Multitape TMs

Let $k \geq 1$. A k-tape TM is a tuple

$$M := \langle Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej} \rangle$$

where everything besides δ is as with a (1-tape) TM

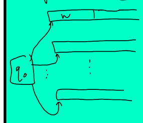
and

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, S, R\}^k$$

L - move 1 cell left
R - move 1 cell right
S - don't move ("stationary")

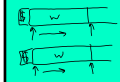
Initial conditions:

Input $w \in \Sigma^*$:



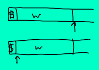
w is on 1st tape (as with a 1-tape TM) & all other tapes blank

Ex: Deciding $\{0^n 1^n : n \geq 0\}$ in linear time on a 2-tape TM



1. scan input tape, check that input is in $0^* 1^*$ (otherwise reject) & copy w onto 2nd tape

2. move 2nd tape head back to the beginning:



3. heads move in opposite directions:

if the same symbol is seen on the scanned cells, the reject

4. else accept

(Implementation-level algo).

Our previous approach on a 1-tape TM takes $\Theta(n^2)$ time $(n = |w|)$

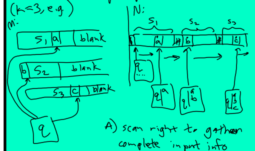
A 1-tape machine can decide this language in time $O(n \lg n)$

Thm: A 1-tape TM running in time $o(n \lg n)$ can only recognize a regular language.
(won't prove)

Thm: Any k-tape TM M can be simulated by a 1-tape TM N, and N simulates t steps of M in time $O(t^2)$
[time = # of steps].

Proof sketch: To simulate M, N concatenates the contents of M's tapes on one tape, separated by some marker (say #) not in M's tape alphabet.

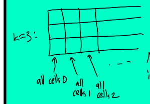
(k=3, e.g.)



A) scan right to gather complete input info to M's transition function

B) scan left, implementing the changes according to M's transition function (hard-coded in N). //

Alternative simulation:



"multitrack tape"

N's tape alphabet is $(\Gamma \cup \dot{\Gamma})^k \cup \{\sqcup\}$

"Reasonably" Encoding various finite objects as binary strings:

- natural numbers [binary rep.]
- integers [2's complement, sign magnitude]
- rational numbers [n#d : n, d numerator & denominator]
- strings over an arbitrary alphabet [$a \mapsto 00$, $b \mapsto 01$, $c \mapsto 10$, $d \mapsto 11$] e.g. 4-letter alpha
- lists (finite) lists of finite objects

"Reasonable" means: for any basic computational operation (e.g. addition, comparison, concatenation, etc.), there is an algo (TM) that performs that operation on the encoding.

<u>Notation</u>: For any finite math object $\mathcal{O}$, we let $\langle \mathcal{O} \rangle$ be the string encoding $\mathcal{O}$ (via some fixed, reasonable encoding).

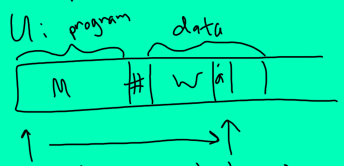
Finite lists of finite objects: Given objects $\mathcal{O}_1, \mathcal{O}_2, \dots, \mathcal{O}_n$ we could encode these into a single string

$$\langle \mathcal{O}_1, \dots, \mathcal{O}_n \rangle := \langle \langle \mathcal{O}_1 \rangle \# \langle \mathcal{O}_2 \rangle \# \cdots \# \langle \mathcal{O}_n \rangle \rangle$$

More finite objects:

- DFAs, NFAs, regexes
- TMs:
  - finite functions
- graphs
- arrays
- trees, etc.

<u>Theorem</u>: There exists a TM U such that, on any input $\langle M, w \rangle$ where M is a TM & w is a string (over M's input alphabet), simulates the computation of M on input w, i.e. accepts/rejects/loops on input $\langle M, w \rangle$ the same as M does on input w.



sees what symbol is being scanned, recording M's state in U's state, looking up the entry in M's desc. of its transition, and performing it on the "data" portion.

U is called a <u>universal TM</u>.