AKHIL 18CS10070, THOC - Assignment-1

(1) Let P be any non trivial property on partial recursive functions

$\exists$ f_u & f_v partial recursive functions

s.t. $P(f_u) = T$ and $P(f_v) = \perp$

Now, let's define $\sigma(x)$ such that

$$\sigma(x) = \begin{cases} f_v(x) & \text{if } P(\sigma) = T \\ f_u(x) & \text{o/w} \end{cases}$$

So, $\exists$ fixed point x_0 $(x_0 = \sigma(x_0))$ because σ is a total recursive function.

NOTES

Now, if $P(\sigma) = T$ then $\sigma(x) = f_v(x) = \perp$

which is a contradiction

similarly, if $P(\sigma) = \perp$ then $\sigma(x) = f_u(x) = T$
which is also a contradiction.

So, non-trivial property of set of partial recursive functions is undecidable.

(21) We will show a reduction from Halting Problem

let (M, x) is an instance of HP

$\therefore \quad (M, x) \longmapsto N$

let's construct N with input alphabet $\Sigma = \{0, 1\}$

We can encode M & x in binary and input (M, x) to N

Now,
for input x' to N, run M on x and use other tape alphabets for doing the computation

if M halts on x, write 1 on the tape and halt
$\therefore N \in L$

else 1 is not written on the tape
$\therefore N \notin L$

So, if L is decidable it means HP is decidable ~~which is a con~~
and we know HP is undecidable
Hence,
L is non decidable.

S	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

AUG 18

3 (a) let $\neg(a)$: P is not a monotone

$\therefore$ T_p is not r.e. by direct application of rice's 2nd theorem.

So T_p is r.e. $\Rightarrow$ (a).

(b) $\neg(b)$: A is a finite langugein S_p and there's no finite subset of A in S_p

- Assume L_p was r.e.

- let M_1 accepts L

Now construct a reduction : $\langle M, w \rangle \Rightarrow M'$ s.t
- M' accepts L if w is not in $L(M)$ and accepts else accepts finite subset of L

Now, on input M_1, M' with input x will simute M on w for $|x|$ moves and if it fails to accept w after $|x|$ moves, M' accepts x.

- if M accepts w say after k moves than we know $L(M') = \{ x \mid x \in L \ \& \ |x| < k \} \subseteq L$

$\therefore$ if M doesn't accept w, $L(M')$ bung a finite subset of L is not in S_p | using $\neg(b)$

NOTES : If T_p in r.e then membership problem is decidable hence T_p is not r.e.

(C) Finite languages in S_p can be represented in binary encoding

Now, as T_p is r.e.

Let an enumerable machine N enumerate T_p.
We will make $M^{(i)}$ to enumerate binary encodings of finite language ~~xx~~ 'i') in S_p.

Then, we will enumerate the binary pairs (i, j) using K, i is a binary encoding of finite language

Now, if N has printed $M^{(i)}$ in j steps, K prints 'i' followed by a delimiter symbol to differentiate from previous and next step encodings.

So, K enumerates all binary codes for finite set in S_p.
$\therefore$ T_p in r.e $\Rightarrow$ (C)

So, we come to conclude that (a), (b) and (c) imply T_p is r.e.

NOTES

(4) let $A = \{ w_1, w_2 \ldots w_k \}$

$B = \{ x_1, x_2 \ldots x_k \}$

Now, $\forall\, w_i \in A$ and $\forall\, x_i \in B$

$\{ w_i, x_i \in \Sigma^* \}$ $\qquad |x_i| = |w_i| = 5$.

• Let C be a variant of PCP as PCP5

$C = (A, B) \in PCP5 \iff \exists$ sequence $i_1 \ldots i_k$
$w_{i_1} \cdots w_{i_k} = x_{i_1} \cdots x_{i_k}$

because all the strings are of length 5, so if a sequence of $i_1, \ldots i_k$ exists that satisfies PCP5, then
$w_{i_1} = x_{i_1}$, $w_{i_2} = x_{i_2} \ldots \Rightarrow w_{i_k} = x_{i_k}$

because all the corresponding strings need to match, it can be done in $O(n+m)$

Now, iterate from $i = 1$ to $\min(n, m)$ & check $w_i == x_i$

if it true atleast once, it can be said that $(A, B) \in PCP5$, otherwise not.

NOTES

Hence, this problem is decidable.

5. We know, deciding whether two grammers generate same languge or not, is undecidable.

i.e. $S = \{ (G_1, G_2) \mid G_1 = G_2 \}$ is undecidable

Now, reducing from S to $P = \{ G \mid L(G) = L(G)^R \}$

for (G_1, G_2) of S, let's construct $L(G)$ s.t $L(G) = \$ L(G_1) \cup L(G_2)^R \$$

So, $L(G) = L(G)^R$ iff $L(G_1) = L(G_2)$

Hence this reduction is invalid
so P is undecidable.

6. we know if set accepted by M is finite then $VALCOMPS_M^t$ is also finite and be represented by a CFG is a CFL.

Now, if M accepts an infinite set, there exists a valid computation history

$$w_1 \# w_2^R \# w_3 \ldots\ldots$$

02 SUNDAY

as w_i's are IDs and $|w_2|$ is greater than pumping lemma's constant, so we can mark

NOTES

the symbols of w_2 as distinguished.

hence we can pump w_2 without pumping w_1 and w_3, thus getting an invalid computation that must be in L,

So we conclude that $VALCOMPS_M^t$ is not CFL
$\therefore$ $VALCOMPS_M^t$ is CFL $\Leftrightarrow$ $L(M)$ is finite.

(8) $\phi = Q_1 x_1 Q_2 x_2 \cdots Q_e x_e \psi$ where $Q \in \{\exists, \forall\}$ and ψ is quantifier free

let $\phi_i = Q_{i+1} x_{i+1} Q_{i+2} x_{i+2} \cdots Q_e x_e \psi$ for i o to e
$\therefore$ $\phi_0 = \phi$ and $\phi_e = \psi$

Now, formula ϕ_i has i free variables. For $a_1 \cdots a_i \in N$ we write $\phi_i [a_1 \ldots a_i]$ to be the sentences obtained by substituting the constants $a_1 \ldots a_i$ for the variables $x_1 \ldots x_i$ in ϕ_i.

this algo builds finite automata A_i which recognizes the collection of strings representing i tuples of numbers that make ϕ_i true.

It first builds A_e directly then for each $i = e$ ... 1 it uses A_i to build A_{i-1}
Once, this algo has A_0 it tests whether accept ϵ in which case it shows that ϕ is true.

Building machine A_i, can be seperated in 2 cases.

Case 1: A_e : since $\phi_i = \psi$ is a boolean combination of additions and we can build a finite automaton to compute single additions and regular languages are closed under union, intersection and complement so we can build A_e.

Case 2: A_i from A_{i+1}

if $\phi_i = \exists\, x_{x+1}\ \phi_{i+1}$: A_i operates as A_{i+1} but it non deterministically guesses the value of a_{i+1} such that A_i accepts the input $(a_1 \ldots a_i)$ if some a_{i+1} exists such that A_{i+1} accepts $a_1 \ldots a_{i+1}$

if $\phi_i = \forall\, x_{x+1}\ \phi_{i+1}$: it is equivalent to $\neg \exists\, x_{i+1} \cdot \neg \phi_{i+1}$. So, we can build a finite automaton that recognizes the complement of A_{i+1} then apply the preceding construction to the $\exists$ quantifier and then account for the complement again to obtain A_i

$\Rightarrow$ A_0 accepts any input iff ϕ_0 is true so if A_0 accepts ϵ, ϕ is true and A_0 accepts otherwise rejects.

NOTES Hence $Th^{+}(\mathbb{N})$ is decidable.

(7) Let $g: \mathbb{N} \to \mathbb{N}$ is a partial recursive fn, that works as follows:

g on input x :-

→ Finds TM M whose encoding is x.
→ Returns the encoding of another TM N that has M hardcoded in it.

N on input y :-

→ Define a lexicographic order $\leq$ in strings of Σ^*
→ Simulate M on x $\forall x \leq y$ one after other in order.
→ Accept y if M accepts y, after M halts on all strings $< y$.

Now, M is total TM, $L(M) = L(N)$ $\therefore$ $L(N)$ is recursive

If M is not total, $\exists\, y$ st. M doesn't halt on y'.

NOTES $\therefore$ $L(N) = \{ y' \mid y' < y \}$

so $L(N)$ is finite $\Rightarrow$ $L(N)$ is recursive.

$\therefore L(g(x))$ is recursive $\forall x \in \mathbb{N}$

we know, g outputs the description of a TM, hence g is total recursive.

Now, let's define an enumeration machine E, s.t E iterates on $\{0, 1, 2, 3 \ldots\}$ and generates $\{g(0), g(1), g(2) \ldots\}$ also, each step of computing $g(i)$ finishes because g is total.

$L(E) = \{ g(i) \mid i \in \mathbb{N} \}$ and $L(E)$ is r.e.

$\forall i$, $L(g(i))$ is recursive.

So, $L(E)$ is an r.e. list of TMs that accept recursive sets. $L(E)$ contains all possible TMs that accept an r.e. set

$\therefore$ given an re set A, $\exists M \in L(E)$ s.t. M accepts A

So $L(E)$ is the required list.