Student Information

Full Name : Furkan Göksel
Id Number : 2237436

Answer 1

a.

$M = (K, \Sigma, \delta, s, H)$ where,
K = $\{s, q, q_a, q_b, q_{al}, q_{bl}, h\}$,
$\Sigma = \{a, b, \sqcup, \triangleright\}$,
s = s,
H = {h},
where transitions as follows,

q	σ	$\delta(q, \sigma)$
s	a	(q, $\rightarrow$)
s	b	(q, $\rightarrow$)
s	$\sqcup$	(q, $\rightarrow$)
s	$\triangleright$	(s, $\rightarrow$)
q	a	(q_{al}, $\sqcup$)
q	b	(q_{bl}, $\sqcup$)
q	$\sqcup$	(h, $\sqcup$)
q	$\triangleright$	(q, $\rightarrow$)
q_{al}	a	(q_a, $\leftarrow$)
q_{al}	b	(q_a, $\leftarrow$)
q_{al}	$\sqcup$	(q_a, $\leftarrow$)
q_{al}	$\triangleright$	(q_{al}, $\rightarrow$)
q_{bl}	a	(q_b, $\leftarrow$)
q_{bl}	b	(q_b, $\leftarrow$)
q_{bl}	$\sqcup$	(q_b, $\leftarrow$)
q_{bl}	$\triangleright$	(q_{bl}, $\rightarrow$)
q_a	a	(q_a, $\leftarrow$)
q_a	b	(q_a, $\leftarrow$)
q_a	$\sqcup$	(h, a)
q_a	$\triangleright$	(q_a, $\rightarrow$)
q_b	a	(q_b, $\leftarrow$)
q_b	b	(q_b, $\leftarrow$)
q_b	$\sqcup$	(h, b)
q_b	$\triangleright$	(q_b, $\rightarrow$)

b.

(i) $(s, \triangleright \sqcup \sqcup b\underline{a}b) \vdash_M$
$(q, \triangleright \sqcup \sqcup ba\underline{b}) \vdash_M$ using $\delta(s, a) = (q, \rightarrow)$
$(q_{bl}, \triangleright \sqcup \sqcup ba\underline{\sqcup}) \vdash_M$ using $\delta(q, b) = (q_{bl}, \sqcup)$
$(q_b, \triangleright \sqcup \sqcup b\underline{a}) \vdash_M$ using $\delta(q_{bl}, \sqcup) = (q_b, \leftarrow)$
$(q_b, \triangleright \sqcup \sqcup \underline{b}a) \vdash_M$ using $\delta(q_b, a) = (q_b, \leftarrow)$
$(q_b, \triangleright \sqcup \underline{\sqcup}ba) \vdash_M$ using $\delta(q_b, b) = (q_b, \leftarrow)$
$(h, \triangleright \sqcup \underline{b}ba)_M$ using $\delta(q_b, \sqcup) = (h, b)$

(ii) $(s, \triangleright a\underline{a}a) \vdash_M$
$(q, \triangleright aa\underline{a}) \vdash_M$ using $\sigma(s, a) = (q, \rightarrow)$
$(q_{al}, \triangleright aa\underline{\sqcup}) \vdash_M$ using $\sigma(q, a) = (q_{al}, \sqcup)$
$(q_a, \triangleright a\underline{a}) \vdash_M$ using $\sigma(q_{al}, \sqcup) = (q_a, \leftarrow)$
$(q_a, \triangleright \underline{a}a) \vdash_M$ using $\sigma(q_a, a) = (q_a, \leftarrow)$
$(q_a, \underline{\triangleright}aa) \vdash_M$ using $\sigma(q_a, a) = (q_a, \leftarrow)$
$(q_a, \triangleright \underline{a}a) \vdash_M$ using $\sigma(q_a, \sqcup) = (q_a, \rightarrow)$
$(q_a, \underline{\triangleright}aa) \vdash_M$ using $\sigma(q_a, a) = (q_a, \leftarrow)$
$(q_a, \triangleright \underline{a}a) \vdash_M$ using $\sigma(q_a, \sqcup) = (q_a, \rightarrow)$
...
Machine never stops.

(iii) $(s, \triangleright \underline{a} \sqcup bb) \vdash_M$
$(q, \triangleright a\underline{\sqcup}bb) \vdash_M$ using $\sigma(s, a) = (q, \rightarrow)$
$(h, \triangleright a\underline{\sqcup}bb)$ using $\sigma(q, \sqcup) = (h, \sqcup)$

Answer 2

Initially, tape contains $(\triangleright\underline{\sqcup}babc)$, (each right arrow presents the input tape situation after the TM actions)

$(\triangleright\underline{\sqcup}babc) \rightarrow (\triangleright \sqcup \underline{b}abc) \rightarrow (\triangleright \sqcup \underline{b}abc)$ (create a register called a and put b in it) $\rightarrow (\triangleright \sqcup b\underline{a}bc) \rightarrow (\triangleright \sqcup ba\underline{b}c) \rightarrow (\triangleright \sqcup bab\underline{c}) \rightarrow (\triangleright \sqcup babc\underline{\sqcup}) \rightarrow (\triangleright \sqcup bab\underline{c}) \rightarrow (\triangleright \sqcup bab\underline{c})$ (create a register called b and put c in it) $\rightarrow (\triangleright \sqcup babc\underline{\sqcup}) \rightarrow (\triangleright \sqcup babc\underline{\sqcup}) \rightarrow (\triangleright \sqcup babc \sqcup \underline{\sqcup}) \rightarrow (\triangleright \sqcup babc \sqcup \underline{b})$ (write what register a contains) $\rightarrow (\triangleright \sqcup babc \sqcup \underline{c})$ (write what register b contains) $\rightarrow (\triangleright \sqcup babc \sqcup c\underline{\sqcup}) \rightarrow (\triangleright \sqcup babc \sqcup c\underline{\sqcup}) \rightarrow (\triangleright \sqcup babc \sqcup c \sqcup \underline{\sqcup}) \rightarrow (\triangleright \sqcup babc \sqcup c \sqcup \underline{b})$ Then the machine stops.

This Turing Machine takes string which is initially configured $(\triangleright \sqcup w)$, and with one move right take the first symbol of string, and saves it. Then by moving until see a blank, and then do one move left and saves the symbol in another register. With these moves TM actually saves first and last character of given string. However, while the first symbol can be any symbol from input alphabet ($\Sigma_0 = \Sigma - \{\sqcup, \triangleright\}$), the last symbol of the string should be c in order to be completed

TM's operations. Then after the save last symbol of string, and then with rest of the operations, TM creates such a string in the tape for a string u such that u = wc ($\triangleright \sqcup wc \sqcup c \sqcup x$) where x contains first symbol of the string u.

Answer 3

a.

For semidecider M we need a language such that in any word from the language machine M stops, otherwise machine never stops. Firstly let's consider for which strings M halts?
-For empty string machine can halt
-For strings just contains a M can halt
-For strings just contains b M can halt
-For strings in form a*b* M can halt
-For strings in form a,b* M can halt
This machine can halt any form of string, with going to $L_{\sqcup}$, since initial configuration is $\triangleright \sqcup w$ machine can always because in the left there is at least one $\sqcup$ that is reachable and machine can halt when reach this. So language is $\{a, b\}^*$.

b.

From definition we know that Let M = $(K, \Sigma, \delta, s, \{h\})$ be a Turing machine, let $\Sigma_0 \subseteq \Sigma - \{\sqcup, \triangleright\}$ be an alphabet, and let $w \in \Sigma^*$. Suppose that M halts on input w, and that $(s, \triangleright \underline{\sqcup} w) \vdash_M^* (h, \triangleright \underline{\sqcup} y)$ for some $y \in \Sigma^*$. Then y is called the output of M on input w, and is denoted M(w). Notice that M(w) is defined only if M halts on input w, and in fact does so at a configuration of the form $(h, \triangleright \underline{\sqcup} y)$ with $y \in \Sigma^*$.
Now let f be any function from Σ^* to Σ^*. We say that M computes function f if, for all $w \in \Sigma^*$, M(w) = f(w). That is, for all $w \in \Sigma^*$ M eventually halts on input w, and when it does halt, its tape contains the string $\triangleright \sqcup f(w)$.

For given machine our $\Sigma_0 = \{a, b\}$, and let's examine machine works:
1-For a given string such that in form of a^*, initial tape will be $\triangleright \underline{\sqcup} a^*$, and machine will move to until the first non-a character and it will see it is blank so it will return the first blank on the left again, and halt in a situation $\triangleright \underline{\sqcup} a^*$. While it is making this computation it doesn't change the number of a's so the function behaves like this for any input $w \in \{a\}^*$, M(w) will be $(s, \triangleright \underline{\sqcup} w) \vdash_M^*$ $(h, \triangleright \underline{\sqcup} f(w))$ such that f(w) = w. so, in a form of $w \in a^*$ f(w)=w.

2-For a given string such that in form of b^*, initial tape will be $\triangleright \underline{\sqcup} b^*$, and machine will move to until the first non-a character which is b and it will continue to move until the first non-b character it will see it is blank so it will return the first blank on the left again, and halt in a situation $\triangleright \underline{\sqcup} b^*$. While it is making this computation it doesn't change the number of b's so the function behaves like this for any input $w \in \{b\}^*$, M(w) will be $(s, \triangleright \underline{\sqcup} w) \vdash_M^* (h, \triangleright \underline{\sqcup} f(w))$ such that f(w) = w. so, in a form of $w \in b^*$ f(w)=w.

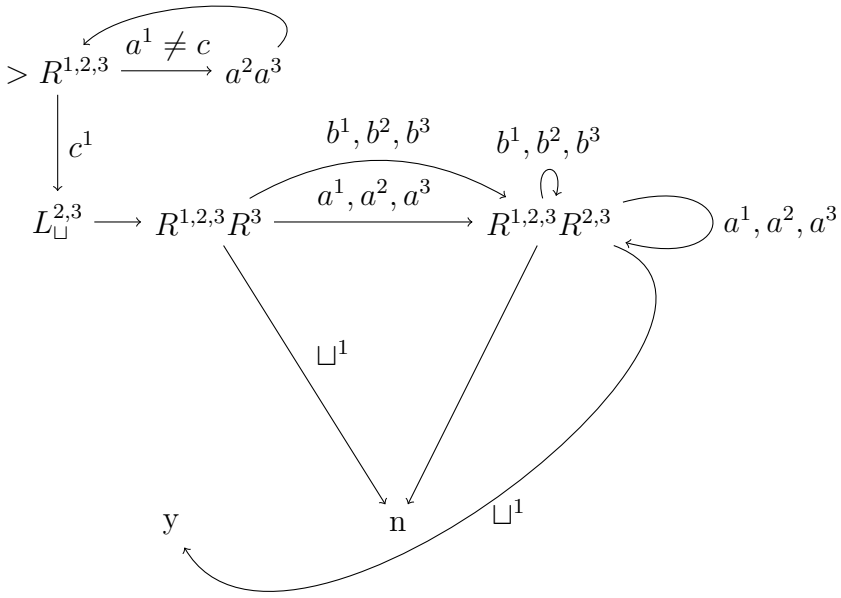
3-For a given string such that in form of a^*b^*, we can easily see that computation first will be like 1, and then goes to 2, and this doesn't change the number of a's and b's. so the machine's function behaves like this for any input $w \in \{a\}^*\{b\}^*$, M(w) will be $(s, \triangleright\sqcup w) \vdash_M^* (h, \triangleright\sqcup f(w))$ such that f(w) = w. so, in a form of $w \in a^*b^*$ f(w)=w.

4-And lastly any form of Σ_0^*, whenever a string like aba or ba machine finds the first a after first b's and mark the position with # and shifts this a to the next to first a's and after that it replaces # with a 'b'. Actually this machine rearranges the string such that first comes a's and then b's, in form of a*b* without changing number of a's and b's.

Hence we can say that M(w) will be $(s, \triangleright\sqcup w) \vdash_M^* (h, \triangleright\sqcup f(w))$ it will halt for any $w \in \Sigma_0$ and for this reason M(w) = f(w) so, M computes function f and $w \in \Sigma_0$ f(w) = $a^n b^m$ n = number of a's in w, m = number of b's in w.

Answer 4

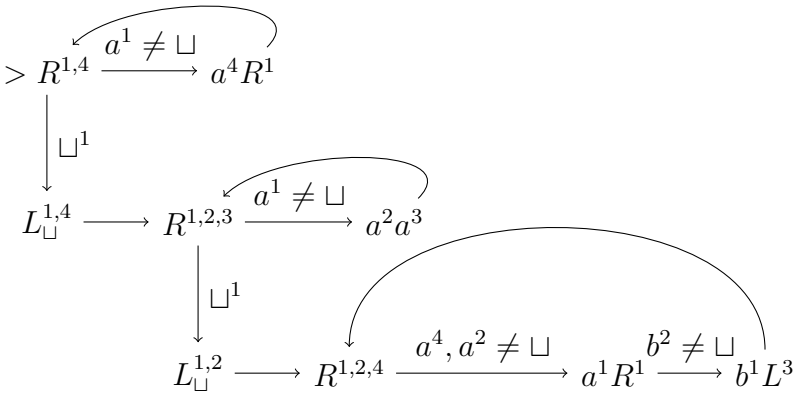
I will use 3-tape Turing Machine. and Here is my algoritm:
1-Take entire string to tape 1
2-Copy the w part of the string to tape 2 and 3 (easily can be copied until seeing c)
3-Move the head of first tape to beginning of u, move tape 2 and 3 to the beginning of w, and move right the tape 3
4- Compare each of them if they are equal move right tape 1 one square, move right tape 2 and 3 two right square, if different says it is not in the language, do this operation until tape 1 sees blank, if it sees blank halt and say it is in the language.
Here is my basic machine notation;



After $R^{1,2,3}R^{2,3}$ if first tape sees blank go to no transition means that u is empty string and empty string is not in the language.

Answer 5

I will use 4-tape Turing Machine, the first tape will contain v (at the end of operation), second and third tape will contain u, and fourth tape will contain our i range (for example for $|u| = 3$ i can be 1 and 2 and then operation ends.) Here is my algorithm: (initial configurations assumed $\triangleright \underline{\sqcup} w$)
1-Starting from the first character of the u count the number of symbols with passing 2 characters (what I mean by this,for instance, for a string abc count a and c not b. Like a pattern w(1),w(3)...), and write a in to the 4th tape as much as you count. When the operation is completed move both tape to the beginning
2-Copy the u into the second and third tape.
3-Start from beginning of the string for tape 1, 2 and 4, for tape 3 start from the end.
4-First write into first tape what you see on tape 1, then move right on first tape, and write what you see on tape 1 and move right on first tape and move right the head of tape4. This operation will continue until the head of 4th tape sees blank (like count)
Here is my basic machine notation,



Answer 6

a.

δ is the transition function $\delta : (K - H) \times \Sigma \to K \times ((\Sigma) \cup \{\downarrow\})$ such that
-For all $q \in K - H$, If $\delta(q, \triangleright) = (p, b)$ then $b \in \Sigma - \{\triangleright\}$
-For all $q \in K - H$, and $a \in \Sigma$, if $\sigma(q, a) = (p, b)$ then $b \neq \triangleright$.

b.

$\delta : (K - H) \times (\Sigma \cup e) \to K \times ((\Sigma) \cup \{\downarrow\})$

c.

Let $M = (K, \Sigma, \delta, s, H)$ be a Queue Turing machine and consider two configurations of M, $(q_1, a_1 w_1 u_1)$ and $(q_2, a_2 w_2 u_2)$, where $a_1 \in \Sigma$, $a_2 \in \Sigma$. Then

$$(q_l, a_1 w_1 u_1) \vdash_M (q_2, a_2 w_2 u_2)$$

if and only if , for some $b \in \Sigma \cup \{\downarrow\}$, $\delta(q_1, a_1) = (q_2, b)$, and either

1) b = $\downarrow$, $a_1 = a_2$, $w_1 = w_2 u_2$ and $u_1 \neq e$
2) b = $\downarrow$, and $w_1 u_1 = e$ then $a_2 = \triangleright$, $w_2 u_2 = e$
3) b $\in \Sigma - \{\triangleright\}$, $a_1 = a_2$, $w_1 u_1 = w_2$ and $u_2 = b$
4) b $\in \Sigma - \{\triangleright\}$, $a_1 = \triangleright$ and $w_1 u_1 = e$ then $a_2 = \triangleright a_2$ (a_2 is front)

(Assumption I already imposed restrictions on part a)

d

Firstly when the input comes we take the first character of string and in order to remember we goes to new state (for instance if a then go to q_a) and push a # to queue. This will indicate the

end. Then push all thing and while pushing change the states in order to remembers what did you pop (for instance if I popped 'a' at the beginning then I am in q_a state then for every character that I popped I will go to either q_{ab} or q_{aa} state c doesn't change the state) and for every character that I popped, I will push queue again. When I encounter the # that means input is end, and if I am in q_{aa} or q_{bb} states I will pop # and I will continue these process, and these process continue until the only character remains c, if I am in q_{ab} or q_{ba} states then machine will go to loop. If only character is c (I can control that adding a symbol to queue after popping c and whether it is in the front or not), then machine will halt. Algorithm works these way.

Answer 7

Before the explaining answer I assumed the insert-delete TM like two head but these two head is fixed at the beginning and front of the tape.

a.

Insert-delete TM defined as a quintuple $(K, \Sigma, \delta, s, H)$ where K is the set of states, Σ is a finite alphabet containing $\triangleright$ to denote end of tape, $s \in K$ is initial state and $H \subseteq K$ and $\delta : (K - H) \times (\Sigma \times \Sigma) \to K \times (\Sigma \cup \{\downarrow\}) \times \{f, r\})$, is the transition function where $\delta(q, (a, b)) = (p, (X, Y))$ for $q \in K - H$, $p \in K$, $a, b \in \Sigma$, $X \in \Sigma - \{\triangleright\}$, $Y \in \{f, r\}$ indicates that whenever the machine is in state q and there is a at the front position and b at the rear position, the machine enters the state p and takes the action (X,Y). X denotes the symbol from alphabet which will to be inserted or $\downarrow$ means delete from Y, and Y denotes the front (f) or rear (r). If $Y = f$ and $X \in \Sigma$ means insert X to the front of tape. If $Y = f$ and $X = \downarrow$ means delete front of tape. If $Y = r$ and $X \in \Sigma$ means insert X to the rear of tape. If $Y = r$ and $X = \downarrow$ means delete rear of tape. If rear and front is $\triangleright$ then it means empty tape no delete is allowed.

b.

A configuration of a insert-delete TM is a member of $K \times \Sigma^*$. Second element of tuple denotes tape in the string and first character of string is front and last character of string is rear. For instance $(q, ab \triangleright bb)$ means current state is q, front is a and rear is b.

c.

$(q, awb) \vdash_M (p, a'wb')$ if and only if for some a,b $\in \Sigma$, $\delta(q, (a, b)) = (p, (X, Y))$:

1- If Y = f and X = Σ then $a' = Xa, b' = b$
2- If Y = r and X = Σ then $a' = a', b' = bX$
3- If Y = f and X = $\downarrow$ then $a' = e, b' = b$
4- If Y = r and X = $\downarrow$ then $a' = a, b' = e$.

L(M) = $\{w | (q, w_1 \triangleright w_2) \vdash_M^* (f, w_a \triangleright w_b)\ f \in H\}$

d.

If we simulate it with the one tape 2-head machine we can say that it is equivalent the standart TM since we know that n-head machine is equivalent to standart TM. First head will show the front and second head will show the rear. And we can easily implement insert mechanism like that:
1- For insert to the front, firstly we can use right shift machine' operation and shift entire string to the right (w $\rightarrow \sqcup w$), and when we do this second head doesn't show the rear so after this operation one right square to the rear (second) head, and after this shift operation assume that first head will show the blank character ($\sqcup\underline{\sqcup} w$) then write the desired character to the first head position then with this way we can insert the desired character at the front.
2- For insert to the rear, firstly we can use left shift machine' operation and shift entire string to the left (w $\rightarrow w\sqcup$), and when we do this first head doesn't show the head so after this operation move the first(front) head to one left square, and after this shift operation assume that second head will show the blank character ($w\underline{\sqcup}$) then write the desired character to the second head position then with this way we can insert the desired character at the rear.
3-For delete from the front, first we can overwrite blank the the character which is shown by first head. Then we move first head to the right. Now current tape will be $\sqcup w$ so we will shift left to the entire string and (by assumption front will be the show first character of string after the shift) move second head to the one square left and operation will be completed.
4-For delete from the rear, first we can overwrite blank the the character which is shown by second head, and move second head to one square left.

Answer 8

From the lectures, we know that $L : \{a^{f(i)}$ where $f(i) = \dfrac{i(i+1)}{2}, i \in N\}$ has a grammar such that $G = (V, \Sigma, R, S)$ where

$$V = \{S, S', R, T, L, X, A, a\}$$
$$\Sigma = \{a\}$$
$$R = \{S \rightarrow S'R,$$
$$S' \rightarrow S'T|LT,$$
$$LT \rightarrow LX,$$
$$XT \rightarrow TXA,$$
$$AT \rightarrow TA,$$
$$AR \rightarrow RA,$$
$$A \rightarrow a,$$
$$LR \rightarrow e,$$
$$XR \rightarrow RA\}$$

In this grammar L and R denotes Left side and Right side of the string, and for generating a string for example i=2, obtain 2 T and apply other rules. It has a pattern like this 1+2+3+... In our example our pattern is 1+3+5+7+9... In this grammar pattern between plus number is determined by $XT \rightarrow TXA$, whenever X and T encounters they replace their position, and add 1 a. It works like that when i=2 it will two T and first T will be turned into X and encounters 1 T then it will add 1 A, at the and we have 2 X and 1 A which both will be turned into a after applying rules (1+2). If we just rewrite this rule such that $XT \rightarrow TXAA$ when they encounter, 2 a will be added, and with this way our pattern will be like 1+3+5... which is desired. Hence our grammar is for this question, $G = (V, \Sigma, R, S)$ where

$$V = \{S, S', R, T, L, X, A, a\}$$
$$\Sigma = \{a\}$$
$$R = \{S \rightarrow S'R,$$
$$S' \rightarrow S'T|L,$$
$$LT \rightarrow LX,$$
$$XT \rightarrow TXAA,$$
$$AT \rightarrow TA,$$
$$AR \rightarrow RA,$$
$$A \rightarrow a,$$
$$LR \rightarrow e,$$
$$XR \rightarrow RA\}$$

(I know you want to sum of n^2 but I stuck here, so I left here.)

Answer 9

By the theorem,A language L is recursively enumerable if and only if there is a Turing machine M that semidecides L. Hence according to this theorem,for L_1, L_2 and L_3, there are Turing Machines M_1, M_2, M_3 such that M_1 semidecides L_1, M_2 semidecides L_2, M_3 semidecides L_3. We can construct a Turing Machine M that semidecides given language L.For any given string w, nondeterministic M works as following and semidecides whether $w \in L$ or not:

1-Given a string w firstly our machine M, applies M_3 operations on this string, if it halts continue to next step,
2-Nondeterministically split given string w as $w = w_1w_2$,
3-Our machines applies M_1 operations on w_1,
4-Our machines applies M_2 operations on w_2,
5-If both 3 and 4 operation halts, then machine halts (string is accepted) otherwise go to step 2.