Review:

**Def**: A Turing Machine is a 7-tuple:

1. \( Q \): set of states
2. \( \Sigma \): input alphabet
3. \( \Gamma \): tape alphabet \((\Sigma \subseteq \Gamma)\)
4. \( \delta \): \( Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\} \) (transition function)
5. \( q_{\text{start}} \)
6. \( q_{\text{accept}} \)
7. \( q_{\text{reject}} \)

New Definitions:

- For TM \( M \): \( L(M) = \{ w | M \text{ accepts } w \} \)
- \( L \) is Turing-Recognizable if \( \exists \ TM \ M \) s.t. \( L = L(M) \)
- A TM \( M \) is a **decider** if it halts on every input. (i.e. \( \forall x \in \Sigma^* \) M either accepts \( x \) or rejects \( x \))
- \( L \) is **Decidable** if \( \exists \) a decider \( M \) s.t. \( L = L(M) \)
Variants on TMs

In the HW, we saw that the TM with left reset is equivalent to the canonical (i.e. standard) TM.

Another variant:

A **multi-tape TM** has \( k \) tapes and a head on each tape.

The transition function is:

\[
Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R\}^k
\]

- one symbol per tape
- one symbol per tape
- move left or right

A multi-tape TM \( M \) works as follows:

- At beginning, Tape 1 has input \( w \) followed by an infinite \( \infty \) of blanks. Tapes 2 to \( k \) are all blank.

\[
\begin{align*}
1 & : w_1 \ldots w_n \ldots \infty \\
2 & : \_ \ldots \_ \ldots \_ \\
\vdots & : \_ \ldots \_ \ldots \_ \\
k & : \_ \ldots \_ \ldots \_ \\
\end{align*}
\]

- At every step, follow \( s \)'s instruction
- If \( M \) reaches a accept halt & accept
- If \( M \) reaches a reject halt & reject
Thm 1: For every k-tape TM M, there exists an equivalent 1-tape TM M'.

\[ \{ w | A \text{ accepts } w \} = \{ w' | B \text{ accepts } w' \} \]

AND \[ \{ y | A \text{ rejects } y \} = \{ y' | B \text{ rejects } y' \} \]

(proof)

We show: given a multi-tape TM, we can simulate it with a single-tape TM (note that given a single-tape TM, we can easily simulate it on a multi-tape TM by only using the 1st tape).

Intuition:

\[ \text{# contents of tape 1} \quad \text{# non-blank contents of tape 2} \quad \text{#} \quad \ldots \quad \text{# contents of tape k} \quad \# \]

Note that even though each tape is infinite, the non-blank contents on a given tape is always finite.

to keep track of where each head was, mark tape at that place.

(i.e. if a head was reading symbol a in the multi-tape TM make the single tape TM mark that spot in the tape with a "#")

For each state \( q_i \) in the multi-tape TM you can replace it with a finite # of states like in the picture at left.
more formal proof:

Given M: multi-tape TM
construct M', single tape TM as follows:

1. On input w, put tape into correct format.

2. To simulate a single move of M do:
   a. Scan tape from # to the nearest head to find out what the head would be pointing at.
   b. Execute the tasks of the transition: scan from # to the end again each time:
      - overwrite a
      - if direction = L then go left & mark that symbol
      - if direction = R if next symbol is # add more space by shifting everything to the right.
      - and then coming back and marking L's else, just go right and mark that symbol.

since M' correctly simulates M,
if M accepts a word, M' will too.
if M enters reject state, M' will too.
A non-deterministic TM (NTM)

transition function changes:

\[ S : Q \times \Gamma \rightarrow S \times \Gamma \]

where \( S \subseteq Q \times \Gamma \times \{ L, R \} \)

i.e. for each situation there can be multiple valid transitions.

(same kind of generalization as the generalization of DFA to NFA)

An NTM N accepts w if \( \exists \) an accepting computation history.

But notice that an NTM could reach an accept state with one computation history and any reject state on another one, and maybe never halts on another history... so it is harder to define what it means to reject.

An NTM N rejects w if

1. it does not accept w
2. there is no infinite computation history of N on input w.

Thm 2: For every non-deterministic TM N, \( \exists \) an equivalent deterministic TM M.

Proof: Thanks to Thm 1 it is sufficient to give a 4 type TM. Input tape [w]

worktape

non-deterministic tape

are we done? [ ]

(this is really just a flag.)
intuition: do a depth first search of all computation branches.

problem: you could get stuck in an infinite loop

solution: first try all computation histories of length 1

then of length 2

etc.

this way if there is a finite computation history leading to accept (or reject) you will eventually find it.

**formal proof**

on input w:

1. Copy input to work tape

2. Use contents of non-det. tape to decide which transitions to follow as you simulate N on input w.

(suppose there are at most b possible transitions you could make at each step. & keep track of which transition you are trying on non-det tape. i.e. first write 1, then increment...)

If you have reached q accept, then accept

If you have reached q reject, go to step 3

If you have run out of instructions on non-det tape set "are we done" flag to "no" then goto step 3

3. If contents of the non-det tape is the last possible comp history of length k

   If "are we done" = "no" then set "are we done"
   to "don't know" and increment non-det tape, go to step 1

   Else reject

   Else goto step 1
Decidable Languages

\[ \text{ADFA} = \{<M, w> | M \text{ is a DFA, } w \text{ is a string, and } M \text{ accepts } w^3 \} \]

what does this notation mean?
\(<X>\) means "an encoding of \(X\)"
m will assume it's encoded into binary.

A turing machine \(T\) can decide \(\text{ADFA}\) as follows:
o on input \(<M, w>\):
simulates \(M\) on input \(w\)
if \(M\) ends in an accept state then accept.
if \(M\) ends in a reject state then reject.

why is \(T\) a decider?
if \(w \in L(M)\) you accept
if \(w \notin L(M)\) you always reject.
you never loop.

not decidable:
\[ \text{ATM} = \{<M, w> | M \text{ is a TM s.t. } w \in L(M) \} \]

consider TM \(T\) that simulates \(M\) on input \(w\).
\(T\) recognizes \(\text{ATM}\)
but does not decide.

in fact \(\text{ATM}\) is not decidable

we will prove this next class.