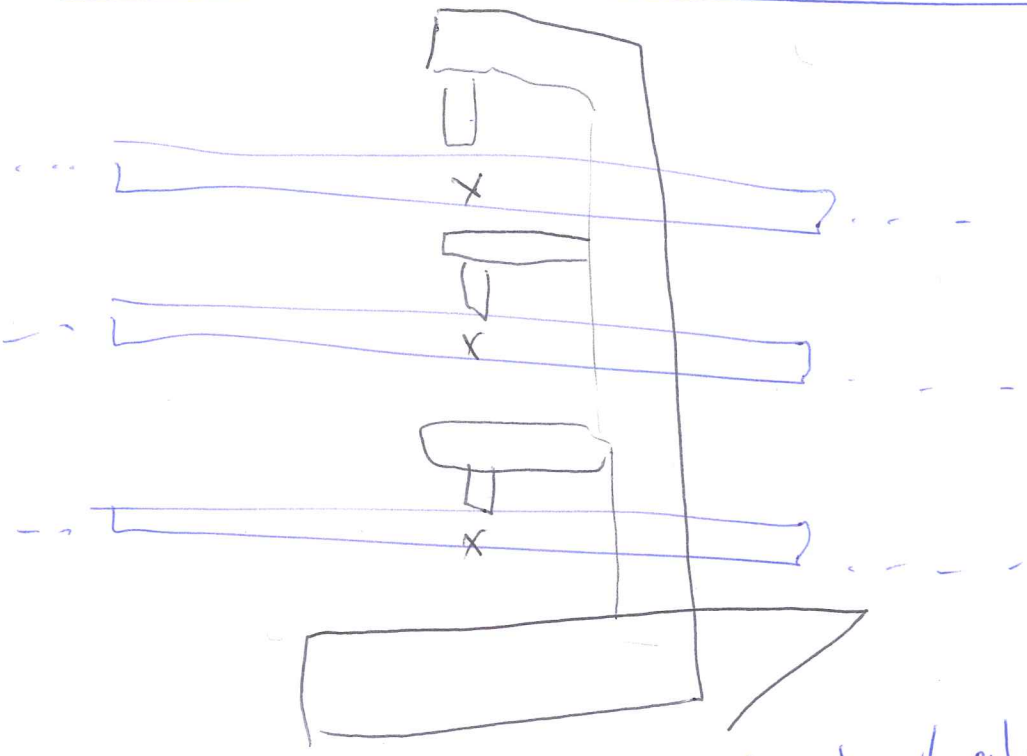
# Multi-head (multi-tape) Turing Machine

11/7/18



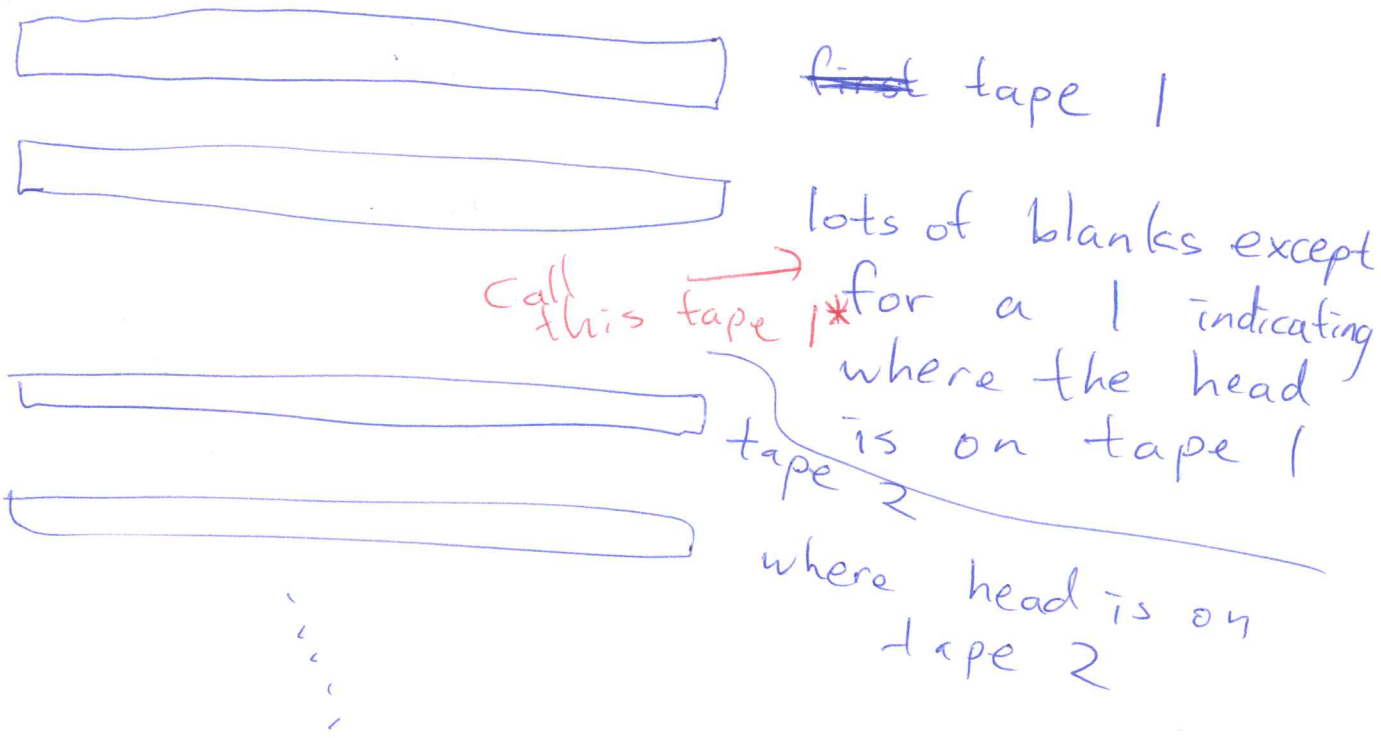
(each w/ alphabet $\Gamma_k$)

We have k tapes - transition function looks like

$$\delta : Q \times \Gamma_1 \times \dots \times \Gamma_k \longrightarrow Q \times \Gamma_1 \times \dots \times \Gamma_k \times \{L, R\}^k$$

Goal: Explain how we could mimic the functionality w/ a standard (i.e. 1 head, 1 tape) TM.

Idea: Have a multi-~~head~~ tape (1-head) TM where the tapes looks like:




The multi-tape TM will have state space

$Q \times$ { some complicated set of states controlling the simulation }

↑ orig states

↑ includes a lot of states remembering part of the current input.

Idea of emulating program:

look on tape 1* for the 1,
read the input on tape 1
(going into a state that remembers it)
look on tape 2* for the 1
read the input on tape 2
(going into a state that rembers both
tape 1 + tape 2 input)

⋮

read input on tape k,

use current state to determine
what emulated state to go into
and what to write and move
on all the tapes (which we
remember in a state)

find the 1 on tape 1* again
write new symbol on tape 1
move the 1 on tape 1*
find the 1 on tape 2*

⋮

This is a sketch of a proof that,

Thm: for every multi-head TM $M$, there exists a standard TM $\hat{M}$ such that the language accepted by $M$ equals the language accepted by $\hat{M}$.

---

## Nondeterministic TMs

This means, given the current state + symbol, we have several possibilities for new state, new symbol, and movement.

i.e. the transition function becomes a ~~finite~~ relation (~~w/ fintely many choices for each~~

$$\delta \subseteq (Q \times \Gamma) \times \left[ (Q \times \Gamma \times \{L, R\}) \cup \{\not\exists\} \right]$$

We can't talk about these nondet TMs giving output, but we can view them as accepters:

If there is, at least one path to acceptance, it accepts the ~~str~~ input.

**Thm:** Given a non-det TM $M$, there exists a ~~det~~ standard TM $\hat{M}$ that accepts the same language.

**Sketch of Pf:** Do a breadth-first search through all the choices.

(Note - the tree of choices has ~~finite~~ bounded width, but unbounded height, so a DFS can go down a rabbit hole and never emerge)

This means:

① Do the comp. that takes choice 1

② " " " " " " 2

⋮

ⓚ " " " . . . . . k

(assume at most k choices at each step)

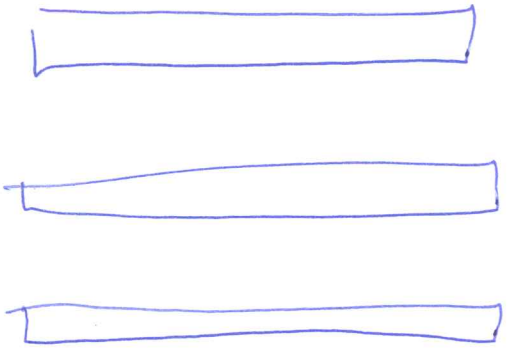
Do the comp. that takes choice 1, then choice 1

" " " " choice 1, then choice 2

When the BFS gets to an accepting, halting state, ~~it~~ the whole TM halts and accepts; otherwise it keeps going.

How does $\hat{M}$ work?

Make a 3-tape, 3-head TM.



tape 1 - the original input

tape 2 - the ~~output~~ computation currently being performed

tape 3 - the choices we should make right now.

Write input on tape 1 (which never chages)

Copy tape 1 to tape 2

Increment tape 3 (as a base k number (a base k number, continually incremented)

~~left to right, big endian~~)

Do computation on tape 2 - when we have choices, consult tape 3, when out of choices, ~~either~~ erase tape 2, repeat