- Sps $P(f)$ is a "property" of recursive functions
  ^ i.e. relation
- e.g. could have $P(f)$ iff $f(0)\downarrow$
  or $P(f)$ iff f is total

- when is there an algorithm to decide, for a given f, whether $P(f)$ or $\neg P(f)$?
  i.e. when is P "recursive?"
- work w/ codes to make this question precise

<u>Def'n</u> The functional relation $P(f)$ is <u>recursive</u> if P^* is recursive, where P^* defined by

$$P^*(e) \text{ iff } P(\varphi_e)$$

<u>Theorem</u> (Rice's Theorem) If $P(f)$ is a property of (unary) recursive functions then P is recursive iff P is trivial, i.e. iff <u>either</u>

very strong undecidability result!!

$P(f)$ true for all rec. f
<u>or</u> $P(f)$ false for all rec. f.

↳ Theorem says there are no nontrivial recursive properties of the recursive functions.

PF: - Sps toward contradiction P is recursive and nontrivial

- wlog may assume $\emptyset \notin P$ and $f \in P$ for some r.c.c. $f \neq \emptyset$

(one of $P, \neg P$ contains $\emptyset$ and the other some $f \neq \emptyset$; exchange P w/ $\neg P$ if necessary — also recursive)

by non-triviality

- let $K \subseteq \mathbb{N}$ be r.e. but not recursive
- we find a total recursive $F: \mathbb{N} \to \mathbb{N}$ s.t.

$$e \in K \text{ iff } \varphi_{F(e)} \in P$$

(i.e. iff $F(e) \in P^*$)

- gives contradiction: F gives a recursive way of deciding membership in K
- namely can define $\chi_K(e) = \chi_{P^*}(F(e))$

F total recursive, recursive $\Rightarrow$ recursive

- more specifically we arrange

$$e \in K \Rightarrow \varphi_{F(e)} = f$$

$$e \notin K \Rightarrow \varphi_{F(e)} = \emptyset$$

"use f as an indicator"

i.e. $\varphi_{F(e)}(x) = \begin{cases} f(x) & \text{if } e \in K \\ \text{undef} & \text{o/w} \end{cases}$

define

$$g(e,x) = f(x) \quad \text{if } e \in K$$
$$= \text{undef} \quad \text{if } e \notin K$$

then g is recursive

(<u>why</u>: graph rel'n: $g(e,x) = y$

$\iff \underbrace{f(x) = y \wedge e \in K}_{\text{r.e. since } K \text{ r.e.}}$

$\Rightarrow$ graph(g) r.e.

$\Rightarrow g$ recursive)

- hence g has a code e_0
- so $g(e,x) = \varphi(e_0, e, x)$

$= \varphi(S(e_0, e), x)$

$= \varphi_{S(e_0,e)}(x)$

- so let $F(e) = S(e_0, e)$ ← total recursive

then $= \varphi_{F(e)}(x) = f(x)$ if $e \in K$

undef if $e \notin K$

$\Rightarrow \varphi_{F(e)} \in P$ iff $e \in K$ ✓

<u>Ex</u>: For f recursive, define $P(f)$ iff $f(0)\uparrow$. Then P is not a recursive property (i.e. $P^*(e)$ iff $P(\varphi_e)$ is not recursive)

Pf: immediate from Rice: P not trivial (some rec f's halt on e, others don't)

↳ giving another pf of last problem of HW 6

<u>Def'n</u> A property $P(f)$ of the recursive functions is <u>recursively enumerable</u> (r.e.) if P^* is r.e.

(... where $P^*(e)$ iff $P(\varphi_e)$)

<u>Ex</u> as above, define $P(f)$ iff $f(a)\downarrow$

Then P is r.e.

Pf: you showed on HW 6 (P^* was called H there)

<u>Rice - Shapiro Theorem</u>

- next going to characterize r.e. properties $P(f)$
- unlike rec properties, there are nontriv
- characterization uses finite functions

- for $x \in \mathbb{N}$, define π_x ("finite function coded by x") as follows:

- if $Seq(x) \wedge lh(x) = n \wedge x = \langle t_1, \dots, t_n \rangle$
  $\wedge (i<j \Rightarrow (t_i)_0 < (t_j)_0)$
  then
  $dom(\pi_x) = \{(t_1)_0, (t_2)_0, \dots, (t_n)_0\}$
  and $\pi_x((t_i)_0) = (t_i)_1$

- else $\pi_x = \emptyset$

- so e.g. if $x = \langle\langle 0,5\rangle, \langle 1,2\rangle, \langle 7,4\rangle\rangle$
  then $\pi_x(0) = 5$ $\pi_x(1) = 2$ $\pi_x(7) = 4$
  undef elsewhere

- this coding is "clearly" p.r. in
  sense that rel'n
  $D(x,y)$ iff $y \in dom(\pi_x)$ is (clearly) p.r.

- also function
  $\pi(x,y) = \pi_x(y)$ if $y \in dom(\pi_x)$
  $= 0$ else
  is "clearly" p.r.
  (... since piecewise p.r.
  $\pi_x(y) = \mu z \le x\, (\exists i \le x\, [((x)_i)_0 = y \wedge ((x)_i)_1 = z])$

- notice: if $y \in dom(\pi_x)$ then $y, \pi_x(y) < x$

Theorem (Rice – Shapiro)

Sps $P(f)$ is a property of rec. functions.
TFAE:

① $P(f)$ is r.c.

② $F_p := \{x : P(\pi_x)\}$ is r.c. (i.e. finite f's in P form r.c. set)

and

$f \in P$ iff $\exists \pi \in F_p$ s.t. $\pi \subseteq f$
(i.e. f extends a finite $\pi \in P$)

③ There is an r.c. set F of finite functions s.t. $f \in P$ iff $\exists \pi \in F$ s.t. $\pi \subseteq f$

(i.e. functions in P are exactly the functions extending the ~~[illegible]~~ functions from some r.c. set of fin. funcs)

Pf: ② ⇒ ③ : trivial: let $F = F_p$. ✓

③ ⇒ ① : WTS P^* is r.c.

– have $P^*(e)$ iff $P(\varphi_e)$ defn

iff $\exists \pi \in F \ (\pi \subseteq \varphi_e)$ by ③

iff $\exists x \ (\pi_x \in F \wedge \pi_x \subseteq \varphi_e)$

$$\underbrace{\text{iff } \exists x \left[\pi_x \in F \wedge \forall y < x \left(y \in dom(\pi_x) \Rightarrow \exists z < x (\pi_x(y) = z \wedge \varphi_e(y) = z)\right)\right]}$$

of form $\exists x$ (r.c.)

hence r.c. ✓

almost $\exists x$(pr.) r.c. part is $\varphi_e(y) = z$

1 ⇒ 2 : this is hard one

- first check F_P is r.e.
- $\forall x, y$ we have $\Pi_x(y) = \Pi(x,y) = \varphi(e_0, x, y)$
  where e_0 = code for $\Pi(x,y)$

$\Rightarrow \Pi_x(y) = \varphi(S(e_0, x), y) = \varphi_{S(e_0,x)}(y)$
for all x, y

$\Rightarrow \Pi_x = \varphi_{S(e_0,x)}$

$\Rightarrow$ so $\Pi_x \in F_P$ iff $\varphi_{S(e_0,x)} \in F_P$ iff
$P^*(S(e_0,x))$
r.e. since P^* r.e. $S(e_0,x)$ total r.e. ✓

Now: WTS
b/c finite $\Pi \in P$

$f \in P$ iff $\exists \Pi \in F_P$ s.t. $\Pi \subseteq f$ (**)

Do this w/ 2 lemmas:

Lemma 1 if $f \in P$ there is a finite $\Pi \subseteq f$ s.t. $\Pi \in P$

PF: - Sps not, i.e. $\exists f \in P$ s.t. $\forall$ finite subfunctions $\Pi \subseteq f$, $\Pi \notin P$

- fix $K \subseteq \mathbb{N}$ r.e. but not recursive.
- let h be rec function w/ $dom(h) = K$; so $h(n)\downarrow$ iff $n \in K$
- let e_0 be code for h, $\varphi_{e_0} = h$.

- we find tot. rec. $G(x)$ s.t.
  $x \in K$ iff $\varphi_{G(x)} \notin P$
  i.e. iff $\neg P^*(G(x))$

- gives contradiction
  - K r.e. by choice of K
  - but $\neg K(x)$ iff $P^*(G(x))$ ← r.e. too since P^* r.e., G tot. rec.
  - hence $\neg K$ r.e. too
    - $\Rightarrow K$ recursive, but it isn't.

- How to get G?
  - idea is: $x \in K \Rightarrow \varphi_{G(x)}$ a finite subfunction of f
  - $x \notin K \Rightarrow \varphi_{G(x)} = f$
- How to do this?

- Define $STOP(e, x, t)$ iff e is code for a TM M that on input x terminates in $\le t$ steps
- $STOP$ "clearly" recursive (actually p.r.)

- define $g(x,t) = \begin{cases} f(t) & \text{if } \neg STOP(e_0, x, t) \\ \text{undef} & \text{o/w} \end{cases}$

  r.e. rel'n

  i.e. if $h(x)\uparrow$ or computation for $h(x)$ takes $> t$ steps

  K function from above

- $g(t)$ checks if we can determine $x \in K$ in $\le t$ steps
- if not, output $f(t)$
  if yes, undef

- g is recursive
- hence has code e^*

  so $g(x,t) = \varphi(e^*, x, t) = \varphi(S(e^*, x), t)$

- let $G(x) = S(e^*, x)$

  $=$ code for function which on input t, outputs $f(t)$ if we can't determine $x \in K$ in $\leq t$ steps

- this works:
  - if $x \in K$, then $h(x)\downarrow$ i.e. machine M coded by e_0 halts on input x in some number t_0 steps

    then $\varphi_{G(x)}(t) = g(x,t)$
    $= \uparrow$ for $t \geq t_0$
    $= f(t)$ $t < t_0$

    so $\varphi_{G(x)}$ fn subfunction of f if $x \in K$

  - if $x \notin K$, $h(x)\uparrow$ always so $\varphi_{G(x)}(t) = g(x,t) = f(t)$ ✓

Note: Lemma 1 shows $\Rightarrow$ direction of (**)

<u>Lemma 2</u> if $f \in P$ and $g \supseteq f$ then $g \in P$ ("re properties are monotone upward")

<u>Pf</u> Sps not, ie $\exists f \in P$ and $g \supseteq f$ st $g \notin P$.

- we find total rec H st

$$x \in K \Rightarrow \varphi_{H(x)} = g$$
$$x \notin K \Rightarrow \varphi_{H(x)} = f$$

which gives $x \in K$ iff $\neg P^*(H(x))$, contradiction as before.

- define $h(x,y) = \begin{cases} g(y) & \text{if } x \in K \\ f(y) & \text{if } x \notin K \end{cases}$
- claim h is recursive

graph rel'n is

$$h(x,y) = z \text{ iff } [x \in K \wedge g(y) = z] \vee [x \notin K \wedge f(y) = z]$$

- everything r.e. except "$x \notin K$" which is co-r.e. ... uh oh!
- but we're saved since $g \supseteq f$, so can rewrite as

$$[x \in K \wedge g(y) = z] \vee f(y) = z$$

which is r.e. hence h is recursive as claimed.

- let e_1 be code for h
- define $H(x) = S(e_1, x)$, works just like before ✓

<u>Note</u>: Lemma 2 gives $\Leftarrow$ direction of ** since if $\pi \in P$ finite and $f \supseteq \pi$ then must have $f \in P$. So this finishes pf of Rice-Shapiro ✓