

29/30

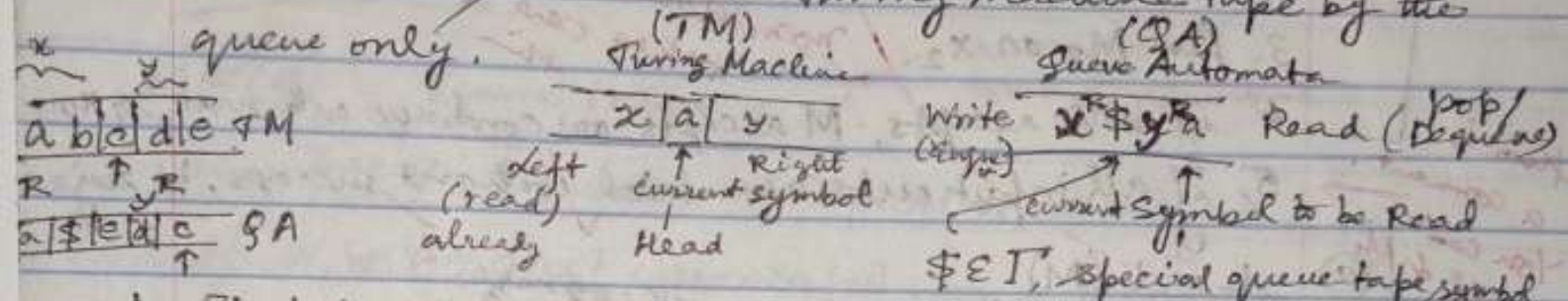
CMSC 651

Sandipan Roy

Homework 1

10/10

1. (\Rightarrow) Let's simulate a Turing machine with the queue automata ~~any given step~~ ^{transition} of Turing machine $\delta: Q \times T \rightarrow Q \times T \times \{L, R\}$. Since the input tape is read only, ~~that~~ for queue automata we need to simulate the Turing machine tape by the queue only.



1. First push ~~all~~ the entire ^{assume input} input string to the queue and terminate ^{has an end marker} by $\$$. (For FIFO property of the queue, the read end will have the first symbol to be read.)
2. ~~For~~ To simulate ~~TM~~ ^{after} $x a a y \vdash_M x b a y$ (TM moves right and current tape symbol a is replaced by b) ~~can~~ by QA, just pop ~~the~~ the current symbol a from right and push it in the front.
3. To simulate $x a a y \vdash_M x a b a y$ (TM replaces a by b and moves left), ~~QA~~ needs to pop current symbol a from rear, push b in the front, by two circular shifts to the left (anti-clockwise). For circular shifting to left, place a marker $\#$ at each end of q , replace each queue symbol x by (w, x) where w is the immediate left symbol (remember by extra state q_w), then dequeue (w, x) and enqueue (w, x) until $x = \#$, then enqueue $\#$ followed by w , finally repeatedly dequeue (w, x) and enqueue w until $\#$ is popped.

(\Leftarrow) Simulate QA with a 2-tape TM (equivalent to single tape TM). First tape contains the input string x , second tape, the queue. We need to simulate the FIFO push/pop. Queue alphabet contains $\$$. ~~push~~ initialization of queue tape: Write a $\$$.
 push: find the first blank space on the tape and write the symbol.
 pop: " " " symbol on tape $\neq \$$, read the symbol, replace

9/10 Let $L_1 = L(M_1)$
 $L_2 = L(M_2)$

2. (b) Construct an NDTM that ~~decides~~ ^{M recognizes} concatenation of L_1, L_2

$M(x)$ (M on input x)

1. for each ^{possible} way cut $x = x_1 x_2$ into two parts x_1, x_2
2. Run M_1 on x_1
3. Run M_2 on x_2 non-determinism takes one of them
4. if both accepts, M accepts. or continue with next x_1, x_2
5. if all input cuts are tried without success, M rejects

this is a comment
 you can tell if M_1 or M_2 solver

(c) $L = L(M)$

Construct NDTM M^* that ^{recognizes} ~~decides~~ L^*

$M^*(x)$

1. for each possible way cut $x = x_1 x_2 \dots x_n$ into n parts not the same n
2. Run M on $x_i \quad \forall i = 1, 2, \dots, n$
3. If M accepts on all of x_i , M^* accepts what if M runs forever?
4. or continue with next possible cut
5. if all cuts are tries without success, M^* rejects

Since length of x is finite n will be finite

(d) $L_1 = L(M_1)$

$L_2 = L(M_2)$

Construct TM M that ^{recognizes} ~~decides~~ $L_1 \cap L_2$

$M(x)$

1. Run M_1 on x , if M_1 rejects M rejects
2. Run M_2 on x , if M_2 rejects M rejects
3. if both M_1, M_2 accept, M accepts,

it's ok here
 If M_1 or M_2 runs forever

3. Let L be an infinite TR language.

$\Leftrightarrow \exists$ enumerator E which enumerates all words $w \in L$.
Let's construct a new enumerator M' which simulates E .

~~TM E~~ M'

$M'(x)$

Ignore input x

enumerators
don't have input

Simulate E . \forall enumerated word w do the following:

1. if w is the first enumerated word, remember it, print and proceed with next enumerated word. ~~(remember it)~~
largest word $\leftarrow w$.
2. else if $w \leq$ ~~largest word~~ (last remembered) then
lexicographic ignore w .

else if $w >$ largest word then
lexicographic

largest word $\leftarrow w$. (remember it)
cache

Go to step 1 (proceed with next enumerated word).

if $L' \equiv L(M')$, L' is still infinite since

$$\forall w' \in L' : \exists w \in L : w' < w$$

Also, since all $w' \in L$ are enumerated in lexicographic order, L' is decidable. $L' \subseteq L$ since the enumerator E for L and L' is infinite. (Proved) ✓