

$$m_A(n) = \begin{cases} 1 & \text{if } n \in A \\ 0 & \text{otherwise} \end{cases} \quad \Bigg\| \text{decidable.}$$

Semidecidable:

$$m_A(n) = \begin{cases} 1 & \text{if } n \in A \\ \emptyset & \text{if } n \notin A \end{cases}$$

↓  
(runs indefinitely)

- + If  $A$  is r.e.  $\rightarrow$  then  $A$  is  $\emptyset$  semi-decidable
- Have an algorithm that prints all elements in  $A$ .

$m_A(n)$  waits and checks every hour whether  $n$  was printed.

halts.  
 $\rightarrow$  true (1)  $\rightarrow n \in A$   
 $\rightarrow$  runs indefinitely  $\rightarrow m_A(n)$

- + If  $A$  is semi-decidable then  $A$  is r.e.

b 26, 09

Review:  $\mu$  recursive implementation on TM

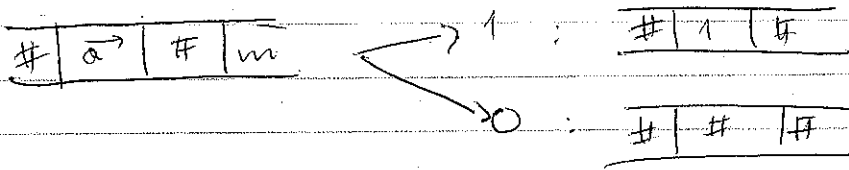
Big Pic:  $\mu$  recursive f.n.  
 $\downarrow$   
 TM

Proving that every  $\mu$ .rec. f.n. is Turing computable!

$0, \sigma, \pi$  by  $\textcircled{0}$   $\textcircled{PR}$   $\mu$  recursion

$\mu_m \times P(\vec{a}, m) = f(\vec{a})$   
 // while  $\rightarrow$  no condition to stop

We know  $P(a^{\rightarrow}, m)$  is Turing computable



We wanna produce a TM for computing.

Analyze the problem.

Idea: try  $m = 0$

if  $P(a^{\rightarrow}, 0)$  return 0;

else

try  $m = 1$

if  $P(a^{\rightarrow}, 1)$  return 1;

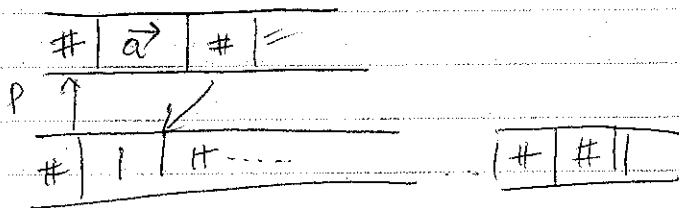
$m = 0$

while ( $\neg P(a^{\rightarrow}, m)$ )

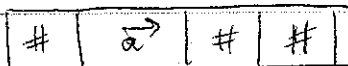
{

$m++$ ;

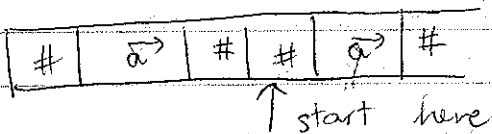
}



\* correct way:



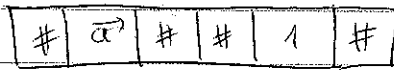
copy  $a^{\rightarrow}$



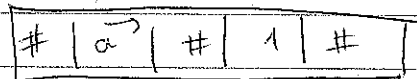
$p(a^2, 0)$   
is true

Result

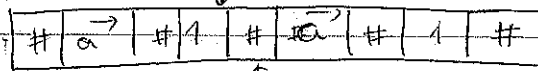
$p(a^2, 0)$   
is false



$m+1$

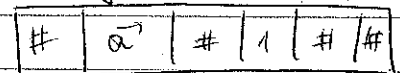
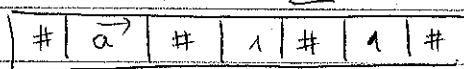


copy



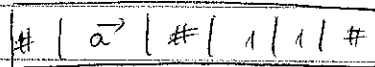
$p(a^2, 1)$   
is true

$p(a^2, 1)$  is false



return 1  
 $\pi_{k+1}^{k+2}$

$m+1$

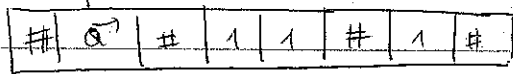


copy



$p(a^2, 2)$  is true

false



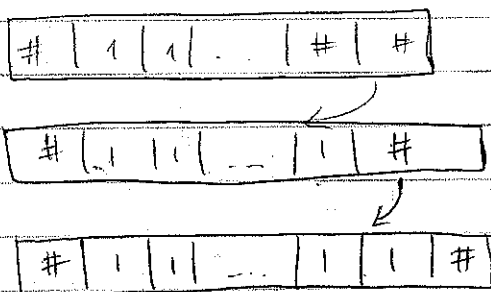
return 2 =  $\pi_{k+1}^{k+2}$

Big problem with what we had so far.

+ when we had a negative result, it was OK.

+ sometimes, actually produced algorithms, but many of these algorithms were unrealistically long.

Example:  $\sigma(n) = n + 1$ .



For security, we have integers: 160-200 digit long.

$10^{100}$  impossible # of steps.

→ we need to separate "theoretical" algorithms from practical algorithms.  
feasible.

+ How do you define an algorithm?

1930's  $\begin{matrix} \swarrow \text{TM} \\ \searrow \mu\text{-rec} \end{matrix}$

+ How do you define a feasible algorithm?

No perfect definition exists.

Existing  $\swarrow$  exhaustive search.  
sorting  $O(n \log n)$

A little bit of physics

$$1 \text{ year} = 365 \text{ days} = 3 \times 10^7 \frac{\text{seconds}}{\text{year}}$$

$$\begin{aligned} T &= 20 \text{ billion years} \\ &\approx 2 \times 10^{10} \text{ years} \\ &= 6 \times 10^{17} \text{ seconds} \end{aligned}$$

$$\Delta t = \frac{\text{dist}}{\text{velo.}}$$

T

$\Delta t \leftarrow$  smallest possible time interval

$t =$  time during which the light goes through the smallest elementary particle.

Heisenberg's uncertainty principle.

$$\Delta x \Delta p \geq \frac{h}{2\pi} \quad h = \text{planck's constant}$$

Q Next Thursday : Quiz / Test.