- once this is done: will have shown all of these notions of "computable function" are equivalent!

<u>Theorem</u> every TM computable function is MA computable.

Pf. - Sps M is a TM on $A = \{\sigma_1, \ldots, \sigma_k\}$ with $\sigma_0 = * =$ empty space symbol
- may assume M is a "table TM"
- states are $Q_0, Q_1, \ldots, Q_p$ (includes some stop states)
- gonna build an "equivalent" MA on $B = \{\sigma_0, \sigma_1, \ldots, \sigma_k, Q_0, \ldots, Q_p, Q_{p+1}, q_{p+2}, \#, \$\}$

- <u>idea</u>: current state of TM can be visualized as $\ldots * * x \underset{Q_t}{\underset{\triangle}{\sigma_j}} y * * \ldots$

where $x, y \in (A \cup \{*\})^*$ (i.e. x, y can include $*$'s)
- $\underset{Q_t}{\triangle}$ indicates r/w head in state Q_t (pointing at square w/ σ_j)

- in our MA alphabet, we'll ~~[illegible]~~ describe this state as

$$\# x Q_t \sigma_j y \#$$

- when Q_t a non-halting state our MA $\mathcal{A}$ will contain productions of the form

$$\# x Q_t \sigma_j y \# \to \# x' Q_t' \sigma_j' y' \#$$

where latter is the description of the TM after it is applied to first ~~[illegible]~~ description

(so x', y' will be one longer ~~[illegible]~~ or shorter than x, y depending on if we "moved" left or right)

- deal w/ halting states separately
- also will include an initial production that transforms a non-empty word w into $\# Q_0 w \#$

↪ More precisely:

Sps Q_i a non-halting state

1) if $(Q_i, \sigma_j, \sigma_n, 1, Q_m)$ in table for our TM, include all productions of the form

$$Q_i \sigma_j \sigma_e \to \sigma_n Q_m \sigma_e \quad (\text{for } 0 \leq e \leq K)$$

as well as the prod'n

$$Q_i \sigma_j \# \to \sigma_n Q_m \sigma_0 \#$$

*

2) if $(Q_i, \sigma_j, \sigma_n, -1, Q_m)$ in table include all

$$\sigma_e Q_i \sigma_j \to Q_m \sigma_e \sigma_n \quad 0 \leq e \leq K$$

as well as

$\# Q_i \sigma_j \to \# Q_m \sigma_0 \sigma_n$

3) if $(Q_i, \sigma_j, \sigma_n, C, Q_m)$ include

$Q_i \sigma_j \to Q_m \sigma_n$

Call collection of these productions P_i (corresponding to state Q_i)

— now, for Q_i a halting state, include the prod'n

$Q_i \to Q_{p+1}$ (call this P_i too)

— also include all prod'ns

$\sigma_j Q_{p+1} \to Q_{p+1} \sigma_j \qquad 0 \le j \le k$

also

$\# Q_{p+1} \to Q_{p+2}$

halting

... so after we reach $\# x Q_i \sigma_j y \#$
these take us to $Q_{p+2} x \sigma_j y \#$

post halt

— also include prod'ns

$Q_{p+2} \sigma_j \to \sigma_j Q_{p+2} \qquad 0 \le j \le k$

$Q_{p+2} \# \to_t 0$

... these take us from $Q_{p+2} x \sigma_j y \#$ to
$x \sigma_j y$ (terminally)

Call these post-halting productions R

- Finally we need "prep" prod'ns to take a nonempty input ω to $\# Q_0 \omega \#$
- as follows:

- include all prod'ns of form

$$\$ \sigma_i \to \sigma_i \$ \qquad 0 \le i \le k$$

also

$$\$ \to \#$$

Call these prod'ns S

- lastly include

$$\emptyset \to \# Q_0 \$ \qquad \text{call this } (+)$$

↳ Let our MA A be these prod'ns in the following order

$$S, P_0, P_1, \ldots, P_p, R, (+)$$

<u>Observe</u>: on nonempty word ω:

- (+) takes to $\# Q_0 \$ \omega$
- S takes to $\# Q_0 \omega \#$
- Now non-halting P_i blocks apply
- correctly simulate TM until we reach $\# x Q_i \sigma_j y \#$ w/ Q_i halting
- then $P_i + R$ takes us to $x \sigma_j y$ ← output of TM ✓

So: to close implication loop, remains to prove

Theorem every MA-computable function is recursive.

Pf. – sps $f:(A^*)^n \to A^*$ is MA-comp'le.

– i.e. there is MA $\mathcal{A}$ and $B \supseteq A \cup \{,\}$ s.t. $\varphi^n_{\mathcal{A}} \upharpoonright (A^*)^n = f$

(recall: this means on input $x_1, x_2, \ldots, x_n$ w/ commas included, $\mathcal{A}$ terminates on $f(x_1, \ldots, x_n)$)

WTS f is recursive.

– Consider the following functions

1.) $I_n(x_1, \ldots, x_n) = x_1, x_2, \ldots, x_n$ (w/ commas)
(i.e. I_n takes $\vec{x}$ and outputs as a word w/ commas)

– then I_n is p.r. since $I_n(x_1, \ldots, x_n)$
$= con_{2n-1}(x_1, ,, x_2, ,, \ldots, ,, x_n)$
which is pr (why?)

2.) $A^*(x,y)$ = unique word z that $\mathcal{A}$ produces on input x in exactly $|y|$ steps

... where we treat all prod'ns as non-terminal and if no prod'ns we repeat output

↳ so $A^*(x,y)$ is defined $\forall x,y \in A^*$

3.) $STOP_A(x,y)$ iff A terminates on input x in $\leq |y|$ steps (i.e. the indicator function for this rel'n)

4.) $G(x) = x'$ ... where $x' = x$ w/ all commas deleted

↳ is pr since

$$G(0) = 0$$
$$G(x\sigma_j) = G(x)\sigma_j \quad \text{if } \sigma_j \neq , \text{ comma}$$
$$G(x,) = G(x)$$

- We'll use these to define F recursively
- WTS 2.) + 3.) are p.r. too

<u>Claim</u> A^* is p.r.

<u>Pf</u> (sketch) – Sps A is $P_1 \to Q_1, \ldots, P_m \to Q_m$ (some of these may be terminal)

- Let $REP(p,q,z)$ = word obtained by replacing first instance of p in z by q (or returns z if p does not appear)
- REP intuitively pr: does <u>bdd</u> search (length $|z|$) for p, then

replaces by q (p.r. operation)
↳ we take for granted is p.r.

– then: $A^*(x,0) = x$
and

$$A^*(x, y\sigma_j) = \begin{cases} REP(p_1, q_1, A^*(x,y)) & \text{if } p_1 \text{ first } p_i \text{ in } A^*(x,y) \\ REP(p_2, q_2, A^*(x,y)) & \text{if } p_2 \text{ " "} \\ \vdots & \\ REP(p_m, q_m, A^*(x,y)) & \text{if } p_m \text{ " "} \\ A^*(x,y) & \text{if no } p_i \text{ appears in } A^*(x,y) \end{cases}$$

– this is piecewise pr (conditions intuitively pr since involve bdd search in $A^*(x,y)$ for $p_1, \dots, p_m$)
– hence pr ✓

<u>Claim</u> $STOP_A$ is p.r.
<u>Pf</u> (sketch) – first consider the relation
$R_A(x)$ iff there is i s.t. p_i is a subword of x, no previous p_k is, and $p_i \to_t q_i$ is terminal

→ p.r. since bounded search for terminal p_i's.

— now observe

$STOP_A(x, \epsilon)$ iff none of p_k's in x

$STOP_A(x, y\sigma_j)$ iff $STOP_A(x,y)$

or $(\neg STOP_A(x,y) \wedge R_A(A^*(x,y)))$

or $(\neg STOP_A(x,y) \wedge$ no p_k appears in $A^*(x,y))$

everything pr in this pr. def^n

$\Rightarrow STOP_A$ is p.r. ✓

Finally, observe that:

for every $(x_1, \ldots, x_n) \in (A^*)^n$

$f(x_1, \ldots, x_n) = \varphi_A^n(x_1, \ldots, x_n)$

$$= G(A^*(I_n(x_1, \ldots, x_n), \mu_1 y[STOP_A(I_n(x_1, \ldots, x_n), y)]))$$

if we let $x = I_n(x_1, \ldots, x_n)$

this looks for shortest y of form $\sigma_1 \sigma_1 \ldots \sigma_1$ (only length is relevant)

s.t. A terminates on input x in exactly $|y|$ steps

and in this case returns output of A on x (w/ commas removed)

i.e. returns $f(x_1, \ldots, x_n)$ ✓

is clearly recursive since one μ_1-operator + p.r stuff ✓