Student Information

Full Name : Ertuğrul Aypek
Id Number : 2171270

Answer 1

a.

Following the algorithm to construct a bottom-up parser on p. 170 in the book, we have following PDA M:
$M = (K, \Sigma, \Gamma, \Delta, p, F)$,
$K = \{p, q\}$,
$\Sigma = \{a, b, c\}$,
$\Gamma = \{S, X, a, b, c\}$,
p is the start symbol,
$F = \{q\}$,
$\Delta = \{((p, a, e), (p, a))$,
$((p, b, e), (p, b))$,
$((p, c, e), (p, c))$,
$((p, e, XaXSa), (p, S))$,
$((p, e, XbXSb), (p, S))$,
$((p, e, c), (p, S))$,
$((p, e, Xa), (p, X))$,
$((p, e, Xb), (p, X))$,
$((p, e, e), (p, X))$,
$((p, e, S), (q, e))\}$

b.

$(p, abbcbabbaa, e) \vdash_M (p, bbcbabbaa, a)$
$(p, bbcbabbaa, a) \vdash_M (p, bcbabbaa, ba)$
$(p, bcbabbaa, ba) \vdash_M (p, cbabbaa, bba)$
$(p, cbabbaa, bba) \vdash_M (p, babbaa, cbba)$
$(p, babbaa, cbba) \vdash_M (p, babbaa, Sbba)$
$(p, babbaa, Sbba) \vdash_M (p, babbaa, XSbba)$
$(p, babbaa, XSbba) \vdash_M (p, abbaa, bXSbba)$
$(p, abbaa, bXSbba) \vdash_M (p, abbaa, XbXSbba)$
$(p, abbaa, XbXSbba) \vdash_M (p, abbaa, Sba)$
$(p, abbaa, Sba) \vdash_M (p, bbaa, aSba)$
$(p, bbaa, aSba) \vdash_M (p, bbaa, XaSba)$
$(p, bbaa, XaSba) \vdash_M (p, bbaa, XSba)$
$(p, bbaa, XSba) \vdash_M (p, baa, bXSba)$

$(p, baa, bXSba) \vdash_M (p, baa, XbXSba)$
$(p, baa, XbXSba) \vdash_M (p, baa, Sa)$
$(p, baa, Sa) \vdash_M (p, aa, bSa)$
$(p, aa, bSa) \vdash_M (p, aa, XbSa)$
$(p, aa, XbSa) \vdash_M (p, aa, XSa)$
$(p, aa, XSa) \vdash_M (p, a, aXSa)$
$(p, a, aXSa) \vdash_M (p, e, aaXSa)$
$(p, e, aaXSa) \vdash_M (p, e, XaaXSa)$
$(p, e, XaaXSa) \vdash_M (p, e, XaXSa)$
$(p, e, XaXSa) \vdash_M (p, e, S)$
$(p, e, S) \vdash_M (q, e, e)$
Since $q \in F$, $w = abbcbabbaa$ can be generated by given CFG G.

Answer 2

a.

Let $M = (K, \Sigma, \delta, s, H)$ be a Turing machine. $K = \{s, h, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_10, q_11, q_12\}$
$\Sigma = \{1, \sqcup, \triangleright\}$
$H = \{h\}$
δ has the followings:
$\delta(s, \sqcup) = (q_1, \rightarrow)$,
$\delta(q_1, 1) = (q_2, \rightarrow)$,
$\delta(q_2, 1) = (q_1, \rightarrow)$,
$\delta(q_1, \sqcup) = (q_3, \leftarrow)$,
$\delta(q_3, 1) = (q_4, \sqcup)$,
$\delta(q_4, 1) = (q_5, \leftarrow)$,
$\delta(q_4, \sqcup) = (q_5, \leftarrow)$,
$\delta(q_5, 1) = (q_3, \leftarrow)$,
$\delta(q_5, \sqcup) = (q_3, \leftarrow)$,
$\delta(q_3, \sqcup) = (q_6, \rightarrow)$,
$\delta(q_6, 1) = (q_7, \rightarrow)$,
$\delta(q_6, \sqcup) = (q_7, \rightarrow)$,
$\delta(q_7, 1) = (q_8, \rightarrow)$,
$\delta(q_7, \sqcup) = (q_8, \rightarrow)$,
$\delta(q_8, 1) = (q_9, \sqcup)$,
$\delta(q_8, \sqcup) = (h, \sqcup)$,
$\delta(q_9, 1) = (q_{10}, \leftarrow)$,
$\delta(q_9, \sqcup) = (q_{10}, \leftarrow)$,
$\delta(q_{10}, \sqcup) = (q_{11}, 1)$,
$\delta(q_{10}, 1) = (q_{11}, 1)$,
$\delta(q_{11}, 1) = (q_6, \rightarrow)$,
$\delta(q_{11}, \sqcup) = (q_6, \rightarrow)$,

$\delta(q_2, \sqcup) = (q_{12}, 1)$,
$\delta(q_{12}, 1) = (h, \rightarrow)$

Answer 3

There are some differences between Turing Machines and Finite State Automaton's like Turing Machines' memory and output abilities. Since a move-restricted Turing Machine can not move left, it can not not read what it wrote to the left. So, it loses its ability of memory. It can only remember what is in head position and write over it. However, having ability to write over head position does not make move-restricted Turing Machine more powerful than a FSA.
Hence, a move-restricted Turing Machine is like a FSA with an extra of giving output. Since FSA's can recognize regular languages only, the set of languages decided by move-restricted Turing Machines is regular languages.

Answer 4

a.

A queue based TM is a sextuple. Let M be a queue based TM.
$M = (K, \Sigma, \Gamma, \delta, s, H)$ where,
K is the set of states.
$\Sigma \supseteq \{\sqcup, \triangleright\}$ is the set of input symbols.

$\Gamma = \{\swarrow, \searrow, \downarrow, \uparrow\}$ is the set of queue operations symbols where $\swarrow$ is used as front, $\searrow$ is used as rear, $\downarrow$ is used as dequeue and $\uparrow$ is used as enqueue.
δ is the transition function from (K-H)xΣxΣxΣxΓ (first two Σ's are for front end rear ends respectively, third Σ is for enqueue input) to KxΣxΣ (Σ's are for front end rear ends respectively) When using operations other than enqueue, enqueue input is ϵ. s$\in K$ is the starting state.
H$\subseteq$K is the set of halting states.

b.

A configuration is a member of Kx$\triangleright\Sigma^*$xΣ^*x$(\Sigma - \{\sqcup\}) \cup \{e\}$. It can be explained as: (state) x (first head's symbol) x (symbols between the first and the second heads) x (second head's symbol).

c.

$(q_1, a_1, a_2 w_1 a_3, a_4) \vdash_M (q_2, b_1, b_2 v_1 b_3, b_4)$

- for $\delta(q_1, a_1, a_4, c, \uparrow)$=$(q_2, b_1, b_4)$ where $b_4 = c$, $b_3 = a_4$, $b_1 = a_1$, $b_2 = a_2$, $v_1 = w_1 a_3$
- for $\delta(q_1, a_1, a_4, \epsilon, \downarrow)$=$(q_2, b_1, b_4)$ where $(a_1 \neq \sqcup \cap a_1 \neq \epsilon)$, $b_1 = a_2$, $b_2 v_1 = w_1$, $b_3 = a_3$, $b_4 = a_4$
- for $\delta(q_1, a_1, a_4, \epsilon, \downarrow)$=$(q_2, b_1, b_4)$ where (a_1=$\sqcup \cup a_1$=ϵ), $b_1 = a_1$, $b_2 = a_2$,$v_1 = w_1$, $b_3 = a_3$, $b_4 = a_4$
- for $\delta(q_1, a_1, a_4, \epsilon, \swarrow)$=$(q_2, b_1, b_4)$ where $b_1 = a_1$, $b_2 = a_2$,$v_1 = w_1$, $b_3 = a_3$, $b_4 = a_4$

- for $\delta(q_1, a_1, a_4, \epsilon, \searrow)$=$(q_2, b_1, b_4)$ where $b_1 = a_1$, $b_2 = a_2$,$v_1 = w_1$, $b_3 = a_3$, $b_4 = a_4$

d.

Queue-based deterministic TM is equivalent to the standard TM since any operations that can be done with one of them can be done by the other. To see it:
To access the rear element in the queue-based TM, it is necessary to go right until the end of the tape in standard TM. There can also be a marker at the end of the input string for simplicity. If there is a marker, we can move head to one step left after we reached the marker to be at the rear position.
To access the front element in the queue-based TM, it is necessary to go left until we reach the start symbol $\triangleright$ in standard TM. After that, we can move one step right to reach the front element.
To enqueue in the queue-based TM, it is necessary to go to the one step right after the rear element and write it over there in standard TM.
To dequeue in the queue-based TM, it is necessary to go to the first input string element and make it blank. And shift the whole input string one step left in standard TM.
To go right in standard TM, it is necessary to dequeue and enqueue what is dequeued. So that we can access it via front operation.
To go left in standard TM, we should use a marker. When such a left operation is requested, we should immediately enqueue our marker. And start to dequeue and enqueue what is dequeued loop until front head has become our marker. When it happens, our rear head shows the left which is the result of requested operation. After then we can dequeue our marker.
To write in standard TM, it is necessary to enqueue a marker, enqueue the new symbol that will be replaced and dequeue the first symbol. After that, start to dequeue and enqueue what is dequeued loop until front head has the marker. After that dequeue marker and stop. Now we have new symbol in our front head. For example to oberve abcd to ebcd: abcdK - abcdKe - bcdKe - cdKeb - dKebc - Kebcd - ebcd.

e.

The idea of algorithm is very straightforward. First enqueue a marker at the end.
We take and dequeue the first symbol and go right until symbol c is found. Move one more step right and compare it with what we took at the beginning.
If they are the same, dequeue it and continue to loop until we encounter a c as a front and a marker as a rear stop computation and give the output that given string is in language.
If they are not same, stop computation and give the output that given string is not in language.
$M = (K, \Sigma, \Gamma, \delta, s, H)$
K=$\{s, y, n, q_0, q_a, q_b, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}, q_{13}, q_{14}\}$
$\Sigma = \{a, b, c, \sqcup, \triangleright, \sharp\}$
Γ is given in part a.
s is the initial state.
$H = \{y, n\}$

δ has the followings:
$\delta(s, \Sigma, \Sigma, \sharp, \uparrow) = (q_0, \Sigma, \sharp)$,
$\delta(q_0, a, \Sigma, \epsilon, \downarrow) = (q_a, \Sigma, \Sigma)$,
$\delta(q_0, b, \Sigma, \epsilon, \downarrow) = (q_b, \Sigma, \Sigma)$,
$\delta(q_a, a, \Sigma, \epsilon, \downarrow) = (q_1, \Sigma, \Sigma)$,
$\delta(q_1, \Sigma, \Sigma, a, \uparrow) = (q_a, \Sigma, a)$,
$\delta(q_a, b, \Sigma, \epsilon, \downarrow) = (q_2, \Sigma, \Sigma)$,
$\delta(q_2, \Sigma, \Sigma, b, \uparrow) = (q_a, \Sigma, b)$,
$\delta(q_a, c, \Sigma, \epsilon, \downarrow) = (q_3, \Sigma, \Sigma)$,
$\delta(q_3, \Sigma, \Sigma, c, \uparrow) = (q_4, \Sigma, c)$,
$\delta(q_4, b, \Sigma, \epsilon, \Gamma) = (n, \Sigma, \Sigma)$,
$\delta(q_4, a, \Sigma, \epsilon, \downarrow) = (q_5, \Sigma, \Sigma)$,
$\delta(q_5, a, \Sigma, \epsilon, \downarrow) = (q_6, \Sigma, \Sigma)$,
$\delta(q_6, \Sigma, \Sigma, a, \uparrow) = (q_5, \Sigma, a)$,
$\delta(q_5, b, \Sigma, \epsilon, \downarrow) = (q_6, \Sigma, \Sigma)$,
$\delta(q_6, \Sigma, \Sigma, b, \uparrow) = (q_5, \Sigma, b)$,
$\delta(q_5, \sharp, \Sigma, \epsilon, \downarrow) = (q_7, \Sigma, \Sigma)$,
$\delta(q_7, \Sigma, \Sigma, \sharp, \uparrow) = (q_0, \Sigma, b)$,
$\delta(q_b, a, \Sigma, \epsilon, \downarrow) = (q_8, \Sigma, \Sigma)$,
$\delta(q_8, \Sigma, \Sigma, a, \uparrow) = (q_b, \Sigma, a)$,
$\delta(q_b, b, \Sigma, \epsilon, \downarrow) = (q_9, \Sigma, \Sigma)$,
$\delta(q_9, \Sigma, \Sigma, b, \uparrow) = (q_b, \Sigma, b)$,
$\delta(q_b, c, \Sigma, \epsilon, \downarrow) = (q_{10}, \Sigma, \Sigma)$,
$\delta(q_10, \Sigma, \Sigma, c, \uparrow) = (q_11, \Sigma, c)$,
$\delta(q_11, a, \Sigma, \epsilon, \Gamma) = (n, \Sigma, \Sigma)$,
$\delta(q_11, b, \Sigma, \epsilon, \downarrow) = (q_12, \Sigma, \Sigma)$,
$\delta(q_12, a, \Sigma, \epsilon, \downarrow) = (q_13, \Sigma, \Sigma)$,
$\delta(q_13, \Sigma, \Sigma, a, \uparrow) = (q_12, \Sigma, a)$,
$\delta(q_12, b, \Sigma, \epsilon, \downarrow) = (q_13, \Sigma, \Sigma)$,
$\delta(q_13, \Sigma, \Sigma, b, \uparrow) = (q_12, \Sigma, b)$,
$\delta(q_12, \sharp, \Sigma, \epsilon, \downarrow) = (q_14, \Sigma, \Sigma)$,
$\delta(q_14, \Sigma, \Sigma, \sharp, \uparrow) = (q_0, \Sigma, b)$,
$\delta(q_0, c, \sharp, \Sigma, \Gamma) = (y, \Sigma, \Sigma)$,
$\delta(q_0, c, a, \Sigma, \Gamma) = (n, \Sigma, \Sigma)$,
$\delta(q_0, c, b, \Sigma, \Gamma) = (n, \Sigma, \Sigma)$

A use of notation: I used Σ, Γ to show that those units do not have an importance to determine the next states. So they can be any symbol from alphabet that I defined at the beginning.

Answer 5

a.

First let's give the algorithm. For each a's make that symbol a blank, go to the end, delete three c's and return back to beginning. Continue this loop until we reach a b in the first input string other than blank symbol. Then, for each blank symbol that is left of the b's, divide b's by two and shift input string one step left (to get rid of a blank symbol which is left of b's). If we have only one b in our tape as an input string, then given string is in the language. Here is a simulation on aabbbbcccccc
aabbbbccc, ⊔abbbbccc, ⊔abbbb, ⊔⊔bbbb, ⊔bb, b
To implement this machine, let us use the machines that are described in p.189 of the book, the function that we have described in q2 to half b's and also shift the tape elements one step left function described in the book.
Let $M = (K, \Sigma, \delta, s, H)$ be a Turing machine. Where,
$K = \{s, q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}, q_{13}, q_{14}\}$,
$\Sigma = \{a, b, c, \triangleright, \sqcup\}$
$H = y, n$
$\delta \subseteq$ (K-H)xΣxKx($\Sigma \cup \{\leftarrow, \rightarrow, L_{\sqcup}, R_{\sqcup}, L_{\overline{\sqcup}}, R_{\overline{\sqcup}}, B_{half}, L_{shift}\}$) $\delta(s, \triangleright) = (q_0, \rightarrow)$,
$\delta(q_0, a) = (q_0, \sqcup)$,
$\delta(q_0, \sqcup) = (q_1, R_{\sqcup})$,
$\delta(q_1, \sqcup) = (q_2, \leftarrow)$,
$\delta(q_2, c) = (q_3, \sqcup)$,
$\delta(q_3, \sqcup) = (q_4, \leftarrow)$,
$\delta(q_4, c) = (q_5, \sqcup)$,
$\delta(q_5, \sqcup) = (q_6, \leftarrow)$,
$\delta(q_6, c) = (q_7, \sqcup)$,
$\delta(q_7, \sqcup) = (q_8, L_{\sqcup})$,
$\delta(q_8, \sqcup) = (q_0, \rightarrow)$,
$\delta(q_0, b) = (q_9, \leftarrow)$,
$\delta(q_9, \sqcup) = (q_{10}, L_{shift})$,
$\delta(q_{10}, \Sigma) = (q_{11}, R_{\sqcup})$,
$\delta(q_{11}, \sqcup) = (q_{12}, B_{half})$,
$\delta(q_{12}, \Sigma) = (q_{13}, L_{\overline{\sqcup}})$,
$\delta(q_{13}, b) = (q_0, b)$,
$\delta(q_0, b) = (q_{14}, \rightarrow)$,
$\delta(q_{14}, \sqcup) = (y, \Sigma)$,

Note: Since I have not enogh time, I did not specify rejecting situations but they are clear I think. Other than these transition functions, output can be taken as rejecting (n).

b.

Let $G = (V, \Sigma, R, S)$ where,
$V = \{a, b, c, S, M, A, B, C, \$\}$,
$\Sigma = \{a, b, c\}$,
$R = \{$
$S \rightarrow b|M$,
$M \rightarrow ABCCC|e$,
$B \rightarrow ABCCC|b$,
$A \rightarrow a\$$,
$C \rightarrow c$,
$\$a \rightarrow a\$$,
$\$b \rightarrow bb\$$,
$\$c \rightarrow c\}$ are all the rules,
S is the start symbol.

Answer 6

a.

Since L_1 is a regular grammar, it is a regular language.
Since there is a deterministic top down parser for L_2, it is a deterministic Context Free Language.
Since there is only an "inherently ambiguous" CFG for L_3, it is a nondeterministic Context Free Language.
Since L_4's complement exists, it may be a recursive language but it may not be a Context Free Language.
Since there is an unrestricted grammar for L_5, it may be a recursively enumerable language.
Since all above languages can not be out of the recursively enumerable languages set, there exists TM's to accept all of them.

b.

For L_1, L_2, L_3 and L_5 there is no membership problems since they are under a full subset of some languages.
However, for L_4 there is a membership problem since L_4 may be recursive but not Context Free. Since recursive languages set contains CFL's set, there is a hole in L_4.

c.

d.

If L_2 is a reguar language, then L is a recursive language. So, yes, since L is a recursive language, $\overline{L}$ is recursive also. Since all recursive languages are recursively enumerable also, a TM can be

constructed that semidecides $\overline{L}$.
If L_2 is a context free language, then its complement can not be taken. So, no.
As shown, we can not come up with a TM that semidecides $\overline{L}$.