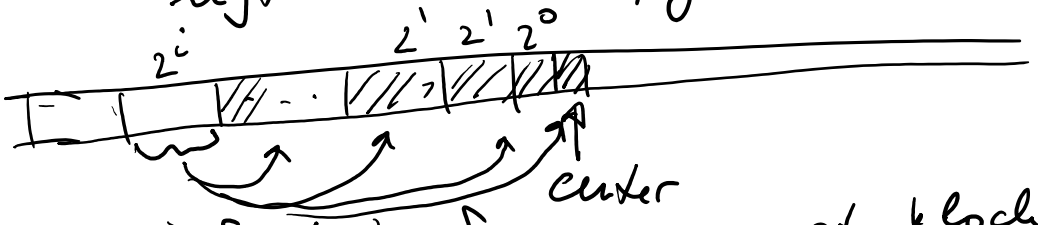


and then move the content of the two full blocks of size 2^{i-1} in its place. Repeat until you free the center.
If the 2^i block is the right block in a pair proceed similarly.

Observe: After the procedure, each pair of blocks of size 2^j, $j < i$,

containing one full & one empty block.

Case 2: need to shift a symbol from left to the (empty) center.



→ find the first non-empty block to the left, say its size is 2^i. Subdivide it into blocks of size $2^{i-1}, 2^{i-2}, 2^{i-3}, \ldots 2^0, 2^0$ and distribute them so that the center gets the rightmost symbol and block of size 2^j, $j<i$, goes into one of the empty pairs 2^j.

Observation: After the procedure, in each pair of blocks 2^j, $j<i$, one block is empty & the other is full.

- When we need to move symbol to/from right of center, proceed symmetrically.

Corollary: If we reach a block of size 2^i during shifting the center left/right, all pairs of blocks of size 2^j, $j<i$, contain

one Full & one empty block.

(Possible exception is for $j=0$)

- we use a second tape to move a block in linear time. (copy in & out.)
- when we reach a block of size 2^i, we spend $O(2^i)$ time to process the moves as $\sum_{j<i} 2 \cdot 2^j = 2 \cdot (2^i - 1)$.
- during execution of T steps of simulation we reach block of size i at most $\frac{T}{(2^i-1)}$ times

Reason: to reach it again we either have to pump in $2^i - 1$ symbols to fill up the empty blocks or we need to pump out $2^i - 1$ symbols to empty the full blocks 2^j, $j<i$. ($2^j, j<i$)

→ we spend time $O(2^i) \cdot \frac{T}{2^i - 1}$ in total whenever we reach block of size 2^i.

$= O(T)$.

There are at most $\lg T$ block size we can reach ⇒ $O(T \lg T)$ total

time.

Universal Nondeterministic Machine

- one can simulate nondeterministic computation running for T steps in $c_i \cdot k_i T$ steps. (nondeterministically)

Pf: idea

guess the history of computation of M_i on input x for T steps. Each step of the history captures what the k_i heads currently see, in what state the machine is and what transition it will make
$\Rightarrow$ the history is of length $O(k_i T)$.

For $j = 1, \dots k_i$, "replay" the content of j^{th} tape on a second tape of M & see it it is consistent with history.

Check if the history is consistent with the transition fcn of M_i

If all checks pass, and the simulated computation accepts $\rightarrow$ ACCEPT o/w REJECT

□