CS 208

HW4 - Q1

SAKSHAM RATHI

22B1003

1) (a) $L_1 = \{n \in \mathbb{N} \mid \exists m \in \mathbb{N}$ s.t. M_n halts on $w_m\}$

So, L_1 is basically the set of all turing machines M_n which halt on atleast one input string ($H(M) \neq \phi$)

Claim : L_1 is undecidable

Proof : It is known that the Halting Problem is undecidable. So, if we reduce halting problem to L_1, then we can show that L_1 is undecidable.

(Halting problem = set of pairs (M, w) such that w is in $H(M)$ i.e. M halts on w.)

We will describe an algorithm that transforms (M, w) into an output M', the code for another turing machine, such that w is in $H(M)$ iff $H(M') \neq \phi$.

We can make M' ignore its input and instead simulate M on input w. If M halts, then M' accepts its own input.

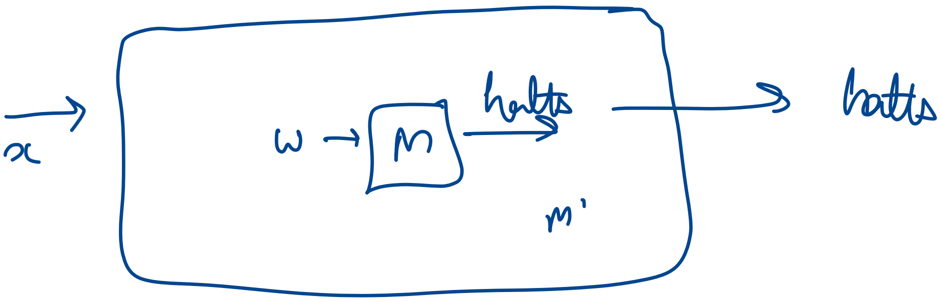
As we can see, if M does not halt on w then M' accepts none of its inputs i.e. $H(M') = \phi$. However, if M halts on w then M' accepts every input and thus $H(M') \neq \phi$

{By ignoring its input x, we mean M' replaces (x) by (M, w), this can be accomplished by some extra q_n states where n = length of the pair (M, w)}

Now using these additional states, M' simulates the turing machine for the halting problem.

Therefore, we have reduced the halting problem to L_1

Now, since halting problem is not recursive, and L_1 is as hard as halting problem, we can deduce that L_1 is undecidable.



<u>Proof of halting problem being undecidable:</u>

Let us assume that we have a turing machine which moves to an accepting state if M halts on w. Call this machine X.

$X(M, w)$

- Yes: if M halts on w
- No: if M does not halt on w

Consider Y such that:

$Y(M)$

- if $X(M, M) = $ yes $\rightarrow$ loop forever
- if $X(M, M) = $ No $\rightarrow$ halt

$Y(Y)$

$X(Y, Y) = \text{Yes}$ → Loop forever

$X(Y, Y) = \text{No}$ → Halt

It will never halt because of the above condition

Therefore, the halting problem is undecidable.

* Another smaller proof for this will be using Rice theorem.

We consider the property that $L(M) \neq \phi$. (Clearly a non-trivial property, because we can create machines that never halt and also machines which always halt.)

Now, using Rice theorem, we can deduce that this property is undecidable so is our language L_1.

CS 208

HW4 - Q2

SAKSHAM RATHI

22B1003

2) Given two lists of strings $u_1, u_2 \ldots u_n$ and $v_1, v_2 \ldots v_n$ over an alphabet Σ, does there exist a sequence of indices $i_1, i_2 \ldots i_k$ such that $u_{i_1} u_{i_2} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}$ (This is the PCP problem)

We will take this instance of PCP and try to construct a grammar G out of this. Now, the PCP will have a solution iff w and w^R are in $L(G)$.

start symbol $= S$

$$S \rightarrow A \mid B$$

$$A \rightarrow u_1 A a_1 \mid u_2 A a_2 \mid \ldots \mid u_n A a_n \mid \varepsilon$$

$$B \rightarrow a_1 B v_1^R \mid a_2 B v_2^R \mid \ldots \mid a_n B v_n^R \mid \varepsilon$$

where $a_1 \ldots a_n$ are extra symbols (different from the elements of Σ) (single letter symbols and distinct from each other, as long as n is finite, we can find such a_i's)

A will have strings of the form:

$$u_{i_1} u_{i_2} \ldots u_{i_k} a_{i_k} a_{i_{k-1}} \ldots a_1 \doteq w_A$$

B will have strings of the form:

$$a_i a_2 \ldots a_{i_k} v_{i_k}^R v_{i_{k-1}}^R \ldots v_1^R = w_B$$

Consider

$$w_B^R = v_{i_1} v_{i_2} \dots v_{i_k} a_{i_k} \dots a_{i_1}$$

Now if w_B^R has to belong to $G(S)$ then it should be part of $G(A)$.

[Because B has a_i in the ~~start~~ and a_is are different from Σ alphabet letters.]

$$w_B^R = w_A' \quad \text{for some } w_A' \in G(A)$$

$$v_{i_1} v_{i_2} \dots v_{i_k} \underbrace{a_{i_k} \dots a_{i_1}} = v_{j_1} \dots v_{j_l} \underbrace{a_{j_l} \dots a_{j_1}}$$

These parts must match because they are different from Σ.

$\Rightarrow \quad k = l$

and $i_t = j_t \quad \forall_{1 \le t \le k}$ Because a_i distinct from a_j $i \neq j$.

From this we get that:

$$v_{i_1} \dots v_{i_k} = v_{i_1} \dots v_{i_k} \quad \text{for some set } (i_1 \dots i_k)$$

Similarly, we can take w_A^R and prove it to be equal to w_B

Therefore, we have deduced the following:

If PCP has a solution $i_1 \dots i_k$ then we can find w and w^R belonging to G.

Similarly, if we can find w and $w^R \in G$, we can have a solution to the PCP problem instance.

Since we have proved that PCP reduces to our grammar G, proving the existence of a terminal string $w \in L(G)$ such that $w^R \in L(G)$ is undecidable.

* One might think that for the set $i_1 \dots i_k$, i_{l_1} can be equal to i_{l_2} for $l_1 \neq l_2$. But it can be shown that for such cases we can remove all the repetitions and our solution will still be valid.

CS 208

HW4 - Q3

SAKSHAM RATHI

22B1003

3b(b) Consider $L = \{ i \mid M_i$ does not halt on any input $\}$

L = set of all turing machines which do not halt at all
(for any inputs, $H(M) = \phi$)

$L_i = L \setminus \{ M_i \}$

M_i = turing machine which moves right by i steps and then moves left indefinitely (on seeing any input on tape)
(does not halt on any input)

Infinite union of $L_i = L$

$\bar{L}$ = complement of L
$= \{ i \mid M_i$ halts on atleast one input $\}$

This string is similar to Q1a of HW4 (Already proved that it is not recursive)

But $\bar{L}$ is RE.
This can be proved by an enumerating turing machine.
(seen in lecture).

We will take a turing machine M as input and iterate over two parameters:

(i) string w

(ii) number of steps to execute.

If M has some string w on which it halts, then it will do so after some n steps and we will stop after finding a single w.

Since, we are able to enumerate, it is clear that

$\overline{L}$ is RE.

It is known from Q1(a) that L is not R.

If L = RE and $\underbrace{\overline{L} = RE}_{\text{known}}$ $\Rightarrow$ $L = R$ (Contradiction!)

$\Rightarrow$ L is not RE

Therefore, both the constraints for the union are satisfied.

Now, we will consider L_i:

* $\overline{L_i} = \overline{L} \cup \{M_i\}$

Now since $\overline{L}$ and $\{M_i\}$ are both RE.

($\{M_i\}$ is RE because we can define it using encoding)

their union must be RE $\Rightarrow$ $\overline{L_i}$ = RE

* $L_i \not\subseteq L_j \quad \forall\ i \neq j$

L_i has M_i and L_j has M_j extra in their language sets. Hence they can't be subsets of each other.

* $L_i \neq RE$

If L_i = RE, we already know that M_i = RE (because we can encode it in string form)

then $L = L_i \cup \{M_i\}$ = RE (Contradiction!)

$\Rightarrow$ $L_i \neq RE$

Thus, all the constraints on L_i are satisfied.

$\therefore$ Hence, we have given a suitable example of $F = \{w \mid \exists i \in \mathbb{N}\ w \in L_i\}$