Student Information

Full Name : Onur Can TIRTIR
Id Number : 2099380

Answer 1

a.

For any language $L_k \in \Sigma^*$, $\overline{L_k} = L_1 \cup L_2 \cup \cdots \cup L_{k-1} \cup L_{k+1} \cup L_{k+2} \cdots \cup L_n$, which is recursively enumerable by following:

Proof 1 *Say we have two recursively enumerable languages L_1 and L_2 such that they can be recognized by the Turing Machines M_1 and M_2. If we construct a new Turing Machine M which recognizes their union, then we can say that whenever either of M_1 and M_2 recognizes w, M also accepts w and when both rejects w, M also rejects w, so these satisfy $M = M_1 \cup M_2$. It is easy to see that M runs as indicated above:*
For any w, run M_1 and M_2 parallel. If either TM accepts w, which is done in finitely many steps, then M accepts w. If both M_1 and M_2 rejects w then both will loop infinitely, which makes M also loop. So M never accepts w as M_1 and M_2 do.

So both L_k and $\overline{L}_k$ are recursively enumerable then by the theorem 5.7.1 in Chapter 5.7, *a language is recursive iff both it and its complement are recursively enumerable*, L_k is recursive.

b.

Since L_1 and $\overline{L}_1$ are recursively enumerable then by the theorem 5.7.1 in Chapter 5.7, L_1 is recursive. $L_2 \setminus L_1 = L_2 \cap \overline{L}_1$. As stated by the theorem 4.2.2 in Chapter 4.2, recursive languages are closed under complement, then $\overline{L}_1$ is also recursive. Now prove that recursive languages are also closed under intersection.

Proof 2 *Say we have two recursive languages L_1 and L_2 such that they can be recognized by the Turing Machines M_1 and M_2. If we construct a new Turing Machine M which recognizes their intersection, then we can say that whenever both of M_1 and M_2 accepts w, M also accepts w and whenever either of them rejects w, M also rejects w, so these satisfy $M = M_1 \cap M_2$. It is easy to see that M runs as indicated above:*
For any w, first run M_1. If M_1 accepts w, then run M_2. If M_2 acceps w, then w is recognized by M. If M_1 does reject w, or M_1 accepts but M_2 rejects w, then w cannot be recognized by M. Note that each rejection and acceptance is done in finitely many steps.

Hence $L_2 \cap \overline{L}_1$ is also recursive by the closure property indicated above.

If L_2 were recursively enumerable, $L_2 \cap \overline{L}_1$ will also be recursively enumerable. Here is the reasons:
1) $\overline{L}_1$ were recursive. This means it is actually recursively enumerable.
2) L_2 is recursively enumerable.
3) Since recursively enumerable languages are closed under intersection, then $L_2 \cap \overline{L}_1$ is also. Now prove recursively enumerable languages are closed under intersection.

Proof 3 *Say we have two recursively enumerable languages L_1 and L_2 such that they can be recognized by the Turing Machines M_1 and M_2. If we construct a new Turing Machine M which recognizes their intersection, then we can say that whenever both of M_1 and M_2 recognizes w, M also accepts w and when either of them rejects w, M also rejects w, so these satisfy $M = M_1 \cap M_2$. It is easy to see that M runs as indicated above:*
For any w, run M_1 and M_2 sequentially. If either TM accepts w, which is done in finitely many steps, then M accepts w. If either of M_1 and M_2 rejects w then M will loop infinitely. So M never accepts.

c.

First give a little explanation recursive languages are closed under concatanetion.

Proof 4 *Say we have two recursive languages L_1 and L_2 such that they can be recognized by the Turing Machines M_1 and M_2. As illustrated in 4-c, TMs can be concataneted. Hence TMs are closed under concatanetion.*

$L_N = L_1 \setminus (L_2 L_3) = L_1 \cap \overline{L_2 L_3}$, which is also a recursive language by the closure properties given in this and above parts.
For any function f's computablity means that there exists a M such that M computes f for all $w \in \Sigma_0^*$, i.e $M(w) = f(w)$. That is to say, for any string from Σ_0^*, M **eventually halts on input w**. The bold words says, by the definition of decidibility, M is decidable. Hence f is also decidable. If f is a bijective mapping from some strings to another strings, then all the things we just said for such a function f are all valid for its inverse because there is no reason for us not to be able to compute the outcome of f^{-1} properly as f was a bijection.

Now take the specific function f^{-1} given in the question at hand. L_N is recursive. That means there is a M that decides it. Since f^{-1} is decidable, then we can say that the TM M computing f^{-1} can decide on L_n.
So $f^{-1}(L_N)$ is Turing-decidable.

Answer 2

a.

$G = \{V, \Sigma, R, S\}$
$V = \{S, L, D, R, a\}$

$\Sigma = \{a\}$

$R = \{S \to LaD,$
$D \to RD,$
$D \to e,$
$aR \to Raa,$
$LR \to L,$
$L \to e\}$

b.

$G = \{V, \Sigma, R, S'\}$
$V = \{S', S, 1, 2, x, y, X, Y, L, E_x, E_y, E_{1,2}\}$
$\Sigma = \{1, 2, x, y\}$

$R = \{S' \to LS,$
$S \to E_y,$
$S \to 1YS,$
$S \to 2XS,$
$X1 \to 1X,$
$Y1 \to 1Y,$
$X2 \to 2X,$
$Y2 \to 2Y,$
$YX \to XY,$
$YE_y \to E_y y,$
$YE_y \to E_x y,$
$XE_x \to E_x x,$
$XE_x \to E_{1,2} x,$
$1E_{1,2} \to E_{1,2}1,$
$2E_{1,2} \to E_{1,2}2,$
$LE_{1,2} \to e,$
$E_y \to E_x,$
$E_y \to E_{1,2},$
$E_x \to E_{1,2}\}$

Answer 3

Note: If there is no go to condition, directives given below will be followed sequentially

a.

0-) If the input is not in the format $a^p b^r c^t x^k y^l z^m$, where $p \geq 1, r \geq 1, t \geq 1, k \geq 1, l \geq 1, m \geq 1$, then reject, else move head to leftmost symbol

A1-) Convert a to A.

A2-) Move head to right to find leftmost b in tape. If there is no b, then reject.

A3-) Convert b to B.

A4-) Move head right to find leftmost c in tape. If there is no c, then reject.

A5-) Convert c to C

A6-) Try to find leftmost b by moving head to left. If found any then go to step A3.

A7-) Move head left sequentially and convert all Bs to b while moving left.

A8-) Try to find leftmost a by moving head to left. If found one a then go to step A1.

A9-) Try to find leftmost c by moving head to right. If found then reject.

B1-) Find leftmost x by moving head to right.

B2-) Convert x to X.

B3-) Try to find leftmost y by moving head to right. If there is no y then reject.

B4-) Convert y to Y.

B5-) Try to find leftmost z by moving head to right. If there is no z then reject.

B6-) Convert z to Z.

B7-) Try to find leftmost x by moving head to left. If found then go to step B2.

B8-) Try to find leftmost y by moving head to right. If found then reject.

B9-) Try to find leftmost z by moving head to right. If found then reject.

H-) Accept.

b.

I will use three tapes, tape1 will keep the input and, tape2 and tape3 will be used to keep the copy of the strings to be compared. In this subsection, head implies the head of tape1 unless otherwise is indicated.

0-) Find first dollar symbol by moving head to right until finding and mark it as X.

1-) Move back to first symbol in tape and then check if there is any Y symbol by moving to right.

2-) If Y is found then go to step A1, else go to step B1.

A1-) Convert current Y to dollar. Try to find next dollar symbol after Y by moving head to right. If there is then mark it Y and go to step Comp1.

A2-) Move back to first symbol in tape and find X symbol by moving right. Make it dollar.

A3-) Try to find next dollar symbol after current dollar symbol by moving head to right.

A4-) If dollar is found then make it X and go to step 1, else accept.

B1-) Move back to first symbol in tape and try to find the leftmost dollar symbol by moving head to right.

B2-) If dollar is found then make it Y and go to step Comp1, else accept.

Comp1-) Copy the string coming after X to TM1 excluding ending dollar symbol and copy the string coming after Y to TM2 excluding ending dollar symbol.

Comp2-) Move heads of TM1 and TM2 in paralled until one of the heads reads $\sqcup$. If both heads read same symbol except $\sqcup$ then move both heads just one right. Else if they read different symbols then go to step 1. Else if both reads $\sqcup$ then reject.

Answer 4

a.

$M = \{K, \Sigma, \delta, s, H\}$ where:
$K = \{s, q_0, q_1, q_2, q_3, q_4, q_5, h\}$,
$\Sigma = \{\sqcup, 1, x, y, \triangleright\}$,
$H = \{h\}$, and δ is defined such that:

$\delta(s, \sqcup) = (s, \rightarrow)$
$\delta(s, 1) = (q_0, \rightarrow)$
$\delta(q_0, 1) = (q_0, \rightarrow)$
$\delta(q_0, \sqcup) = (q_1, y)$
$\delta(q_1, y) = (q_1, \leftarrow)$
$\delta(q_1, 1) = (q_1, \leftarrow)$
$\delta(q_1, \sqcup) = (q_2, \rightarrow)$
$\delta(q_1, x) = (q_2, \rightarrow)$
$\delta(q_2, 1) = (q_3, x)$
$\delta(q_3, x) = (q_3, \rightarrow)$
$\delta(q_3, y) = (q_3, \rightarrow)$
$\delta(q_3, 1) = (q_3, \rightarrow)$
$\delta(q_3, \sqcup) = (q_1, 1)$
$\delta(q_2, y) = (q_5, 1)$
$\delta(q_5, 1) = (q_4, \leftarrow)$
$\delta(q_4, x) = (q_5, 1)$
$\delta(q_4, \sqcup) = (h, \sqcup)$

Also note that $\forall q \in K - H,\ \delta(q, \triangleright) = (p, \rightarrow)$.

Our TM's behaviour will be like that:
1-) Find first blank symbol after string and put a y here to mark.
2-) While there exists at least one 1 before y, convert leftmost 1 to x and put another 1 after symbol y.
3-) Convert y to 1.
4-) While there exists at least one x before y, convert rightmost x to 1 and move left.
5-) Halt when arrived to blank symbol right after $\triangleright$.

b.

$M = \{K, \Sigma, \delta, s, H\}$ where:
$K = \{s, q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, h\}$,
$\Sigma = \{\sqcup, 1, x, \triangleright\}$,
$H = \{h\}$, and δ is defined such that:

$\delta(s, \sqcup) = (s, \rightarrow)$
$\delta(s, 1) = (q_0, 1)$
$\delta(q_0, 1) = (q_0, x)$
$\delta(q_0, x) = (q_1, \rightarrow)$
$\delta(q_1, 1) = (q_1, \rightarrow)$
$\delta(q_1, \sqcup) = (q_2, \leftarrow)$
$\delta(q_2, 1) = (q_3, \sqcup)$
$\delta(q_3, \sqcup) = (q_3, \leftarrow)$
$\delta(q_3, 1) = (q_4, \sqcup)$
$\delta(q_4, \sqcup) = (q_4, \leftarrow)$
$\delta(q_4, 1) = (q_5, \sqcup)$
$\delta(q_5, \sqcup) = (q_5, \leftarrow)$
$\delta(q_5, 1) = (q_6, \sqcup)$
$\delta(q_6, \sqcup) = (q_6, \leftarrow)$
$\delta(q_6, 1) = (q_7, 1)$
$\delta(q_7, 1) = (q_7, \leftarrow)$
$\delta(q_7, x) = (q_0, \rightarrow)$
$\delta(q_2, x) = (q_8, x)$
$\delta(q_3, x) = (q_8, x)$
$\delta(q_4, x) = (q_8, x)$
$\delta(q_5, x) = (q_8, x)$
$\delta(q_6, x) = (q_8, x)$
$\delta(q_8, 1) = (q_8, \leftarrow)$
$\delta(q_8, x) = (q_8, 1)$
$\delta(q_8, \sqcup) = (h, \sqcup)$

Also note that $\forall q \in K - H,\ \delta(q, \triangleright) = (p, \rightarrow)$.

Our TM's behaviour will be like that:
1-) Find leftmost 1 and mark it as x.
2-) Start from the rightmost 1 in tape and try to delete(convert to $\sqcup$) 4 rightmost 1s by traversing the tape starting from the rightmost 1 and goint to left.
3-) If you succeded and have still 1s in tape(checking that the symbol left to current head position is 1 or x) then go to step 1. Else you have 0, 1, 2 or 3 1s and you delete them.
4-) Convert all the x's in tape 1 by traversing the tape right to left.

c.

Rename M for TM in part a as M_1, K as K_1, δ as δ_1 and H as H_1. Add subscript $_1$ for all $q \in K$. Redefine all the definitions of δ for δ_1.

Rename M for TM in part a as M_2, K as K_2, δ as δ_2 and H as H_2. Add subscript $_2$ for all $q \in K$. Redefine all the definitions of δ for δ_2. Also append the y symbol to alphabet of M_2 to make it equivalent to alphabet of TM in part a, so as to make it consistent for our new TM M_3, which will be the answer of this part.

Then:

$\Sigma = \{\sqcup, 1, x, y, \triangleright\}$,

$M_1 = \{K_1, \Sigma, \delta_1, s_1, H_1\}$ where:
$K_1 = \{s_1, q_{10}, q_{11}, q_{12}, q_{13}, q_{14}, q_{15}, h_1\}$,
$H_1 = \{h_1\}$, and δ_1 is defined such that:

$\delta_1(s_1, \sqcup) = (s_1, \rightarrow)$
$\delta_1(s_1, 1) = (q_{10}, \rightarrow)$
$\delta_1(q_{10}, 1) = (q_{10}, \rightarrow)$
$\delta_1(q_{10}, \sqcup) = (q_{11}, y)$
$\delta_1(q_{11}, y) = (q_{11}, \leftarrow)$
$\delta_1(q_{11}, 1) = (q_{11}, \leftarrow)$
$\delta_1(q_{11}, \sqcup) = (q_{12}, \rightarrow)$
$\delta_1(q_{11}, x) = (q_{12}, \rightarrow)$
$\delta_1(q_{12}, 1) = (q_{13}, x)$
$\delta_1(q_{13}, x) = (q_{13}, \rightarrow)$
$\delta_1(q_{13}, y) = (q_{13}, \rightarrow)$
$\delta_1(q_{13}, 1) = (q_{13}, \rightarrow)$
$\delta_1(q_{13}, \sqcup) = (q_{11}, 1)$
$\delta_1(q_{12}, y) = (q_{15}, 1)$
$\delta_1(q_{15}, 1) = (q_{14}, \leftarrow)$
$\delta_1(q_{14}, x) = (q_{15}, 1)$
$\delta_1(q_{14}, \sqcup) = (h_1, \sqcup)$

Also note that $\forall q \in K_1 - H_1,\ \delta_1(q, \triangleright) = (p, \rightarrow)$.

$M_2 = \{K_2, \Sigma, \delta_2, s_2, H_2\}$ where:
$K_2 = \{s_2, q_{20}, q_{21}, q_{22}, q_{23}, q_{24}, q_{25}, q_{26}, q_{27}, q_{28}, h_2\}$,
$H_2 = \{h_2\}$, and δ_2 is defined such that:

$\delta_2(s_2, \sqcup) = (s_2, \rightarrow)$
$\delta_2(s_2, 1) = (q_{20}, 1)$

$\delta_2(q_{20}, 1) = (q_{20}, x)$
$\delta_2(q_{20}, x) = (q_{21}, \rightarrow)$
$\delta_2(q_{21}, 1) = (q_{21}, \rightarrow)$
$\delta_2(q_{21}, \sqcup) = (q_{22}, \leftarrow)$
$\delta_2(q_{22}, 1) = (q_{23}, \sqcup)$
$\delta_2(q_{23}, \sqcup) = (q_{23}, \leftarrow)$
$\delta_2(q_{23}, 1) = (q_{24}, \sqcup)$
$\delta_2(q_{24}, \sqcup) = (q_{24}, \leftarrow)$
$\delta_2(q_{24}, 1) = (q_{25}, \sqcup)$
$\delta_2(q_{25}, \sqcup) = (q_{25}, \leftarrow)$
$\delta_2(q_{25}, 1) = (q_{26}, \sqcup)$
$\delta_2(q_{26}, \sqcup) = (q_{26}, \leftarrow)$
$\delta_2(q_{26}, 1) = (q_{27}, 1)$
$\delta_2(q_{27}, 1) = (q_{27}, \leftarrow)$
$\delta_2(q_{27}, x) = (q_{20}, \rightarrow)$
$\delta_2(q_{22}, x) = (q_{28}, x)$
$\delta_2(q_{23}, x) = (q_{28}, x)$
$\delta_2(q_{24}, x) = (q_{28}, x)$
$\delta_2(q_{25}, x) = (q_{28}, x)$
$\delta_2(q_{26}, x) = (q_{28}, x)$
$\delta_2(q_{28}, 1) = (q_{28}, \leftarrow)$
$\delta_2(q_{28}, x) = (q_{28}, 1)$
$\delta_2(q_{28}, \sqcup) = (h_2, \sqcup)$

Also note that $\forall q \in K_2 - H_2,\ \delta_2(q, \triangleright) = (p, \rightarrow)$.

And finally our answer is:

$M_3 = \{K_1 \cup K_2, \Sigma, \delta_3, s_2, H_1\}$ and define δ_3 such that:

$\forall \sigma \in \Sigma$:
1-) $If\ q \in K_1 - H_1,\ then\ \delta_3(q, \sigma) = \delta_1(q, \sigma)$
2-) $If\ q \in K_2 - H_2,\ then\ \delta_3(q, \sigma) = \delta_2(q, \sigma)$
3-) $If\ q \in H_2,\ then\ \delta_3(q, \sqcup) = (s_1, \sqcup)$.

Our TM M_3 will work such that: first M_2 runs and halts in the blank symbol just right of $\triangleright$, then M_1 runs and it also halts in that blank symbol and hence computation is done.

Answer 5

a.

Say we have a Turing Machine M_1, counting the number of states of M ,say s, and simulating M in a tape indicated in initial configuration up to s moves. M_1 will accept if M's simualation writes a non-blank symbol. We also know if $M \notin L$ then $M \notin L(M_1)$. For another case, assume $M \in L$ and M will write first non-blank symbol after s' transitions and these transitions should definitely be startting in s' states, In the opposite key, the mean s' will contradict. So we can conclude that $s' < s$ and M' will accept. Thus L is decidable. Note that this does not conflicts with halting problem.

b.

A decider, by definiton, should accept the strings in the language and rejects the strings not in the language and halt for both cases. Say we have a language L which does not accept any string $w \in \Sigma_0^*$. So we can say that our TM M is just looping whenever an input w s.t $w \in \Sigma_0^*$ is provided. But L is decidable since we can just define another TM M' such that it, again, does not accept any possible input w,however, this property is not provided by $infinite\ looping$, so M' decides L and hence L is decidable.

c.

We do not know if D_1 accepts or rejects $w \notin L$. Also we do not know if D_2 accepts or rejects $w \in L$. Then there may exists such ambigious situations:

For example, as we stated above, D_2 may reject the strings $w \in L$, but D_1 will certainly accept these strings. So we do not know these kind of words are in the language or not. So some of these language Ls may not be decidable.

Answer 6

a.

In part b, we easily constructed a deterministic PDA since we know from which set w came by the help of the symbols x and y. However in part a, the absence of these strings may create some problems:
For instance, to recognize a string like $a^n b^{2n}$, the rule is easy: for every a push aa to stack then pop a for every b in w. Since we do not know from which set w came, it is a must to include this rule in Δ. However, in case of a string $a^n b^{n+m} c^m$, where $m \neq n$, if we are to apply the same rule, then some a will be left in stack. Another example is that if we use the rules for the strings from first set, then we fail to accept the strings from second set because there is no c in these strings. In both examples, it was a must to use the rules indicated to construct our TM properly but we

could not.
Then only solution is to put both transitions to Δ and benefit from backtracking, which makes our automaton a nondeterministic PDA. Then the language given in this part is not a deterministic CFG.

b.

We can write a deterministic pushdown automaton for L such that:
$M = (K, \{a, b, c, x, y\}, \{a, b, c\}, \Delta, s, f)$ where

$\Delta = \{$
$((s, x, e), (q_0, x)),$
$((q_0, a, e), (q_1, a)),$
$((q_1, a, e), (q_1, a)),$
$((q_1, b, a), (q_1, e)),$
$((q_1, e, x), (q_2, x)),$
$((q_2, b, e), (q_2, b)),$
$((q_2, c, b), (q_2, e)),$
$((q_2, e, x), (f, e)),$
$((s, y, e), (q_3, y)),$
$((q_3, a, e), (q_4, aa)),$
$((q_4, a, e), (q_4, aa)),$
$((q_4, b, a), (q_4, e)),$
$((q_4, e, y), (f, e))$
$\}$

Since there is no two distinct compatible transitions in M, M is deterministic. Then $L = L(M)$ is a CFG.