Student Information

Full Name : Ugur Duzel
Id Number : 2171569

Answer 1

a.

Let PDA $M = (\{p, q\}, \Sigma, \Gamma, p, \{q\})$ be the bottom-up parser for G where $\Sigma = \{a, b, c\}$, $\Gamma = V$ and Δ contains the following:

$$\begin{aligned}\Delta = \{&((p, a, e), (p, a)), \quad (1)\\ &((p, b, e), (p, b)), \quad (2)\\ &((p, c, e), (p, c)), \quad (3)\\ &((p, e, XaXSa), (p, S)), \quad (4)\\ &((p, e, XbXSb), (p, S)), \quad (5)\\ &((p, e, c), (p, S)), \quad (6)\\ &((p, e, Xa), (p, X)), \quad (7)\\ &((p, e, Xb), (p, X)), \quad (8)\\ &((p, e, e), (p, X)), \quad (9)\\ &((p, e, S), (q, e))\} \quad (10)\end{aligned} \tag{1}$$

b.

The table for successive configurations is on the next page. The string is accepted by M because input and stack both are empty and we are in the final state at the end.

Table 1:

Step	**State**	**Unread Input**	**Stack**	**Transition**
0	p	abbcbabbaa	e	-
1	p	bbcbabbaa	a	1
2	p	bcbabbaa	ba	2
3	p	cbabbaa	bba	2
4	p	babbaa	cbba	3
5	p	babbaa	Sbba	6
6	p	babbaa	XSbba	9
7	p	abbaa	bXSbba	2
8	p	abbaa	XbXSbba	9
9	p	abbaa	Sba	5
10	p	bbaa	aSba	1
11	p	bbaa	XaSba	9
12	p	bbaa	XSba	7
13	p	baa	bXSba	2
14	p	aa	XbXSba	9
15	p	aa	Sa	5
16	p	a	aSa	1
17	p	a	XaSa	9
18	p	a	XSa	7
19	p	e	aXSa	1
20	p	e	XaXSa	9
21	p	e	S	4
22	q	e	e	10

Answer 2

a.

$M = (K, \Sigma, \delta, s, H)$ where

$$K = \{s, q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, h\}$$

$$\Sigma = \{1,\ \sqcup,\ \triangleright,\ \leftarrow \rightarrow,\ *\}$$

$$H = \{h\}$$

$$\begin{aligned}
&\delta(s, \triangleright) = (s, \triangleright, \rightarrow)\\
&\delta(s, \sqcup) = (q_0, \sqcup, \rightarrow)\\
&\delta(q_0, 1) = (q_1, 1, \rightarrow)\\
&\delta(q_1, 1) = (q_0, 1, \rightarrow)\\
&\delta(q_1, \sqcup) = (h, 1, \rightarrow)\\
&\delta(q_0, \sqcup) = (q_2, *, \leftarrow)\\
&\delta(q_2, 1) = (q_3, \sqcup, \leftarrow)\\
&\delta(q_3, 1) = (q_2, 1, \leftarrow)\\
&\delta(q_2, \sqcup) = (q_4, \sqcup, \rightarrow)\\
&\delta(q_3, \sqcup) = (q_4, \sqcup, \rightarrow)\\
&\delta(q_4, 1) = (q_4, 1, \rightarrow)\\
&\delta(q_4, \sqcup) = (q_5, \sqcup, \rightarrow)\\
&\delta(q_5, 1) = (q_6, \sqcup, \leftarrow)\\
&\delta(q_5, \sqcup) = (q_5, \sqcup, \rightarrow)\\
&\delta(q_5, *) = (h, \sqcup, \rightarrow)\\
&\delta(q_6, \sqcup) = (q_6, \sqcup, \leftarrow)\\
&\delta(q_6, 1) = (q_7, 1, \rightarrow)\\
&\delta(q_7, \sqcup) = (q_4, 1, \rightarrow)
\end{aligned} \tag{2}$$

Answer 3

The set of languages decided by Move-Restricted Turing Machines is the set of regular languages. If it had more than one tape or were able to move the head both sides than we could have been able to imitate stack operations. In this case there no way to imitate such operations therefore the set if not the set of context-free languages.
Moreover, we can go right, read and write. Only write capability is the extra feature compared to finite state machine. Since we go only right there is no use of write feature. Therefore it is the set of regular languages.

Answer 4

a.

$M = (K, \Sigma, O, \Delta, s, H)$ where,
K is a finite set of states,
Σ is an aplhabet containing blank symbol $\sqcup$ and left end symbol $\triangleright$,
$O = \{"E", "D"\}$ is the set of operations,
$s \in K$ is the initial state,
$H \subseteq K$ is the set of halting states,
δ is the transition function from $(K - H)x\Sigma x\Sigma$ to $KxOx\Sigma$

b.

Configuration is in the following form:

$$Kx \triangleright ((\Sigma - \{\sqcup\}) \cup \{e\})\Sigma^*(\Sigma - \{\sqcup\}) \cup \{e\})$$

c.

$$(q_1, a_1w_1b_1) \vdash_M (q_2, a_2w_2b_2)$$

if for some $\sigma \in \Sigma$, $\delta(q_1, a_1, b_1) = (q_2, "E", \sigma)$ then $b_2 = \sigma$ and $a_2w_2 = a_1w_1b_1$
And $\delta(q_1, a_1, b_1) = (q_2, "D", \_)$ then $a_2w_2b_2 = w_1b_1$

(Note that in the dequeue operation underscore means that we don't care the sigma)

Let $M = (K, \Sigma, O, s, \{y, n\})$ be a Queue-based Turing Machine then

$$L(M) = \{w \in \Sigma^* \ : \ (s, w) \vdash_M^* (y, w')\}$$

d.

Assuming the initial configuration as $\triangleright\underline{\sqcup}w$ and no blank characters exists in the string.
Assume we are currently in the middle of the string.
- To get to the front element we go left until we see a blank when we do we go right for one step then we get the front element.
- To get to the rear element we go right until we see a blank when we do we go left for one step then we get the rear element.
Now we can choose the right transition from the given Queue-based TM by looking at (we can use a temporary variable for front just as we do with the basic machine transitions) the state, front element and rear element. Then we can decide whether it is defined as enqueue operation or dequeue operation.
- To perform enqueue operation we get to the rear element just like we did earlier but this time we don't go left for one step we stay at the last blank cell and we basically write the given symbol.

- To perform dequeue operation we get to the front element just like we did earlier but this time we don't go right for one step we stay at the first blank cell and we basically write the blank symbol. This shows that all the operations of Queue-based TM can be done with a standard Turing Machine. Thereforem Queue-based Turing Machines is equivalent to standard Turing Machines.

e.

Before answering the question, to clarify I used underscore where the element does not matter to keep it simpler for both me and you, dear reading assistant. Otherwise there would be a lot more transitions.
Let $M = (K, \{a, b\}, \{"E", "D"\}, \delta, s, \{y, n\})$ be a Queue-based TM,
where $K = \{s, p_a, p'_a, p_b, p'_b, q_a, q_b, q_{control}\}$ and transition function is the following

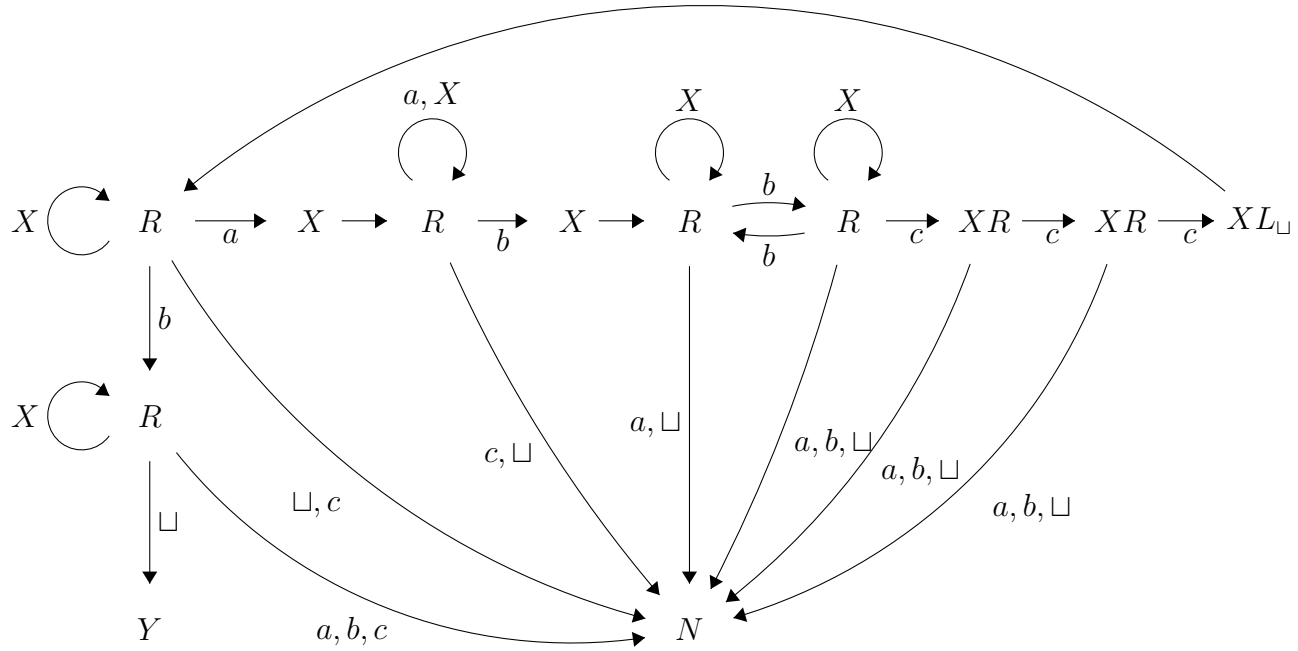
$$
\begin{aligned}
&\delta(s, a, \_) = (p_a, "D", \_), \delta(s, b, \_) = (p_b, "D", \_),\\
&\delta(p_a, a, \_) = (p'_a, "E", a), \delta(p'_a, a, \_) = (p_a, "D", \_),\\
&\delta(p_a, b, \_) = (p'_a, "E", b), \delta(p'_a, b, \_) = (p_a, "D", \_),\\
&\delta(p_a, c, \_) = (p'_a, "E", c), \delta(p'_a, c, \_) = (q_a, "D", \_),\\
&\delta(q_a, a, \_) = (q_{control}, "D", \_), \delta(q_{control}, a, \_) = (p_a, "D", \_),\\
&\delta(q_a, b, \_) = (n, "D", \_),\\
&\delta(q_a, c, \_) = (n, "D", \_),\\
&\delta(p_b, a, \_) = (p'_b, "E", a), \delta(p'_b, a, \_) = (p_b, "D", \_),\\
&\delta(p_b, b, \_) = (p'_b, "E", b), \delta(p'_b, b, \_) = (p_b, "D", \_),\\
&\delta(p_b, c, \_) = (p'_b, "E", c), \delta(p'_b, c, \_) = (q_b, "D", \_),\\
&\delta(q_b, b, \_) = (q_{control}, "D", \_), \delta(q_{control}, b, \_) = (p_b, "D", \_),\\
&\delta(q_b, a, \_) = (n, "D", \_),\\
&\delta(q_b, c, \_) = (n, "D", \_),\\
&\delta(q_{control}, c, \_) = (y, "D", \_)
\end{aligned}
\tag{3}
$$

My idea here was to record the first cell and go to the corresponding state, meaning if the read cell is a then I go to q_a. In this state a go right until I see a c until that time I enqueue every symbol I saw (except the first cell recorded). If the cell one after the c does not correspond to the state then I halt as "n" else I continue.

Answer 5

a.

Using the book's notations for the diagram.



b.

$G = (V, \Sigma, R, S)$ where $V = \{S, P, R, \#, a, b, c\}$, $\Sigma = \{a, b, c\}$ and R is as follows,

$$\begin{aligned} S &\to \#bP \\ P &\to RPccc \\ bR &\to Rbb \\ \#R &\to a\# \\ P &\to e \\ \# &\to e \end{aligned} \tag{4}$$

Answer 6

Before going to the subsections of this question, I want to show which language is what.
L_1 is a regular language because from L-G theorem we know that a language is regular if and only if there exists a regular grammer.
L_2 is DPCL because we can built deterministic top-down parser for it.
L_3 is CFL because we know that all DCFLs have unambiguous grammars. Therefore it suggests that L_3 is only CFL.
L_4 is decidable (recursive).
L_5 is semi-decidable (recursive enumerable).

a.

Yes. All five classes of the given languages are subsets of Turing Machines (recursively enumerable languages). Turing Machine has the most computational power among them.

b.

L_1, L_2, L_3 and L_4 are subsets of decidable languages therefore there are algorithms for them. However, L_5 is semi-decidable therefore there may not be an algorithm for L_5

c.

Let's say machine for L_1 is $M_1 = (K_1, \Sigma_1, \delta_1, s_1, H_1)$
Let's say machine for L_2 is M_2 . We can easily construct a machine for $\overline{L_2}$ by exchanging yes and no halting states of M_2. Let's call this machine M_2'.
Let's say machine for L_3 is $M_3 = (K_3, \Sigma_3, \delta_3, s_3, H_3)$
Let's say machine for L_4 is M_4 . We can easily construct a machine for $\overline{L_4}$ by exchanging yes and no halting states of M_4. Let's call this machine M_4'.
Let's say machine for L_5 is $M_5 = (K_5, \Sigma_5, \delta_5, s_5, H_5)$

To construct the left hand side of the union what we do is, we first start M_2' with the given input then when it halts we start it with M_1. Concurrently we can start it with M_5 both of them halt. Then we can loop this.

To construct the right side of the union what we do is, we start with M_3 and we loop it as many time as we want and then we continue with M_4'

d.

L is clearly a semi-decidable language. Decidable (recursive) languages are closed under complement. However, since L is a semi-decidable language there is no guarantee that there is a TM that semi-decides $\overline{L}$. Meaning recursive enumerable languages are not closed under complementation. For example (from our lecture with Aysenur Hoca),
$H = \{"M""w" \ : \ TMMhaltsonw\}$ is not recursive. But H is recursively enumerable. H is semi-decided by U.
U will halt on w if and only if M halts on w.
$H_1 = \{"M" \ : \ Mhaltsonw\}$ and then
$\overline{H_1} = \{$w is not a valid TM encoding or w is a TM M string but M does not halt on "M" $\}$
$\overline{H_1}$ is the diagonal language, and not recursive. Also it cannot be recursive enumerable.
This justifies the previous claim.