CSE 303 115060128
HW11

Problem 1:-

In the given simpler algorithm, there are some inacuracies hence that algo does not work.

In Step 2, if M loops of any of $S_i$, then it will ~~run~~ forever. ~~It~~ cannot check anything after $S_i$
So E will fail to enumerate its language.
~~So~~ the forward direction proof works in parallel for all strings so if it loops on certain string, we can discard that part. Hence this proof doesnt work.

Problem 2:-

For a value of $(x_1, x_2, \ldots, x_k)$ is a particular unique value.

So in Step 1, we will have infinite choices for $x_1$, infinite choices for $x_2$ .... infinite choices for $x_k$.

So to store these values it will require infinite memory which is not possible.

For Step 2, evaluating p on infinitely many values would take infinite processing time which again is impossible.

So turing machine "M" bad requires infinite memory & time to execute steps 1&2.

But these steps have to be completed in a finite no. of steps

Thus the given is not a description of a legitimate Turing machine.

Problem 3:-

Given $L = \{0^m \# 0^n \# 0^p \mid m \geq 0, n \geq 0, p \geq 0, m+n=p\}$

Let M be the TM that decides the Language L.
The Implementation level description of M is as follows:-

M = "On input w:

1) Scan the tape. If we find anything apart from 0 & #, reject.
2) Scan the tape from left. If there is # mark it as ⊔. ~~Scan again~~ If not present reject.
3). Scan to the right from ⊔. If there is #, no # then reject.
4) Start from #, mark the first 0 to the right of # ~~mark~~ it, which is unmarked & mark it.
5) Mark the leftmost 0 unmarked which is still to the left of #.
6). Repeat ~~steps~~ 4-5 till no 0's remain to the right of # & ~~rigt~~ left of #. If this is case accept. If there are 0's to the left

of # and no 0's to the right of #, reject.
Similarly if there are no 0's to the right of #
& no 0's to the left of #, reject.

Commentary.
Tape

000 # 00 # 00000

| | |
|---|---|
| ⇒ 000 ⊔ 00 # 00000 | 000 ⊔ 00 # 00000 |
| ⇒ 000 ⊔ ø0 # ø0000 | ø00 ⊔ 00 # ø0000 |
| ⇒ 00ø ⊔ øø # øø000 | øø0 ⊔ 00 # øø000 |
| ⇒ 00ø ⊔ øø # øøø00 | øøø ⊔ 00 # øøø00 |
| ⇒ 0øø ⊔ øø # øøøø0 | øøø ⊔ ø0 # øøøø0 |
| ⇒ øøø ⊔ øø # øøøøø | øøø ⊔ øø # øøøøø |

Problem 4:-

Given $L = \{ 0^n \# 0^{2^n} \in \{0,1\}^* \mid n \geq 0 \}$

Let M be the TM that decides the language L.
The implementation level description of M is as follows.

1) Scan the tape from L to R. If we find anything apart from 0, #, reject.
2) Scan from the ~~left~~. If ~~we~~ we find #, mark it as ⊔. If we dont find reject.
3). Scan to the right of ⊔ if we find # reject.

4) Scan to the right of ⊔ symbol for the first 0. If 0 does not exist reject. If it exists mark it.

5. Scan ~~to the~~ to the left of ⊔ for first unmarked 0 & mark it. If no unmarked 0's are left, go to.

6) Scan to the right of ⊔ for last marked 0 & replace it with \$ symbol.

7.) Continue scanning to the right of ⊔ for first unmarked 0 & mark it as \$. If no 0 was unmarked, reject.

8) Zig-Zag between corresponding pos to the right of ⊔ & right of ~~seco~~ 2nd \$ symbol & for each marked 0 to the right of ⊔ find an unmarked 0 to right of second \$. If at any pt unmarked 0 does not exist reject.

9). Scan to the right of ⊔ & mark the 0's where there was \$.

10) Repeat step 5 onwards

11.) Place head to the right of ⊔ symbol & find unmarked 0, If no unmarked 0 exists, accept else reject.

Problem 5.

We need to show that Turing machines with left reset recognize the class of Turing-recognisable languages.

Let turing machine with left reset be M.
Let N be a normal turing machine.

~~If we can show M behaves in the same way as N, then the proof is done.~~

We will show M simulates N.

When N makes right transition, M follows it in the same way as N does.
When N makes a left transition, with symbol a, b M replaces it with A or B resp.
So the alphabet set N is M $\cup$ {A, B} & then M does a left reset.
Then shift all content of the tape by one position to the right for all symbols other than {A, B}.

The above process is repeated until all the content of the tape are shifted to the right & then M does a reset again.
All right transactions are checked.
Whenever it reaches to {A, B} it works in the same way as N does.