Lecture 26 COL 352 Mar 17

Multiplying two integers using a function COPY

$$0^m 1 0^n \overset{*}{\vdash} B^{m+n+1} 0^{mn}$$

Idea: Use COPY to copy 0^n m-times

$$0^m 1 0^n \vdash 0^m 1 0^n 1 \xrightarrow[\text{COPY}]{} B 0^{m-1} 1 0^n \underline{1} 0^n$$

$$B 0^{m-1} 1 0^n 1 0^n \xrightarrow[\text{COPY}]{} BB\, 0^{m-2} 1 0^n \underline{1} 0^{2n}$$

$\vdots$

COPY

$$q_0 0^m 1 0^n 1 0^i \vdash^* B 0^{m-1} 1 q_1 0^n 1 0^i$$

$i=0$ initially

q_1 to make an additional sequence of 0^n)

$$0^j 1 q_1 0^n 1 0^i \vdash^* 0^{j-1} 1 q_5 0^n 1 0^{i+n}$$

q_1 : Change a 0 of 0^j to B for countdown and enter state q_2

q_2 : Change a 0 in 0^n to 2 and copy 0 to right

q_3 : Move left until 2 ; change to q_2 and repeat

until $0^{j-1} 1 2^n 1 0^{i+n}$

q_4 : $0^{j-1} 1 2^n 1 0^{i+n} \vdash^* 0^{j-1} 1 0^n 1 0^{i+n}$

Copy function

	0	1	2	B
q_1	$q_2, 2, R$	$q_4, 1, L$		
q_2	$q_2, 0, R$	$q_2, 1, R$		$q_3, 0, L$
q_3	$q_3, 0, L$	$q_3, 1, L$	$q_1, 2, R$	
q_4		$q_5, 1, R$	$q_4, 0, L$	

$$\delta(q_0, 0) = (q_6, B, R)$$

$$\delta(q_6, 0) = (q_6, 0, R)$$

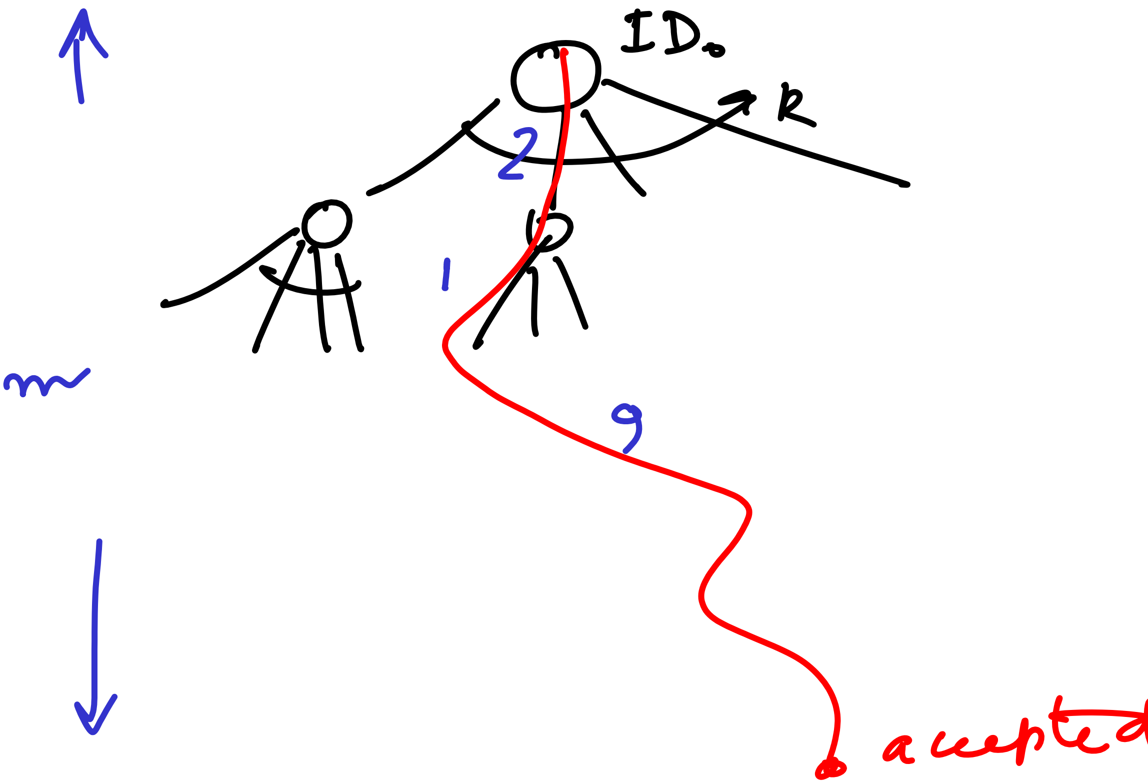
$$\delta(q_6, 1) = (q_1, 1, R)$$

function call

How about non-det TM?

Are these more powerful

$$\delta(q_i, a) \text{ contain } k \left\{ \begin{array}{l} (q_{i_1}, b, R) \\ (q_{i_2}, c, L) \\ (\quad) \end{array} \right.$$



How do we simulate using Multitape det. TM?

$s_1 \; s_2 \; s_3 \; s_4 \; \cdot \; \cdot \; s_m$ $\qquad s_i \in \{1, \dots 10\}$

Given a TM M,

$L(M) = \{ w \mid M \text{ accepts } w \}$

What happens if M doesn't accept w?

It may reach an undefined state

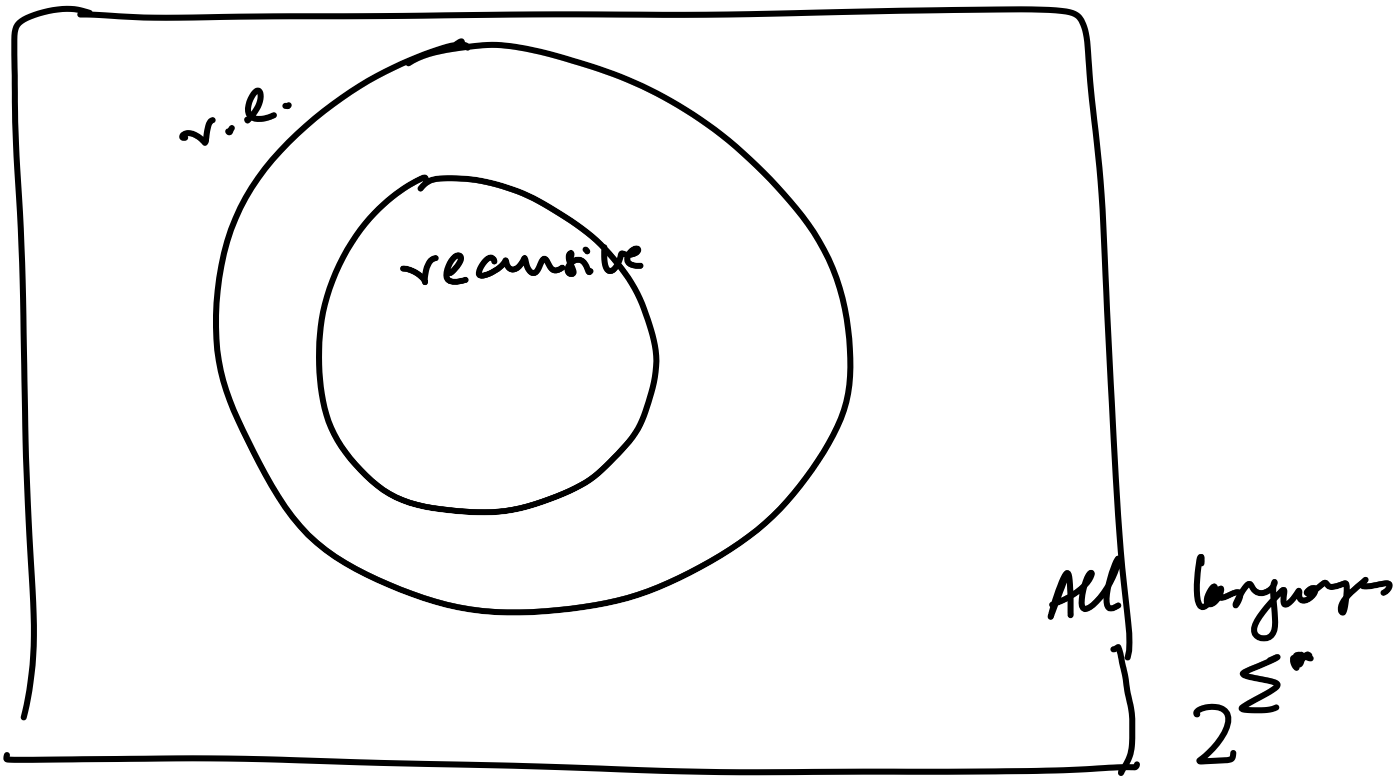
It goes on forever

ID repeating

Head keeps moving right

- The class of languages accepted by a TM is called
  <u>recursively enumerable</u> (r.e.)

- The class of languages for which we can design a TM that always stops
  <u>recursive</u> languages



An alternate model of TM

Special tape : output tape

Write a symbol and move right

output tape

read only tape input

A <u>generator</u> TM writes out all the strings of a given language on the output tape.

A generator for a language L is guaranteed to write out $w \in L$ on the output tape "eventually". A string $w \notin L$ will never be written.

<u>Claim</u>: There is a generator $G(L)$ for every r.e. language L

Canonical ordering: order according to lengths of strings

s_1 $\boxed{s_2}$ s_3 ... $\boxed{\ }$

Given a TM M for L.

for all strings: Run M on s_1 if it accepts write out s_1 on output tape

<u>Modification</u>: Run M for j steps on s_i $\forall j$

Generate all pairs (i, j)

Run M for j steps on string s_i

if accepted write it out on output tape

Note that any string $w \in L$ will correspond to some (i, j)

Claim: Given a generator $G(L)$ for a language L, then L is r.e. (i.e. accepted by some TM M)

Proof. Run the generator and accept w if w is written on the output tape.

Generator for recursive languages.

The strings of a recursive language L can be enumerated in the canonical ordering.

L : Canonical ordering

w_{i_1} w_{i_2} w_{i_3} $\ldots w_{i_k}$

$i_1 < i_2 < i_3 < .$

[Show equivalence between TM that always stops (recursive) and generators that print strings in canonical order