More complicated versions of TMs are not more powerful:

For PDAs - we know we can modify them - e.g. by adding a second stack - so that the new class of machines can accept a language no machine in the old class can.

We'll see ~~to~~ over the next week that various ways we can think of to enhance TMs don't make them more powerful (in the sense of being able to do something the old model can't (be programmed to))

Def'n: A machine is some mathematical object - a class of machines is just some set $\mathcal{C}$.

Eg. C could be the class of DFAs.

$C = Q \times \Sigma \times$ [illegible]

$C = \{$ all objects of the form $(Q, \Sigma, \delta, q_0, F)\}$

Given a class of machines, we also have a function

$L : C \longrightarrow \{$ sets of strings $\}$

$L(M)$ = the language M accepts,

for any $M \in C$

Defn: Two classes $(\mathcal{C}, L)$, $(\mathcal{C}', L')$ of machines (along w/ their "language functions") are equivalent if, for every

$M \in \mathcal{C}$, there exists $M' \in \mathcal{C}'$

so that $L(M) = L'(M')$

and, for every $M' \in \mathcal{C}'$,

there exists $M \in \mathcal{C}$

so that $L(M) = L'(M')$.

First idea for more powerful TMs:

- ~~of~~ my TMs are also allowed to sit at the same place after reading a letter, changing state, and writing a letter.

Formally: δ is now a function

$$Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R, S\}$$

There's a new description of how instant. descr. change on S's.

We want to prove that

{TM-with-stay-option}

is equivalent to

{~~TM~~ standard-TM}

I have to prove

1) For every standard-TM, M, there is a TM-with-stay-option M' so that $L(M) = L'(M')$.

2) For every TM-w/stayoption M', there is a ~~$\exists$~~ standard-TM so that $L(M) = L'(M')$.

---

(1) is easy- ~~pass~~ given M, have M' be the same - never actually use the stay option.

(2) - given M', we make an M where the states of M are

$$\left(\begin{array}{l}\text{state} \\ \text{of} \\ M'\end{array}, \begin{array}{c}\text{normal} \\ \text{or} \\ \text{remember-to-move-right}\end{array}\right)$$

And, every transition

$$(q, \ell) \longrightarrow (q', \ell', L \text{ or } R)$$

in M' gives a transition

$$(q\text{-normal}, \ell) \longrightarrow (q'\text{-normal}, \ell', L \text{ or } R)$$

in M, transitions

$$(q, \ell) \longrightarrow (q', \ell', S) \text{ in } M'$$

gives transitions

$$(q\text{-normal}, \ell) \longrightarrow (q'\text{-remember to move } R, \ell', L)$$

in M,

and ~~we~~ for every state q (of M') and every letter ℓ, we have a transition

$(q\text{-remember-to-move}_{R}^{L}, \ell) \rightarrow (q\text{-normal}, \ell, {}_{R}^{L})$

Then we're supposed to write a proof that $L'(M') = L(M)$ - we'll skip this - it's pretty obvious.

In general, we'll do these arguments w/ less formal detail in the future.

Next idea: I get 5 tapes (but still only one read/write head).

So I read 5 inputs, ~~w~~ change state, write on all 5 tapes, then move right or left (the same way on all the tapes).

~~Thm.~~

Equivalence is trivial - I can make my tape alphabet what I want.

So I just make ~~a~~ a ~~TM~~ single tape TM where ~~my alph~~ tape alphabet is

a letter in my new tape alphabet is a ~~list~~ stack of 5 letters from the old tape alphabets squished into a super-letter.

i.e. $\Gamma_M = \Gamma_{M'} \times \Gamma_{M'} \times \Gamma_{M'} \times \Gamma_{M'} \times \Gamma_{M'}$

($\square_M = (\square_{M'}, \ldots, \square_{M'})$, ~~etc~~)

$\Sigma_M = (\Sigma_{M'}, \square_{M'}, \ldots, \square_{M'})$