Automata Excercises

Theory of Automata (COMSATS University Islamabad)

Chapter 2

PROBLEMS

1. Consider the language **S*,** where S = {a, *b}.*
How many words does this language have of length 2? of length 3? of length n?

Step 1 of 2

Given that, Language S = {a b}

So, the language S* with different words with the language S and different lengths will be

$S^* = \{\Lambda, a, b, aa, ab, ba, bb, aaa, aab, aba, \ldots.\}$

Step 2 of 2

The language S contains n (here n = 2 i.e., a, b) words

If we consider L to be the length of the words from the language S*, we can derive n^L combinations with the words of language S.

Thus to get words of length 2, L becomes 2.

Total words of Length 2 $= n^L$

$= 2^2$

$= 4$ which were nothing but {aa, ab, ba, bb}

Similarly, to get words of length 3, L becomes 3.

Total words of Length 3 $= n^L$

$= 2^3$

$= 8$ which were nothing but {aaa, aab, aba … …}

In general, total words of Length L $= n^L$

2. Consider the language **S*,** where S = *{aa, b}.*
How many words does this language have of length 4? of length 5? of length 6? What can be said in general?

Step 1 of 3

Generating words of the language S^*

Language S = {*aa b*} then, the language

S^* for string of length 4: There are 5 strings $\{aaaa,\ aabb,\ baab,\ bbaa,\ bbbb\}$

S^* for string of length 5: There are 8 strings

$\{aaaab,\ aabaa,\ bbbaa,\ aabbb,\ bbaab,\ baabb,\ baaaa,\ bbbbb\}$

S^* for string of length 6: There are 13 strings

$$\left\{\begin{array}{l} aaaaaa,\ aaaabb,\ bbaaaa,\ aabaab,\ baabaa,\ aabbaa,\ bbaabb,\ aabbbb, \\ bbbbaa, baaaab,\ bbbbbb,\ baabbb,\ bbbaab \end{array}\right\}$$

Step 2 of 3

To figure out the general case, for strings of length n, we can observe that the length n=7. Since n is odd there must be at least one *b* in the word.

So possibilities (1) 3 aa's and 1 b – $C(6,1)$

(2) 2 aa's and 3 b's – $C(4,3)$

(3) 5 b's and 1 aa – $C(5,2)$

(4) 7 b's and 0 aa's – $C(7,0)$

Number of different ways to place the *b's* and adds these all

Number of r-combinations of a set with n elements, where n is non-negative $0 \le r \le n$

$$C(n,r) = \frac{n!}{r!(n-r)!}$$

Calculate for $C(4,3)$.

$$\begin{aligned} C(4,3) &= \frac{4!}{3!(4-3)!} \\ &= \frac{4\times 3\times 2\times 1}{(3\times 2\times 1)\times 1} \\ &= 4 \end{aligned}$$

Calculate for $C(6,1)$.

$$C(6,1)=\frac{6!}{1!(6-1)!}$$
$$=\frac{6\times5\times4\times3\times2\times1}{(5\times4\times3\times2\times1)\times1}$$
$$=6$$

Step 3 of 3

Calculate for $C(5,2)$.

$$C(5,2)=\frac{5!}{2!(5-2)!}$$
$$=\frac{5\times4\times3\times2\times1}{2\times1\times(3\times2\times1)}$$
$$=\frac{5\times4}{2}$$
$$=10$$

Calculate for $C(7,0)$.

$$C(7,0)=\frac{7!}{(7-0)!}$$
$$=\frac{7!}{7!}$$
$$=1$$

Hence

$$C(4,3)+C(5,2)+C(6,1)+C(7,0)=4+10+6+1$$

= 21

3. Consider the language S*, where S = *{ab, ba}*. Write out all the words in S* that have seven or fewer letters. Can any word in this language contain the substrings aaa or *bbb?*

Step 1 of 3

Language $S=\{ab\ ba\}$, then the language

$S^*=\{\Lambda, ab, ba, abab, abba, baab, baba, ababab, ababba, abbaab, abbaba, baabab,$

$baabba, babaab, babab a,.....\}$

Step 2 of 3

The language does not contain the substrings "aaa" and "bbb", because the language set S is defined over {ab ba} which does not contain 3 or more consecutive a's or b's

Step 3 of 3

The smallest word that is not in this language is either 'aa' or 'bb'.

4. Consider the language S*, where S = *{a ab* bal. Is the string *(abbba)* a word in this language? Write out all the words in this language with seven or fewer letters. What is another way in which to describe the words in this language? Be careful, this is not simply the language of all words without bbb.

Step 1 of 4

Language $S=\{a\ ab\ ba\}$ then the language

$S^*=\{\Lambda, a, aa, ab, ba, aab, aba,baa, abab, abba, baab, baba ...\}$

Step 2 of 4

No, the "abbba" is not a word in this language

Step 3 of 4

$S^* = \{\Lambda$, a, aa ,ab, ba ,aaa, aab, aba, baa, aaaa, aaab, aaba, abaa ,abab, abba, baaa, baab, baba, aaaaa, aaaab, aaaba, aabaa, aabab, aabba, abaaa, abaab, ababa, abaaa, abaab ,ababa, abbaa, baaaa, baaab, baaba, babaa, aaaaaa, aaaaab, aaaaba, aaabaa, aaabab, aaabba, aabaaa, aabaab, aababa, aabaaa, aabaab, aababa, aabbaa, abaaaa, abaaab, abaaba, ababaa abaaaa, abaaab, abaaba, ababaa, ababab, ababba, abbaaa ,abbaab, abbaba, baaaaa, baaaab, baaaba, baabaa, baabab, baabba, babaaa, babaab, bababa,.....}

Provide feedback (0)

Step 4 of 4

Another way:

Another way to represent the words of this language is

$$(a+ab+ba)^*$$

That means the words that contain either 'a' or 'ab' or 'ba' and the strings of their combination.

5. Consider the language **S*,** where S = {aa *aba baa}.* Show that the words *aabaa*, *baaabaaa*, and *baaaaababaaaa*a re all in this language. Can any word in this language be interpreted as a string of elements from S in two different ways? Can any word in this language have an odd total number of a's?

Step 1 of 3

Language $S = \{\text{aa aba baa}\}$ then, the language

$S^* = \{\Lambda$, aa, aba, baa, aaaa, aaaba, aabaa, abaaa, abaaba, ababaa, baaaa, baaaba, baabaa, aaaaaa, aaaaaba, aaaabaa, aaabaaa, aaabaaba, aaababaa, aabaaaa, aabaaaba, aabaabaa, abaaaaa, abaaaaba, abaaabaa, abaabaaa, abaabaaba, abaababaa, ababaaaa, ababaaaba, ababaabaa, baaaaaa, baaaaaba, baaaabaa, baaabaaa, baaabaaba, baaababaa, baabaaaa, baabaaaba, baabaabaa .....}

Step 2 of 3

The word aabaa is divided into (aa)(baa) which belongs to Language S, so the word (aabaa) is in this language .

The word baaabaaa is divided into (baa)(aba)(aa) which belongs to Language S,so the word(baaabaaa) is in this language.

The word baaaaababaaaa is divided into (baa)(aa)(aba)(baa)(aa) which belongs to Language S ,so the word (baaaaababaaaa) is in this language.

Provide feedback (0)

Step 3 of 3

No interpretation for string of elements from S in two different ways.

No odd of total number of a's in this language because each word in the given language contains even number of a's. So, by concatenating these words we will not get odd number of a's

6. Consider the language **S*** where S = {xx xxx}. In how many ways can **x"9** be written as the product of words in **S?** This means: How many different factorizations are there of x19 into xx and xxx?

Step 1 of 2

Product of words and factorizations

Factorization is the different possible ways of representing a number with the given set of words.

For example, x^{19} can be factorized into $\{x^1, x^{18}\}, \{x^2, x^{17}\}, ..., \{x^9, x^{10}\}$.

Consider the language $S^* = \{\Lambda\ xx\ xxx\ xxxx ...\}$

where $S = \{xx\ xxx\}$ is the set of words in the language.

Calculate number of ways representing the product of words for x^{19}.

The lengths of the words in the language are $xx = x^2 = 2$ and $xxx = x^3 = 3$.

This can also be understood as, factors of 19 with multiples of 2 and 3, or

$$(x^2)^a + (x^3)^b = x^{19}$$

Then, the possible sets are {16, 3}, {10, 9} and {4, 15}

1. 8 number of xx and 1 xxx

$$(x^2)^8 + (x^3)^1 = x^{19}$$

$$x^{16} + x^3 = x^{19}$$

The total number of words that can be formed using

8 words of length 2 + 1 word of length 3= (8+1) = 9 .....(1)

2. 5 number of xx and 3 number of xxx

$$(x^2)^5 + (x^3)^3 = x^{19}$$

$$x^{10} + x^9 = x^{19}$$

The total number of words that can be formed using

5 words of length 2 + 3 word of length 3= (5+3) = 8 .....(2)

3. 2 number of xx and 5 number of xxx

$$(x^2)^2 + (x^3)^5 = x^{19}$$

$$x^4 + x^{15} = x^{19}$$

The total number of words that can be formed using

2 words of length 2 + 5 word of length 3= (2+5) = 7 .....(3)

Step 2 of 2

Calculate the total number of ways or factorizations, representing x^{19} .

Use the formula of combinations to find the total number of possible different combinations of the given word:

$$C(n,r)=\frac{n!}{r!(n-r)!}$$

This can be interpreted as the factorial of total number of factors possible divided by the number of repetitions in each type.

From the equation (1) calculate for 9,

$$\begin{aligned}C(9,1)&=\frac{9!}{1!(9-1)!}\\&=\frac{9!}{8!}\\&=9\,\text{combinations}\end{aligned}$$

From the equation (2) calculate for 8,

$$\begin{aligned}C(8,3)&=\frac{8!}{3!(8-3)!}\\&=\frac{8\times7\times6\times5\times4\times3\times2\times1}{(3\times2\times1)(5\times4\times3\times2\times1)}\\&=56\,\text{combinations}\end{aligned}$$

From the equation (3) calculate for 7,

$$\begin{aligned}C(8,3)&=\frac{7!}{2!(7-2)!}\\&=\frac{7\times6\times5\times4\times3\times2\times1}{(2\times1)(5\times4\times3\times2\times1)}\\&=21\,\text{combinations}\end{aligned}$$

Calculate total combinations from equation solvation,

$$\begin{aligned}&=9+56+21\\&=86\end{aligned}$$

Hence, total number of combinations possible for x^{19} is $\boxed{86}$.

7. (i) Prove that if x is in PALINDROME then so is x" for any n.

Step 1 of 4

I) It is true for n = 1 by assumption x^n is in palindrome.

Assume that it is true for n -1, namely, x^{n-1} is in palindrome and $\left(x^{n-1}\right)^R = x^{n-1}$

We know that $(xy)^R = y^R x^R$ 'R' means reverse of string.

$$\left(x^n\right)^R = \left(x^{n-1}x\right)^R$$

$$= x^R\left(x^{n-1}\right)^R$$

$$= xx^{n-1}$$

$$= x^n$$

$\therefore$ x^n is in palindrome

(ii) Prove that if y3 is in PALINDROME then so is y.

Step 2 of 4

II) PALINDROME can be defined as

PALINDROME= {$\wedge$, and all strings *x* such that *reverse(x)=x*}

Let y^3 be in PALINDROME then

Reverse $\left(y^3\right)$ = $\left(\text{reverse } (y)\right)^3$

Reverse $\left(y^3\right)$ = reverse (y). reverse (y). reverse (y)

Since y^3 and y have the same lengths,

$y \in$PALINDROME

(iii) Prove that if z' is in PALINDROME for some n (greater than 0) then z itself is also.

Step 3 of 4

III) Let W = Z^n

If n = 1, then the result is trivial

Suppose n >1, then z^n= reverse (z^n)

z^n = (reverse(z))^n

∴ z .....z = reverse (z) ..... reverse (z)

But the z and reverse (z) have the length are same

∴ z = reverse (z)

(iv) Prove that PALINDROME has as many words of length 4 as it does of length 3.

Step 4 of 4

IV) List of words the palindromes of length 3 are aaa aba bab bbb.

List of words the palindromes of length 4 are aaaa abba baab bbbb

V) PALINDROME of length 2n, where (n = 1) {aa bb}

PALINDROME of length 2n, where (n = 2) {aaaa abba baab bbbb}

PALINDROME of length 2n – 1, where (n = 1) {a b}

PALINDROME of length 2n – 1, where (n = 2) {aaa aba bab bbb}

Hence, PALINDROME of length 2n and 2n-1 have 2 words each.

(v) Prove that PALINDROME has as many words of length 2n as it has of length ***2n***- 1.

8. Show that if the concatenation of two words (neither A) in PALINDROME is also a word in PALINDROME then both words-are powers of some other word; that is, if x and y and xy are all in PALINDROME then there is a word z such that x = z' and y = ***zm*** for some integers n and *m* (maybe n or *m* = 1).

Step 1 of 1

1. If the lengths of x and y are equal, then there exists a word z that equals either x or y. (m and n would equal 1)

2. If the length of y is equal to n * length(x), then there exists a word z that equals y, and each string of length(x) / n would equal z.

3. If the length of x is less than y, the length of x or y is not 0, and the leftmost portion of length x from string y meets the criteria of 1., then make y equal to the unused rightmost portion of string y. and repeat steps 1 and 2.

4. Only words produced by 1 and 2 are allowed in the language, such that if r() = the reverse of some variable, then z = r(z).

9. Let S = *{ab, bb}* and let *T = {ab, bb, bbbb}.* Show that S* = T*. What principle does this illustrate?

Step 1 of 3

(i) Both are Symmetry(S*=T*) by the following proof

S={ab bb} then S*={∧,ab,bb,abab,abbb,bbab,bbbb,bbbbab.....}

T={ab bb bbbb} then T*={∧,ab,bb,abab,abbb,bbab,bbbb,bbbbab.....}

Since, $S^* \subseteq T^* \& T^* \subseteq S^*$

∴ S*=T*

Step 2 of 3

(ii) S = {ab bb} then S* = { ^, ab, bb, abab, bbbb,.....}

T = {ab bb bbb} then T* = { ^, ab, bb, bbb, abbb, bbbab,.....}

Here $S^* \subseteq T^*$ but $T^* \not\subset S^*$ because S* does not contain the word 'bbb', which means every word in S is present in T but, every word in T is not present in S.

Thus,

$S^* \neq T^*$

Provide feedback (0)

Step 3 of 3

(iii) The principle illustrated in the second part is 'unfortunate situation'. An unfortunate situation arises when a word or words are present in one language (T) but not in the other(S).

10. How does the situation in Problem 9 change if we replace the operator * with the operator + as defined in this chapter? Note the language **S'** means the same as S* but does not allow the "concatenation of no words" of S.

Step 1 of 3

If we replace the operator * with the operator + in problem 9 in this chapter, then

I) Both are Symmetry($S^+=T^+$) by the following proof:

S={ab bb} then S^+ ={ab, bb, abab, abbb, bbab, bbbb, bbbbab.....}

T={ab bb bbbb} then T^+={ab, bb, abab, abbb, bbab, bbbb, bbbbab.....}

Since, $S^+ \subseteq T^+ \& T^+ \subseteq S^+$

$\therefore S^+=T^+$

Here S^+ without $\wedge$,so this is does not allow the concatenation of no words of S.

Step 2 of 3

II) S = {ab bb} then S^+ = { ab, bb, abab, bbbb,.....}

T = {ab bb bbb} then T^+ = {ab, bb, bbb, abbb, bbbab,.....}

Here $S^+ \subseteq T^+$ but $T^+ \not\subset S^+$ because S^+ does not contain the word with 'bbb'

Thus, $S^+ \neq T^+$

Step 3 of 3

III) This is called unfortunate situation from part II.

---

12. Prove that for all sets **S,**
(i) (S+)* =**(S*)***
(ii) (S+)' =**S**
(iii) Is (S*)+ = (S+)* for all sets **S?**

Step 1 of 3

Proof:

i) $(S^+)^* = (S^*)^*$

Every factor from S+ is made up of factors from S, excluding Λ if it's not in the set.
Every factor from $(S^+)^*$ is made up of factors from S^+, including Λ.

Every factor from S* is made up of factors from S, including Λ. Every factor from (S*)* is made up of factors from S*, including Λ.

Therefore, every word in $(S^+)^*$ and (S*)* is made up of factors from S. Therefore, every word in $(S^+)^*$ is also a word in (S*)*.

Step 2 of 3

ii) (S+)+ = S^+

Every factor from S^+ is made up of factors from S, excluding Λ if it's not in the set.
Every factor from (S+)+ is made up of factors from S^+, excluding Λ again.

Therefore, every word in (S+)+ and S^+ is made up of factors from S excluding Λ.
Therefore, every word in (S+)+ is also a word in S^+.

Step 3 of 3

iii) Is $(S^*)^+ = (S^+)^*$ for all sets S?

Every factor from S* is made up of factors from S, including Λ. Every factor from $(S^*)^+$ is made up of factors from S^+, excluding Λ if it doesn't exist. However, since S* will always contain Λ, $(S^*)^+$ will also always contain Λ.

Every factor from S^+ is made up of factors from S, excluding Λ if it's in the set. Every factor from $(S^+)^*$ is made up of factors from S^+, including Λ even if it doesn't exist in S^+.

Therefore, every word in $(S^*)^+$ is made up of factors of S including Λ, and every word in $(S^+)^*$ is made up of factors from S including Λ. Therefore, every word in $(S^+)^*$ is also a word in $(S^*)^+$

12. Let *S = {a, bb, bab, abaab}.* Is *abbabaabab* in ***S*?*** Is *abaabbabbaabb?* Does any word in S* have an odd total number of b's?

Step 1 of 4

Language S = {a bb bab abaab} then, the language

S*= { ∧, a, aa, bb, abb, bab, bba, bbbb, abab, baba, abaab, aabaab, bbbab, babbb, aabaab, babbab, abaaba, bbabaab, abaabbb, bababaab, abaabbb, abaabbab, abaababaab}

Step 2 of 4

The string "abbabaabab" does not contain in this language.

Step 3 of 4

The string "abaabbabbaabb" does not contain in this language.

Step 4 of 4

NO, the S* don't have odd of total number of b's because each word that contains b's are even and hence by concatenating the even number of b's we get even number of b's only.

13. Suppose that for some language L we can always concatenate two words in L and get another word in L if and only if the words are not the same. That is, for any words w, and w2 in L where *w, 4-* w2, the word wIw 2 is in L but the word *w1w1* is not in L. Prove that this cannot happen.

Step 1 of 1

Given theorem is contradiction, because

For example consider, Language S = {w1 w2}

Then S* = { ^, w1, w2, w1w1, w1w2, w2w2, w1w1w1,.....}

Thus, the language contains both the strings w1w2 (for w1 ≠ w2) and w1w1. That is the language set is closed under concatenation.

And hence the given statement is false.

14. By definition
(S)* = S*****
is this set bigger than S*? Is it bigger than **S?**

Step 1 of 1

$$\left(S^{**}\right)^{*} = S^{***}$$

$$\left(S^{*}\right)^{*} = S^{***} \qquad \left(\because\ S^{***} = S^{*}\right)$$

$$S^{*} = S^{***}$$

∴ This set is equal to S*, and is bigger than S.

15th ans

Step 1 of 2

(I) We know that S and T are languages such that

T=S + w and T*=S*

e.g.

Let S = {a aa bb}

Let w = aaa

Then, T = {aa bb aaa}

Still, S* = T*, as S belongs to T*, so S* also belongs to T* and are equal.

But it does not mean, that w belongs to S, as stated in this part.

Step 2 of 2

(II) **True**, because S* = T*, as w belongs to T* hence it also should belong to S*

LANGUAGES 25

16. Give an example of a set S such that the language **S*** has more six letter words than seven letter words. Give an example of an **S*** that has more six letter words than eight letter words. Does there exist an ***S**** such that it has more six letter words than twelve letter words?

Step 1 of 1

For a set e.g. S = {aabbaa} we get a closure of:

S*={aabbaa aabbaaaabbaa aabbaaaabbaaaabbaa ...}

We see we hava one 6-letter long word and none 7 or 8 long letters.

Other example for 7 letters might be S = {aa bb}, as we can create many 6-letter words and none 7-letter.

Other example of 8 ltters might be S = {abb bbb} - many six letter words, none 8-letter words

It is not possible to have more six-letter words than 12-letter words, as 6 is a divisor of 12. In other words, we can concatenate 3-letter words to get 12-letter one, we can concatenate two 6-letter words, 4x3-letter words etc.

17. (i) Consider the language **S*** where S {aa, *ab*, *ba*, *bb}*. Give another description of this language.

(ii) Give an example of a set S such that **S*** contains all possible strings of a's and b's that have length divisible by three.

Step 1 of 3

I) S={aa ab ba bb}
Language representation for S* is
S*={Λ aa ab ba bb aaaa aaab aaba bbaa abbb..}
S*={λ Plus all the words composed of aa ab ba bb where length of all words is even}

Step 2 of 3

II) S*={aba abb baa bba aab aba aabbbb aababa.....}

The S^* only contains all possible strings of a's and b's this have length divisible by 3.

S={aaa aba baa bbb abb bba aab bab}

Step 3 of 3

III) S={a b}

S = {a b aaa aab aba abb baa bab bba bbb aaaaa aaaab ...}

S* = {a b aa ab ba bb aaa ...} = all strings of a's and b's

Note: $S \subset S^*$, S contains only odd length strings. S* contains all possible even length and odd length strings.

18th ans

Step 1 of 1

i) For a given language S = {a b} we might create its closure:

S*={^ a b aa ab ba bb aaa aab aba ...}

We clearly see that S* contains all the possible strings cocatenated of a's and b's. If T*=S*, then T must also contain a and b (with perhaps sth additional). This is the only way, when we create all possible strings using a and b - to have them in language.

e.g. T={a b aa} or T={a b aabb} --> T contains S

ii) Another example of these kind of sets:

S={a} T = {a aa aaaa}

S={aa bb} T = {aa bb aabbaa}

19. One student suggested the following algorithm to test a string of a's and b's to see if it is a word in **S*** where S = {aa, *ba*, *aba*, *abaab}*. Step 1, cross off the longest set of characters from the front of the string that is a word in S. Step 2, repeat step 1 until it is no longer possible. If what remains is the string A, the original string was a word in ***S*.** If what remains is not A (this means some letters are left but we cannot find a word in *S* at the beginning), the original string was not a word

in **S*.** Find a string that disproves this algorithm.

Step 1 of 1

For set S = { aa ba aba abaab} we have to find a word, which does belong to S* but does not follow the crossing algorithm (cross the longest factor in S from left repeatedly, until left with null string or non-empty string, which means the word either is in S* or not).

As an example let's take word abaaba, which is a concatenation of aba and aba. If we cross out the longest factor of S from left side, meaning abaab, we are left with a, which is not a factor of S and cannot be cross out longer. It would mean, that abaaba does not belong to S*, which is fault statement.

20. The reason * is called the "closure operator" is because the set ***S**** *is closed under concatenation.* This means that if we take any two words in S* and concatenate them, we get another word in ***S*.*** If *S = {ab, bbb}*, then S is not closed under concatenation since *abab* is not in ***S,*** but ***S**** is closed under concatenation.
(i) Let T be any set that contains all of ***S,*** and suppose T is closed under concatenation. Show that T contains **S*.**
(ii) Explain why we may say **"S*** is the smallest set that is closed under concatenation that contains S."
(iii) What is the smallest set, closed under concatenation, that contains both the sets of words *P* and **Q?**

Step 1 of 1

We have to show that for a given language T, of property:

if t1 belongs to T, and t2 belongs to T, then t1t2 belongs to T,

we get:

if $S \subset T$, and $T \neq S^*$, Then S^* is smaller than T.

Firstly, we see that for any given words belonging to T, the concatenation of those two also belongs to T. Next, the concatenation of the created word with another in the set also belongs to T.

As we see, **S belongs to T,** therefore, all the word in S are part of set T. Using the given property, we can concatenate any of the words belonging to S with each other and the word stil belongs to T. If we state, that the concatenation goes forever, then we would eventually get S^*.

We now see, that $S^* \subset T$. We also know that $T \neq S^*$. It means then, that S^* is a proper subset of T and therefore is smaller than T.

Chapter 3

PROBLEMS

1. Write another recursive definition for the language L, of Chapter 2.

Step 1 of 1

Recursive definition:

$$L_1 = \{x, xx, xxx, xxxx, xxxxx, \ldots\ldots\}$$

The language Rule 1 : x is in L_1

Rule 2 : If x is in L_1, then so x^{n+1} for $n = 0,1,2,34\ldots\ldots$

So, the alternative form of L_1 is,

$$L_1 = \{ x^{n+1} \text{ for } n = 0,1,2,3,\ldots\ldots\}$$

2. Using the second recursive definition of the set EVEN, how many different ways can we prove that 14 is in EVEN?

If x and y both are even then so x+y

There are three different ways to prove that 14 is EVEN, based on the above definition,

I) Rule 1 : 2 is in EVEN
Rule 2 : x=2, y=2, 2+2 = 4 is in EVEN
Rule 2 : x=2, y=4, 2+4 = 6 is in EVEN
Rule 2 : x=4, y=4, 4+4 = 8 is in EVEN
Rule 2 : x=6, y=8, 6+8 = 14 is in EVEN

II) Rule 1: 2 is in EVEN
Rule 2: x= 2, y= 2 , 2+2=4 is in EVEN
Rule 2: x= 4, y= 4 , 4+4=8 is in EVEN
Rule 2: x= 8, y= 4 , 8+4=12 is in EVEN
Rule 2: x=12,y= 2 , 12+2=14 is in EVEN

III) Rule 1 : 2 is in EVEN
Rule 2 : x=2, y=2, 2+2 = 4 is in EVEN
Rule 2 : x=4, y=4, 4+4 = 8 is in EVEN
Rule 2 : x=2, y=8, 2+8 = 10 is in EVEN
Rule 2 : x=4, y=10, 4+10 = 14 is in EVEN

3. Using the second recursive definition of EVEN, what is the smallest number of steps required to prove that 100 is EVEN? Describe a good method for showing that 2n is in EVEN.

Step 1 of 3

The second recursive definition of EVEN is:

Rule 1 2 is in EVEN.

Rule 2 If x and y are both in EVEN then so is

x+y (even if x and y have the same number)

Hence, for the number 100,

By Rule 1 2 is in the set EVEN

By Rule 2 x=2, y=2 $\rightarrow$ 4 is in the set EVEN

By Rule 2 x=4, y=2 $\rightarrow$ 6 is in the set EVEN

By Rule 2 x=6, y=4 $\rightarrow$ 10 is in the set EVEN

By Rule 2 x=10, y=10 $\rightarrow$ 20 is in the set EVEN

By Rule 2 x=20, y=20 $\rightarrow$ 40 is in the set EVEN

By Rule 2 x=40, y=40 $\rightarrow$ 80 is in the set EVEN

By Rule 2 x=80, y=20 $\rightarrow$ 100 is in the set EVEN

Step 2 of 3

$\therefore$ 100 is in set EVEN The total number of steps required to prove that 100 is in the set EVEN is 8

Provide feedback (0)

Step 3 of 3

To show 2n is even,

For n=1, 2n = 2 which is EVEN

For n=2, 2n = 4, By Rule 2: x=2, y=2, 2+2 = 4 is in EVEN

For n=3, 2n = 6, By Rule 2: x=2, y=4, 2+4 = 6 is in EVEN

.

.

.

.

If 'n' is even

x=2, y=n $\rightarrow$ 2n is EVEN

If 'n' is odd

x=2, y=n+1 $\rightarrow$ 2n is EVEN

4. Show that the following is another recursive definition of the set EVEN.
Rule 1 2 and 4 are in EVEN.
Rule 2 If x is in EVEN, then so is x + 4.

Step 1 of 2

Recursive definition of set EVEN

Recursive definition of the set EVEN is,

Rule 1: 2 and 4 are in EVEN

Rule 2: If *x* is in EVEN, then so $x+4$

Define EVEN set for all positive integers divisible by 2.

EVEN defined by three rules

Rule 1: 2 and 4 are in EVEN,

Rule 2: If *x* is in EVEN, then so $x+4$

Rule 3: The elements in the set EVEN can produced from above two rules

Suppose proving of 18 is in the set EVEN

By Rule 1: **2** and **4** are in EVEN

By Rule 2: 2 is in EVEN, then **2+4** = **6** is in EVEN

By Rule 2: 6 is in EVEN, then **6+4** = **10** is in EVEN

By Rule 2: 10 is in EVEN, then **10+4** = **14** is in EVEN

By Rule 2: 14 is in EVEN, then **14+4** = **18** is in EVEN

Step 2 of 2

We can make another definition of the set EVEN

Rule 1: 2 and 4 are in EVEN,

Rule 2: *x* and *y* are in EVEN, then so $x+y$ or $x+y+4$ are in EVEN

Rule 3: The elements in the set EVEN can produced from above two rules

Suppose proving the same 18 is in EVEN by causing new implemented rules

By Rule 1: **2** and **4** are in EVEN

By Rule 2: 2 and 4 are in EVEN, then **2+4** = **6** or **2+4+4** = **10** is in EVEN

By Rule 2: 6 and 4 are in EVEN, then **6+4** = **10** or 6**+4+4** = **14** is in EVEN

By Rule 2: 10 and 4 are in EVEN, then **10+4** = **14 or 10+4+4** =**18** is in EVEN

By Rule 2: 14 and 4 are in EVEN, then **14+4** = **18 or 14+4+4=22** is in EVEN

Hence it is another recursive definition of the set EVEN

5. Show that there are infinitely many different recursive definitions for the set EVEN.

Step 1 of 5

Recursive definitions for any set can be defined by the following three steps,

1) Specify the basic words/objects in the set
2) Rules for constructing more words from ones already known (recursive case)
3) Declare that no word except those constructed by following rules 1 and 2 are in the language.

Provide feedback (0)

Step 2 of 5

Recursive definitions for EVEN set are

Def 1: EVEN = { All positive whole numbers divisible by 2}

Eg : Consider number 12 ,

This is divisible by 2. Thus, 12 is EVEN

Step 3 of 5

Def 2 : EVEN = { 2n / n = 1 2 3 4 .....}

Provide feedback (0)

Step 4 of 5

Def 3 : The set EVEN is defined by these rules,

Rule 1 : 2 is in EVEN

Rule 2 : If x is in EVEN , then so x+2

Rule 3 : The only elements in the set EVEN are those that can be produced from the two rules above.

Eg : 12 is in EVEN

Rule 1: 2 is EVEN

Rule 2: x = 2, 2 + 2 =4 EVEN

Rule 2: x = 4, 4 + 2 =6 EVEN

Rule 2: x = 6, 6 + 2 = 8 EVEN

Rule 2: x = 8, 8 + 2 = 10 EVEN

Rule 2: x = 10, 10 + 2 = 12 EVEN

Step 5 of 5

Def 4 : The set EVEN is also defined as
Rule 1 : 2 is EVEN
Rule 2 : If x & y are both EVEN, then so x+y
Eg : 12 is in EVEN
Rule 1: 2 is EVEN
Rule 2: $x = 2, y = 2, 2 + 2 = 4$ EVEN
Rule 2: $x = 4, y = 4, 4 + 4 = 8$ EVEN
Rule 2: $x = 4, y = 8, 4 + 8 = 12$ EVEN

Thus, there are many different recursive definitions for the set EVEN

6. Using any recursive definition of the set EVEN, show that all the numbers in it end in the digits 0, 2, 4, 6, or 8.

Step 1 of 3

1) Basic step:

$0 \in$ EVEN by definition, therefore the property is true of the zero' the step since $0 \in \{0,2,4,6,8\}$

Provide feedback (0)

Step 2 of 3

2) Induction hypothesis:

Assume that the last digit of $(m+2) \in \{0,2,4,6,8\}$ for $0 < m < n$

Step 3 of 3

3) Induction step:

n	∈ EVEN
0	2
1	4
2	6
3	8
4	10

n	∈ EVEN
n	2n+2
n+1	(2n+2)+2
n+1	2(n+1)+2

∴ 0+2=2, 2+2=4, 4+2=6, 6+2=8, 8+2=0 ∈ {0, 2, 4, 6, 8}

Hence all the numbers end in digits 0, 2, 4, 6 ..

7. The set POLYNOMIAL defined in this chapter contains only the polynomials in the one variable x. Write a recursive definition for the set of all polynomials in the two variables x and y.

Step 1 of 1

The recursive definition for set of polynomials for two variables x and y is defined as,

Rule 1 : Any number is in POLINOMIAL
Rule 2 : Variable 'x' is in POLINOMIAL
Rule 3 : Variable 'y' is in POLINOMIAL
Rule 4 : If p and q are in POLINOMIALS, then so are p+q, p-q, p and pq

Now consider an example to illustrate this definition,

$5x^2+9y^2+7y+3x-9$

Rule 1 : 5 is in POLINOMIAL
Rule 2 : x is in POLINOMIAL
Rule 4 : (5x)(x) is in POLINOMIAL ; call it $5x^2$

Rule 1 : 9 is in POLINOMIAL
Rule 3 : y is in POLINOMIAL
Rule 4 : (9y)(y) is in POLINOMIAL ; call it $9y^2$

Rule 1 : 3 is in POLINOMIAL
Rule 4 : (3)(x) is in POLINOMIAL

Rule 1 : 7 is in POLINOMIAL
Rule 4 : (7)(y) is in POLINOMIAL

Rule 1 : -9 is in POLINOMIAL

Rule 4 : $5x^2+9y^2+7y+3x-9$ is in POLINOMIAL

8. Define the set of valid algebraic expressions ALEX as follows:
Rule 1 All polynomials are in ALEX.
Rule 2 If *f(x)* and *g(x)* are in ALEX then so are
(i) (f(x))
(ii) - ***(f(x))***
(iii) *f(x)* + *g(x)*
(iv) *f(x)* - *g(x)*
(v) *f(x)g(x)*
(vi) *f(x) / g(x)*
36 AUTOMATA THEORY
(vii) ***f(x)9(x)***
(viii) *ftg(x))*
(a) Show that (x + 2)3x is in ALEX.

(b) Show that elementary calculus contains enough rules to prove the theorem that all algebraic expressions can be differentiated.
(c) Is Rule (viii) really necessary?

Step 1 of 2

(a) By rule 1 all polynomials are in ALEX, so the $(x+2)^{3x}$ is in ALEX

Proof:
By Rule2: x is in ALEX
By Rule1:2 is in ALEX
By Rule3: $x+2$ is in ALEX
By Rule3: $(x+2)$ is in ALEX
By Rule3: $(x+2).(x+2)$ is in ALEX
By Rule3: $(x+2).(x+2).(x+2)$ is in ALEX
By Rule3: $(x+2)^3$ is in ALEX
By Rule3: $(x+2)^{3.x}$ is in ALEX.

Provide feedback (2)

Step 2 of 2

(c) It's necessary for $f(x)$ contain the $g(x)$

9. Using the fact that 3x2 + 7x - 9 = (((((3)x) + 7)x)-9), show how to produce this polynomial from the rules for POLYNOMIAL using multiplication only twice. What is the smallest number of steps needed for producing x' + x4? What is the smallest number of steps needed for producing 7x7 + 5*x*5 + 3X3 + *X?*

Step 1 of 3

$3x^2+7x-9=\left(\left(\left(\left((3)x\right)+7\right)x\right)-9\right)$

Rules for POLYNOMIAL

Rule 1: Any number is in POLYNOMIAL

Rule 2: The variable x is in POLYNOMIAL

Rule 3: If p and q are in POLYNOMIAL, then so are p + q, p − q, (p), and p q

By rule 1: 3 is in POLYNOMIAL
By rule 3: (3) is in POLYNOMIAL
By rule 2: x is in POLYNOMIAL
By rule 3: ((3)x) is in POLYNOMIAL; call it 3x
By rule 1: 7 is in POLYNOMIAL
By rule 3: ((3)x)+7 is in POLYNOMIAL; call it 3x+7
By rule 3: (((3)x)+7) is in POLYNOMIAL; call it (3x+7)
By rule 3: ((((3)x)+7)x) is in POLYNOMIAL; call it (3x+7)x i.e., $3x^2+7x$
By rule 1: -9 is in POLYNOMIAL
By rule 3: $3x^2+7x-9$ is in POLYNOMIAL

Step 2 of 3

x^8+x^4

By rule 2: x is in POLYNOMIAL

By rule 3: $(x)(x)$ is in POLYNOMIAL; x^2

By rule 3: $\left(x^2\right)\left(x^2\right)$ is in POLYNOMIAL; x^4

By rule 3: x^4+1 is in POLYNOMIAL

By rule 3: $\left(x^4\right)\left(x^4+1\right)$ is in POLYNOMIAL; x^8+x^4

By rule 3: x^8+x^4 is in POLYNOMIAL

Step 3 of 3

$7x^7+5x^5+3x^3+x$

By rule 1: 7 is in POLYNOMIAL
By rule 2: x is in POLYNOMIAL
By rule 3: $(7)(x)$ is in POLYNOMIAL
By rule 3: $(7x)(x)$ is in POLYNOMIAL
By rule 3: $(7x^2)(x^2)$ is in POLYNOMIAL
By rule 1: (x^3) is in POLYNOMIAL
By rule 3: $(7x^4)(x^3)$ is in POLYNOMIAL
By rule 3: $(7x^7)$ is in POLYNOMIAL
By rule 1: +5 is in POLYNOMIAL
By rule 3: $+(5)(x)$ is in POLYNOMIAL
By rule 3: $+(5)(x^4)(x)$ is in POLYNOMIAL
By rule 3: $+(5x^5)$ is in POLYNOMIAL
By rule 1: +3 is in POLYNOMIAL
By rule 3: x is in POLYNOMIAL
By rule 1: (x^2) is in POLYNOMIAL
By rule 3: $+(3)(x)(x^2)$ is in POLYNOMIAL
By rule 3: $+3x^3$ is in POLYNOMIAL
By rule 3: $7x^7+5x^5+3x^3$ is in POLYNOMIAL
By rule 3: $7x^7+5x^5+3x^3+x$ is in POLYNOMIAL

10. Show that if *n* is less than 29, then x' can be shown to be in POLYNOMIAL in fewer than eight steps.

Step 1 of 1

By rule 2: x is in POLYNOMIAL
By rule 3: $(x)(x)$ is in POLYNOMIAL
By rule 3: $(xx)(xx)$ is in POLYNOMIAL
By rule 3: $(xxxx)(xxxx)$ is in POLYNOMIAL
By rule 3: $(xxxxxxxx)(xxxxxxxx)$ is in POLYNOMIAL
By rule 3 : $(xxxxxxxxxxxxxxxx)(xxxxxxxxxx)$ is in POLYNOMIAL
By rule 3: $(xxxxxxxxxxxxxxxxxxxxxxxxxxxx)(xxxx)$ is in POLYNOMIAL

Only it takes less then 8 steps.

11. In this chapter we mentioned several substrings of length 2 that cannot

occur in arithmetic expressions, such as **(/,** +), // and ***/.** What is the complete list of substrings of length 2 that cannot occur?

Step 1 of 1

List of substrings:

//
aa
bb

12. Are there any substrings of length 3 that cannot occur that do not contain forbidden substrings of length 2? (This means that *1/H* is already known to be illegal because it contains the forbidden substring **//.)** What is the longest forbidden substring that does not contain a shorter forbidden substring?

Step 1 of 1

aaabbbaaa.....

13. The rules given above for the set AE, allow for the peculiar expressions **(((((9)))))** and -(-(-(-())
It is not really harmful to allow these in AE, but is there some modified definition of AE that eliminates this problem?

Step 1 of 1

Yes, Modified definition of AE eliminates this problem by Rule 2

In Rule 2: if x is in AE, then so are
(i) (x)
(ii) –x (**Provided x does not already start with a minus sign**)

By form of Rule 2
(i) (((((9))))) it can be acceptable

Forbidden strings are not acceptable
(ii) Instead of - (-(-(-(9)))) it changed in to –((((9))))

14. Write out the full recursive definition for the propositional calculus that contains the symbols V and A as well as ---i and --. What are all the forbidden substrings of length 2 in this language?

Step 1 of 2

I) The new language is calculus

Rule 1: x is in calculus
Rule 2: $\wedge, \vee, \neg,$ and $\rightarrow$ are in calculus

Provide feedback (0)

Step 2 of 2

II) No far bidden substring in the language

15. (i) When asked to give a recursive definition for the language PALINDROME over the alphabet **I** = *{a, b}*, a student wrote:
Rule 1 a and b are in PALINDROME
Rule 2 If x is in PALINDROME, then so are axa and *bxb*
Unfortunately all of the words in the language defined above have an odd length and so it is not all of PALINDROME. Fix this problem.
(ii) Give a recursive definition for the language EVENPALINDROME of all palindromes of even length.

Step 1 of 2

i) Rule 1 : **Λ**, a, b are in PALINDROME
Rule 2 : If x is in PALINDROME, so axa, bxb

The set of PALINDROME = { ^, a, b, aa, bb, aaa, aba, bab, bbb, aaaa, abba,.....}

We have added the Null string to rule 1 it allows us to have strings of the form aa since Null string can be substituted for x in axa (per Rule 2 x is in PALINDROME and Null string meets that requirement). Once we prove that aa is in PALINDROME we can build all the other EVEN PALINDROMES as well.

Consider example for even length Palindrome

→ axa
→aaxaa [from rule 2]
→aaaa [from rule 1]

Hence it supports both even and odd length PALINDROME.

Step 2 of 2

ii) The set of EVEN PALINDROME is as shown,
{ ^, aa, bb, aaaa, bbbb, abba, baab,.....}

Rule 1 : aa and bb are in EVENPALINDROME.
Rule 2 : If x is in EVENPALINDROME, then so are axa, bxb.

16. (i) Give a recursive definition for the set ODD = {1,3,5,7 . . .
(ii) Give a recursive definition for the set of strings of digits 0, 1, 2, 3, . . . 9 that cannot start with the digit 0.

Step 1 of 2

I)

rule 1: 1 is in ODD
rule 2: if x is in ODD then so is x + 2

proof:
1 is in ODD by rule 1
3 is in ODD by rule 2 (1 + 2 = 3)
5 is in ODD by rule 2 (3 + 2 = 5)
7 is in ODD by rule 2 (5 + 2 = 7)
.....

Provide feedback (0)

Step 2 of 2

II) We call this set of strings POSITIVE 01

rule 1: 0 and 1 are in positive
rule 2 if x is not zero and in positive then so is x + 1

proof:
0 and 1 are in positive by rule 1
1 is in positive by rule 1, and is not 0
2 is in positive by rule 2 (1+1 = 2)
3 is in positive by rule 2 (2+1 =3)

17. (i) Give a recursive definition for the language S* where S = *{aa,b}*.
(ii) Give a recursive definition for the language T* where
T = {wj, w2, w3, *W41*,
where these w's are some particular words.

Step 1 of 1

Positive numbers is in L

i.e. 1 is in L

The language L defined in "set of positive integers" of mathematical set form
Prove: If x = 1 and y = 1 are in L,

Then $x + y = 1 + 1$
$= 2$ is positive
$x \times y = 2 \times 1$
$= 2$ is positive
$\frac{x}{y} = \frac{2}{2} = 1$ is positive

Note: It is a positive numbers of infinite decimal representations.

18. Give two recursive definitions for the set
POWERS-OF-TWO = **{1** 2 4 8 16 . . . }
Use one of them to prove that the product of two powers of two is also a power of two.

Step 1 of 1

Definition 1:

1 is a POWERS – OF – TWO
If x is a POWERS – OF – TWO, then $2x$ is also a POWERS – OF – TWO

Definition 2:

1 and 2 are POWERS – OF – TWOS
If x and y are POWERS – OF – TOWS, then xy is a POWERS – OF –TWO

19. Give recursive definitions for the following languages over the alphabet *{a,b}:*
(i) The language EVENSTRING of all words of even length.
(ii) The language ODDSTRING of all words of odd length.
(iii) The language AA of all words containing the substring aa.
(iv) The language NOTAA of all words not containing the substring aa.

Step 1 of 4

Recursive definition defined over { a b } are as follows,

i) Recursive definition for EVENSTRING

Rule 1 : aa, ab, ba, bb are the strings in EVENSTRING
Rule 2 : If 'w' and 'x' are strings in EVENSTRING then xw or wx is EVENSTRING.

Step 2 of 4

ii) Recursive definition for ODDSTRING

Rule 1 : a, b are strings in ODDSTRING
Rule 2 : If 'x' is a string in ODDSTRING then axa, bxb, axb, bxa are all strings in ODDSTRING.

Provide feedback (0)

Step 3 of 4

iii) Recursive definition for AA

Rule 1 : aa is the string in AA
Rule 2 : If 'x' is in AA, then so xa, xb, and bx are all strings in AA.

Provide feedback (3)

Step 4 of 4

iv) Recursive definition for NOTAA

Rule 1 : $\in$, a, b are strings in NOTAA
Rule 2 : If 'x' and 'w' are in NOTAA. then so xbwb is in NOTAA

20. (i) Consider the following recursive definition of 3-PERMUTATION
(a) 123 is a 3-PERMUTATION
(b) if *xyz* is a 3-PERMUTATION then so are
zyx and yzx
Show that there are six different 3-PERMUTATION's.
(ii) Consider the following recursive definition of 4-PERMUTATION
(a) 1234 is a 4-PERMUTATION
(b) if *xyzw* is a 4-PERMUTATION then so are
wzyx and *yzwx*

How many 4-PERMUTATION's are there (by this

Step 1 of 2

(I) Rule 1 123 is a 3-PERMUTATION
Rule 2 If the xyz is a 3-PERMUTATION, then xyz, xzy,yxz,yzx,zxy,zyx is also a PERMUTATION

i.e.

(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)
(x, y, z), (x, z, y), (y, x, z), (y, z, x), (z, x, y), (z, y, x)

Thus, there are 6 different 3-PERMUTATIONS

Provide feedback (0)

Step 2 of 2

(II) For N objects the number of PERMUTATIONS are N!
For 4 objects the total number of PERMUTATIONS is 4! = 24

Chapter 4

PROBLEMS

1. Let ri, r 2, and r3 be three regular expressions. Show that the language associated with (r, + r 2)r3 is the same, as the language associated with rjr 3 + r2r 3. Show that r1 (r2 + r 3) is equivalent to r1r2 + r j r3 . This will be the same as "proving a distributive law" for regular expressions. Construct a regular expression defining each of the following languages over the alphabet **I** = *{a, b}.*

Step 1 of 2

The r_1, r_2, r_3 are three regular expressions and L_1, L_2, L_3 are three languages

If regular expressions r_1 generates the language L_1, regular expressions r_2 generates the language L_2, and regular expressions r_3 generates the language L_3, then the $(r_1+r_2)r_3$ generates to the $(L_1+L_2)L_3$ languages and $(r_1 r_3+r_2 r_3)$ generates to the $(L_1+L_2)L_3$ languages

$\therefore$ $(r_1+r_2)r_3$ and $(r_1 r_3+r_2 r_3)$ generates the same language then

$$(r_1+r_2)r_3 = (r_1 r_3+r_2 r_3)$$

Step 2 of 2

Similarly $r_1(r_2+r_3)$ generates the $L_1(L_2+L_3)$ languages and $(r_1 r_2+r_1 r_3)$ generates the $L_1(L_2+L_3)$ so

$$\therefore r_1(r_2+r_3) = (r_1 r_2+r_1 r_3)$$

2. All words in which a appears tripled, if at all. This means that every clump of a's contains 3 or 6 or 9 or 12... a's.

Step 1 of 1

Construct regular expression

Input alphabet $\sum = \{a\ b\}$

Generate words as 'a' tripled

Every clump of a's contains 3 or 6 or 9 or 12...a's $(aaa)^*$

In a language any occurrences of *a's* and *b's* are possible, whereas *a's* are always tripled or multiple of three.

Construct regular expression *R*,

$$R = (b+aaa)^*$$

Examples of strings generated $baaa, aaaaaab, bbaaaaaabbaaa, etc$

Hence, the regular expression is $\boxed{(b+aaa)^*}$.

3. All words that contain at least one of the strings *s, s2 s3* or *s4* .

Step 1 of 1

Assume Anystring(a+b)*

At least one of the substrings as s1, s2, s3 or s4:

(a+b)*(s1+s2+s3+s4) (s1+s2+s3+s4)*(a+b)*

4. All words that contain exactly three b's in total.

Step 1 of 1

a*ba*ba* + a*ba*ba*ba*

5. All words that contain exactly two b's or exactly three b's, not more.

6. (i) All strings that end in a double letter.
(ii) All strings that have exactly one double letter in them.

Step 1 of 2

I) $(a+b)^* (aa+bb)$ -the string end with a double letter

Provide feedback (4)

Step 2 of 2

II) $(a+b)^* (ab+ba)+a+b+\in$ -the string do not end in a double letter

Step 1 of 1

$(b+\in)(ab)^* aa(ba)^* (b+\in)+(a+\in)(ba)^* bb(ab)^* (a+\in)$ -the string have exactly one double letter

7. All strings in which the letter b is *never* tripled. This means that no word contains the substring *bbb*.

Step 1 of 1

$(a+ba+bba)^*(bb+b+\varepsilon)$ - here no word contains the substring bbb

8. All words in which a is tripled or b is tripled, but not both. This means each word contains the substring aaa or the substring bbb but not both.

Step 1 of 1

(^+ b + bb)(a + ab + abb)*aaa(^ + b + bb)(a + ab + abb)* +(^ + a + aa)(b + ba + baa)*bbb(^ + a + aa)(b + ba + baa)*

9. (i) All strings that do not have the substring ab.
(ii) All strings that do not have both the substrings bba and *abb*.

Step 1 of 2

(i) The regular expression for the words that do not have substring 'ab' is, b^*a^*

Provide feedback (15)

Step 2 of 2

(ii) a*(baa*)*b* + b*(a*ab)*a*

10. All strings in which the *total* number of a's is divisible by three, such as *aabaabbaba*.

Step 1 of 1

Regular Expression :

$(b^*ab^*ab^*ab^*)^*b^*$

The first expression has * around it, it can occur 0 or more times to give us any number of a's that is divisible by 3

11. (i) All strings in which any b's that occur are found in clumps of an odd number at a time, such as *abaabbbab*.
(ii) All strings that have an even number of a's and an odd number of *b's*.
(iii) All strings that have an odd number of a's and an odd number of b's.

Step 1 of 3

i) The regular expression for the strings for any b's that are found in clumps of an odd number at a time is,

a*(b(bb)*aa*)*(Λ+b(bb)*)

Compulsory a after some number of odd b's are there. ODD+ODD = EVEN, so here needed to separate the odd clumps.

Step 2 of 3

ii) The regular expression for strings of even number of a's and odd number of b's

Divide this language to two groups:

a) When words that start with b and followed by even number of a's and even number of b's. It becomes odd number of b's and even number of a's

b) When words that start with a and followed by odd number of a's and odd number of b's. It also becomes odd number of b's and even number of a's

b[aa+bb+(ab+ba)(aa+bb)*(ab+ba)]*+
a[[aa+bb+(ab+ba)(aa+bb)*(ab+ba)]*(ab+ba)[aa+bb+(ab+ba)(aa+bb)*(ab+ba)]*]

Step 3 of 3

iii) The regular expression for the strings of odd number of a's and odd number of b's .

The small string is ab or ba we can add even letter string left or right or both.

[aa+bb+(ab+ba)(aa+bb)*(ab+ba)]*(ab+ba)[aa+bb+(ab+ba)(aa+bb)*(ab+ba)]*

REGULAR **EXPRESSIONS 61**

12. Let us reconsider the regular expression

(a + b)*a(a + b)*b(a + **b)***

(i) Show that this is equivalent to

(a + b)*ab(a + b)*

in the sense that they define the same language.

(ii) Show that

(a + b)*ab(a +**b)*** + b*a* = (a + **b)***

(iii) Show that

(a + **b)*** ab[(a + b)*ab(a + **b)*** + b*a*] + b*a* = (a + **b)***

(iv) Is (iii) the last variation of this theme or are there more beasts left in this cave?

Step 1 of 1

1) **(a+b)*a(a+b)*b(a+b)*** defines a language with atleast one **a** & one **b** such that there is atleast one **a** followed by **b**. The same language is defined by regular expression **(a+b)*ab(a+b)*.**

2) **(a+b)*ab(a+b)*+b*a*** has a choice.First option accepts all words with a **ab** in it while second option accepts all words that don't contains **ab** in it (it accepts strings of only **a** or **b** or all **b** before **a**). Together this accepts all words whose regular expression is **(a+b)***

3) **(a+b)*ab[(a+b)*ab(a+b)*+b*a*]+ b*a*** is a regular expression which can be deduced step by step from above parts

(a+b)*ab[(a+b)*ab(a+b)*+b*a*]+ b*a* => (a+b)*ab[(a+b)*]+ b*a*

=> (a+b)*ab(a+b)*+ b*a*

=> (a+b)*

4)Part 3 actually has no variation but it is using the same formula again & again within the expression to make it more complex. Actually it is adding nothing to it.

13. We have defined the product of two sets of strings in general. If we apply this to the case where both factors are the same set, S = T, we obtain squares, ***S2.*** Similarly we can define ***S3, S4.*** Show that
(i) S* = **A** + ***S*** + ***S′*** + ***S2*** + S3 + S 4 + ...
(ii) ***S***+ = *S* + ***S1*** + ***S2*** + ***S3*** + *S4* + **...**
Show that the following pairs of regular expressions define the same language over the alphabet **I** = {a, *b}.*

Step 1 of 1

For two identical sets S=T, product of them is equal to S^2, for three of them - S^3 and so on. Now we have to prove that:

a) $S^* = \lambda + S + S^1 + S^2 + S^3 + \ldots$

OK, to understand this relation we need to think about the nature of Kleene closure. It is obtained by concatenaiting with each other a finite number of words belonging to a certain language + λ. It means that this concatenation can always be presented as a combination of basic words "multiplied" by each other.

In this theorem, the RHS is containing all possible sets generated by multiplying S by S finitely many times. In other words, by doing so we get all possible words of all lengths comprised of basic words belonging to S e.g.:

S = {aa bb}

Then S* = {λ aa bb aaaa aabb bbaa bbbb aaaaaa aaaabb ...}

RHS = λ + {aa bb} + {aaaa aabb bbaa bbbb} + {aaaaaa aaaabb ... bbbbbb} +... = S*

b) In this case we need to prove that:

$S^+ = S + S^1 + S^2 + S^3 + \ldots$

This is a concatenation of two sets R and S, denoted RS, is {st|s ε S ad t ε R}. We will sometimes write $S^2$for SS, the concatenation of S with itself, and S^3 for SSS {SS^2}.This set does not contain λ.

By analogy we see that the only difference is that both sets does not contain λ, so the theorem from the first part still holds.

14. (i) (ab)*a and a(ba)*
(ii) (a* + **b)*** and (a + **b)***
(iii) (a* + **b*)*** and (a + **b)***

Step 1 of 1

The difference between L^2 and L^* is the word $\wedge$, and words of length more than 2. Because L^* contains all words of L^2. but L^2 does not contain all words of L*.

15. (i) **A*** and A
(ii) (a*b)*a* and a*(ba*)*
(iii) (a*bbb)*a* and a*(bbba*)*

Step 1 of 3

Two Regular expressions define the same language

i) $(ab)^* a$ and $a(ba)^*$

The language defined by the expression $(ab)^* a$

Set of all strings of a's and b's, that have at least 'a' as string and that have nothing but b's inside.

Language1 $(ab)^* a$.

$(ab)^* = \{\wedge, ab, abab, ababab\}$

$(ab)^* a = \{a, aba, ababa, abababa\}$

The language defined by the expression $a(ba)^*$

Set of all strings of a's and b's, that have at least 'a' as string and that have nothing but b's inside.

Language2 $a(ba)^*$.

$(ba)^* = \{\wedge, ba, baba, bababa\}$

$a(ba)^* = \{a, aba, ababa, abababa\}$

Hence Language1 and Language2 define the same set of language.

Step 2 of 3

ii) $\left(a^* + b\right)^*$ and $\left(a + b\right)^*$

The language defined by the expression $\left(a^* + b\right)^*$

Set of all strings of a's and b's

Language1 $\left(a^* + b\right)^*$.

$\left(a^* + b\right) = \{\wedge, a, b, aa, aaa, ..\}$

$\left(a^* + b\right)^* = \{\wedge, a, b, aa, ab, ba, bb, aaa, ..\}$

The language defined by the expression $\left(a + b\right)^*$

Set of all strings of a's and b's,

Language2 $\left(a + b\right)^*$.

$\left(a + b\right)^* = \{\wedge, a, b, aa, ab, ba, bb, aaa, ..\}$

Hence Language1 and Language2 define the same set of language.

Step 3 of 3

iii) $\left(a^* + b^*\right)^*$ and $\left(a+b\right)^*$

The language defined by the expression $\left(a^* + b^*\right)^*$

Set of all strings of a's and b's

Language1 $\left(a^* + b^*\right)^*$.

$\left(a^* + b^*\right) = \{\wedge, a, b, aa, bb, aaa, bbb..\}$

$\left(a^* + b^*\right)^* = \{\wedge, a, b, aa, ab, ba, bb, aaa, aabbaa, ..\}$

The language defined by the expression $\left(a+b\right)^*$

Set of all strings of a's and b's,

Language2 $\left(a+b\right)^*$.

$\left(a+b\right)^* = \{\wedge, a, b, aa, ab, ba, bb, aaa, aabbaa, ..\}$

Hence Language1 and Language2 define the same set of language.

16. (i) ((a + bb)*aa)* and **A** + (a + bb)*aa
(ii) (aa)*(A + a) and a*
(iii) a(aa)*(A + a)b + **b** and a*b

Step 1 of 3

I) $\wedge^*$ and $\wedge$

$\wedge^*$ consists of all strings $(\wedge)^n$ for $n = 0, 1, 2, ...$

1) $\wedge^0 = \wedge$

2) $\wedge^1 = \wedge$

Thus, $\wedge^*$ and $\wedge$ are equal

Step 2 of 3

(II) $(a^*b)^* a^*$ and $a^*(ba^*)^*$

$(a^*b)^* a^*$ consists of all strings $(a^*b)^n a^n$ for $n = 0,1,2,.....$

$a^*(ba^*)^*$ consists of all strings $a^n(ba^*)^n$ for $n = 0,1,2,.....$

Using induction, we will prove $(a^*b)^n a^n = a^n(ba^*)^n$

Assume that $(a^*b)^{n-1} a^{n-1} = a^{n-1}(ba^*)^{n-1}$

Now consider

$$(a^*b)^n a^n = (a\,b)\ (a^*b)^{n-1} a^{n-1}$$

$$= a\,b\ (a^{n-1}(ba^*)^{n-1})$$

$$= a^n(ba^*)^n$$

Thus, both the regular expressions define the same language

Step 3 of 3

(III) $(a^*bbb)^* a^*$ and $a^*(bbba^*)^*$

$(a^*bbb)^* a^*$ consists of all strings $(a^*bbb)^n a^n$ for $n = 0,1,2.....$

$a^*(bbba^*)^*$ consists of all strings $a^n(bbba^*)^n$ for $n = 0,1,2,.....$

Using induction, we will prove $(a^*bbb)^n a^n = a^n(bbba^*)^n$

Now consider

$$(a^*bbb)^n a^n = a\,bbb\ (a^*bbb)^{n-1} a^{n-1}$$

$$= a\,bbb(a^{n-1}bbba^*)^{n-1}$$

$$= a^n(bbba^*)^n \text{ for } n = 0,\ 1,\ 2,.....$$

Thus, both the regular expressions define the same language

17. (i) a(ba + a)*b and aa*b(aa*b)*
(ii) **A** + a(a + b)* + (a + b)* aa(a + b)* and ((b*a)*ab*)*
Describe (in English phrases) the languages associated with the following

regular expressions.

Step 1 of 5

(i) $((a+bb)^*aa)^*$ and $\wedge+(a+bb)^*aa$

$((a+bb)^*aa)^*$

This regular expression generates the strings the either start with 'a' or 'bb' and ends with 'aa' and also contains the empty string

$\wedge+(a+bb)^*aa$

This also generates the strings the either start with 'a' or 'bb' and ends with 'aa' along with empty string

Thus, both the regular expressions define the same language, and hence they are equal.

Step 2 of 5

(ii) $(aa)^*(\wedge+a)$ and a^*

$(aa)^*(\wedge+a)$

This regular expression generates the strings that contains all possible a's including empty string.

a^*

This also generates the strings that contains all possible a's including empty string.

Thus, both the regular expressions define the same language, and hence they are equal.

Step 3 of 5

(iii) $a(aa)^*(\wedge+a)b+b$ and a^*b

$a(aa)^*(\wedge+a)b+b$

This regular expression generates the strings that contains only exactly one 'b' or starting with all possible combinations of 'a' that end with single 'b'

a^*b

This also generates the strings that contains only exactly one 'b' or starting with all possible combinations of 'a' that end with single 'b'

Thus, both the regular expressions define the same language, and hence they are equal.

Step 4 of 5

(iv) a (ba + a)*b and aa*b (aa* b)*

a (ba + a)*b

This regular expression generates the strings that contains all combinations of 'ba' or 'a' that start with 'a' and end with 'b'

aa*b (aa* b)*

This also generates the strings that start with 'a' and end with 'b' and all possible combinations of a's or ba's

Thus, both the regular expressions define the same language, and hence they are equal.

Step 5 of 5

(v) ∧ + a(a + b)* + (a + b)* aa(a + b)* and ((b*a) ab*)*

∧ + a(a + b)* + (a + b)* aa(a + b)*

This regular expression generates all possible words that contains a's or b's

((b*a) ab*)*

This also generates all possible words that contains a's or b's

Thus, both the regular expressions define the same language, and hence they are equal.

18. (i) (a + **b)*** a(A + **bbbb)**
(ii) (a(a + **bb)*)***
(iii) (a(aa)*b(bb)*)*
(iv) (b(bb)*)*(a(aa)*b(bb)*)*
(v) (b(bb)*)*(a(aa)*b(bb)*)*(a(aa)*)*
(vi) ((a + b)a)*

Step 1 of 6

i) Any string that ends in 'a' or abbbb.

Provide feedback (0)

Step 2 of 6

ii) All words that do not begin with 'b' and in which b's appear in clumps of even length

Provide feedback (0)

Step 3 of 6

iii) All words that contains odd number of a's followed by any odd number of b's or a null string. The series begins with a's and ends with b's

Step 4 of 6

iv) All words that start with 0 or more odd number of b's followed by odd number of a's that end in odd number of b's, which also includes an empty string.

Provide feedback (0)

Step 5 of 6

v) All words with all b's, in odd numbers, or words that is all a's, in odd number, or any word that has odd groups of a's and b's. This language also includes the null string.

Provide feedback (0)

Step 6 of 6

vi) All words with even lengths and which start in a's or b's and end in a's. This language also includes null string.

19. (D.N. Arden) Let *R*, ***S,*** and *T* be three languages and assume that. A is not in S. Prove the following statements.
(i) From the premise that R = *SR* + T, we can conclude that R = *S*T.*
(ii) From the premise that R = *S*T,* we can conclude that R = *SR* + *T.*

20. Explain why we can take any pair of equivalent regular expressions and replace the letter a in both with any regular expression R and the letter b with any regular expression S and the resulting regular expressions will have the same language. For example, 15.(ii)

(a*b)*a* = a*(ba*)*
becomes the identity
(R*S)*R* = R*(SR*)*
which is true for all regular expressions R and S. In particular
R = a + **bb, S** = ba* results in the complicated identity
((a + bb)*(ba*))*(a + bb)* = (a + **bb)*** ((ba*)(a + **bb)*)***
What is the deeper meaning of this transformation?
What identity would result from using
R = (ba*)* **S** = (A + b)

Step 1 of 2

I) Because the regular expressions are closed under the following properties, there will not be any change in replacing letter by regular expression

1) The distributive property

2) The associative property

3) The commutative property

4) The identity property

5) The order of operations including exponents and nested parentheses

6) The combination of like terms

Step 2 of 2

II)

$$(((ba^*)^*)^*(\wedge+b))^*((ba^*)^*)^* = ((ba^*)^*)^*((\wedge+b)((ba^*)^*)^*)^*$$

$$((ba^*)(\wedge+b))^*(ba^*) = ((ba^*)((\wedge+b)(ba^*))$$

Chapter 5

PROBLEMS
1. Write out the transition table for the FA's on pages 68, 70 (both), 73, 74 and 80 that were defined by pictures. If the states in the pictures were not labeled, assign them names so that we can build the table.

Step 1 of 9

P: 56

Transition table:

	a	b
Start x-	y	z +
y	-x	z +
Final z +	z +	z +

Step 2 of 9

P: 58

Transition table:

	a	b
Start -	+	+
Final +	+	+

Step 3 of 9

Transition table:

	a	b
Start final ±	±	±

Step 4 of 9

P: 63

Transition table:

	a	b
Start 1 -	2	3
2	4 +	3
3	2	4 +
Final 4 +	4 +	4 +

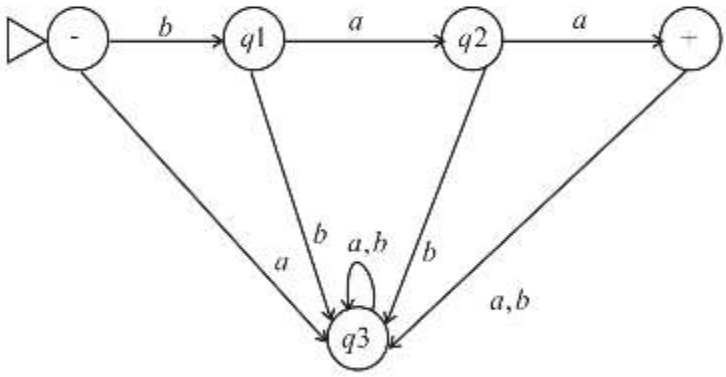
Step 5 of 9

Transition table:

	a	b
Start 1 -	2	2
2	3	3
3	4	5 +
4	4	4
Final 5 +	5 +	5 +

Step 6 of 9

Transition graph:



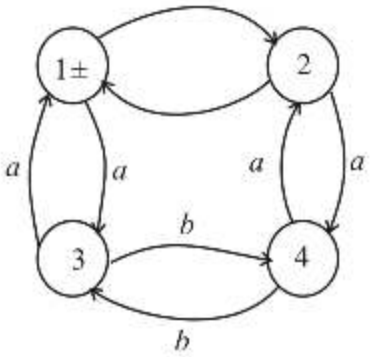
Step 7 of 9

Transition table:

	a	b
Start -	$q3$	$q1$
$q1$	$q2$	$q3$
$q3$	$q3$	$q3$
$q2$	+	$q3$
Final +	$q3$	$q3$

Step 8 of 9

P: 69:



Provide feedback (1)

Step 9 of 9

Transition table:

	a	b
Start final 1±	3	2
2	4	1±
3	1±	4
4	2	3

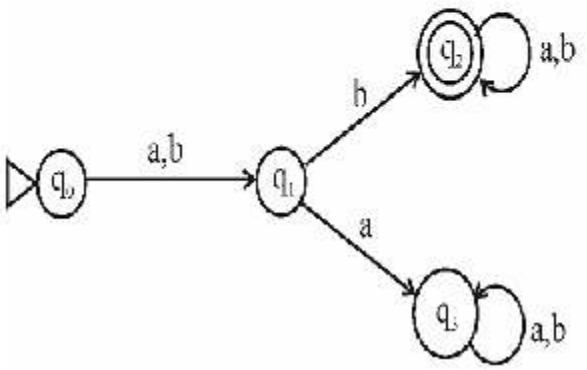
2. Build an FA that accepts only the language of all words with b as the second letter. Show both the picture and the transition table for this machine and find a regular expression for the language.

Step 1 of 2

Regular expression for the words with 'b' as the second letter:

$(a+b)b(a+b)*$

Transition Diagram:



Step 2 of 2

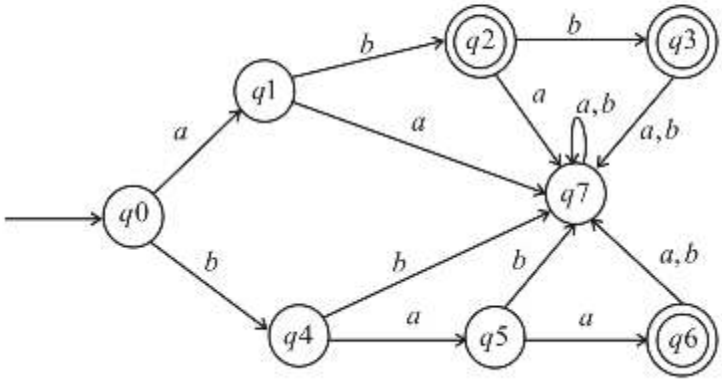
Transition table:

	a	b
Start $q0$	$q1$	$q1$
$q1$	$q3$	$q2$
Final $q2$	$q2$	$q2$
$q3$	$q3$	$q3$

3. Build an FA that accepts only the words *baa*, *ab*, and *abb* and no other strings longer or shorter.

Step 1 of 1

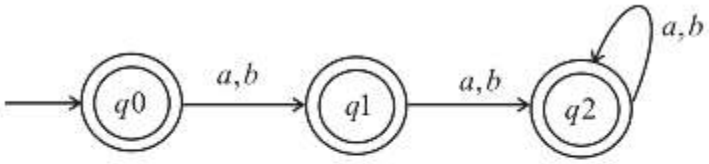
Finite automata:



4. (i) Build a new FA that accepts only the word A.
(ii) Build an FA with three states that accept all words.

Step 1 of 4

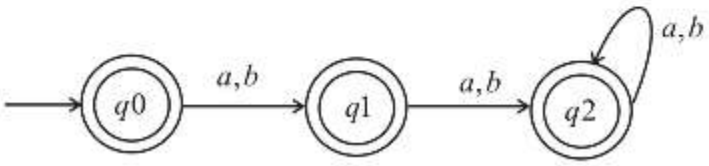
I)



Provide feedback (1)

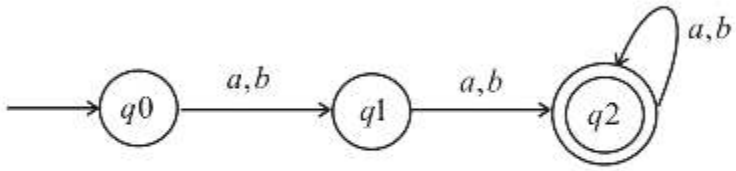
Step 2 of 4

II)



Step 3 of 4

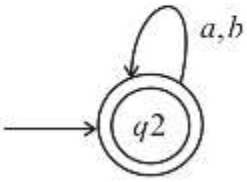
III)

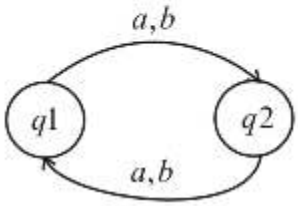


Provide feedback (0)

Step 4 of 4

In this FA we have only one final state $q2$. It rejects null, a, b, etc. so it reject some inputs proved!
But situation like this will never reject any input

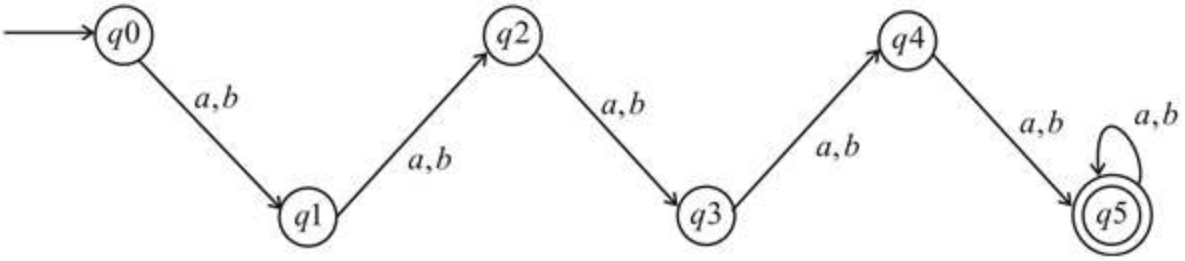




5. Build an FA that accepts only those words that have an even number of letters total.

Step 1 of 3

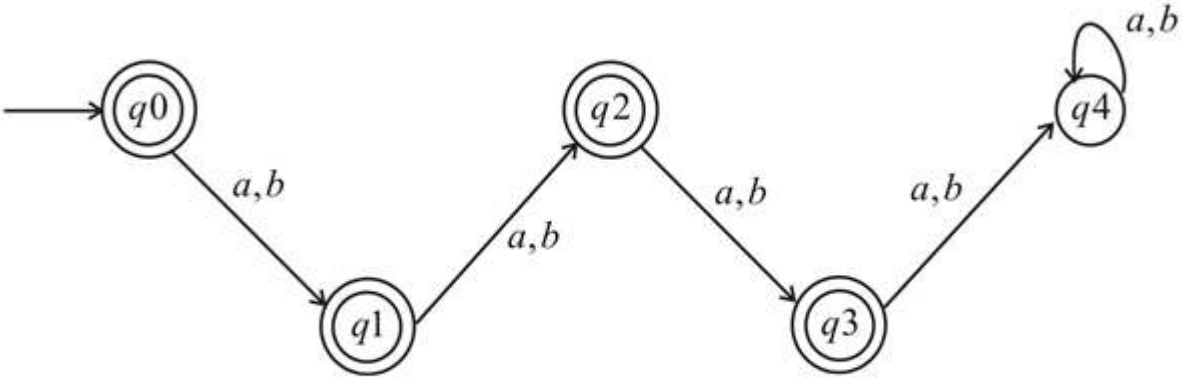
I)



The above FA accepts four letters up to q4 state. After q4, q5 accepts another word while after q5, the kleene closure of {a, b} to q5 allows it to accept any set of words including null string. Hence, the above FA has more than four words.

Step 2 of 3

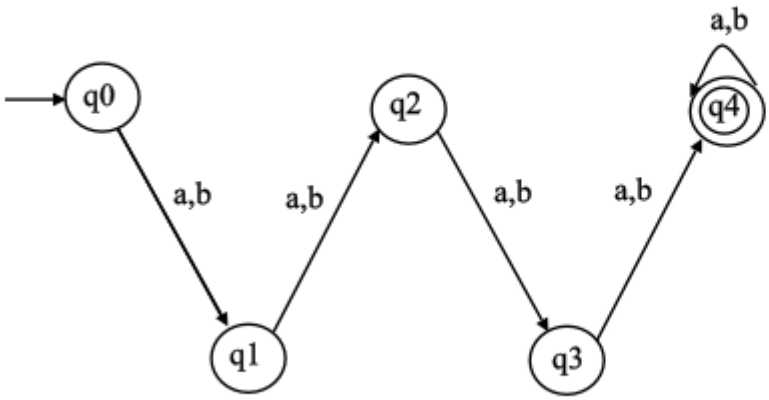
II)



The above FA has four accepting or terminating states. If the control terminates at q0, no word will be present in the language. If termination takes place at q1, the language will have one word. Similarly, at q2, language has two words and at q3, the language has three words. So far, all possible words will have lengths less than four. Even if the control does not terminate at q3, the maximum word length up to q4 is four.

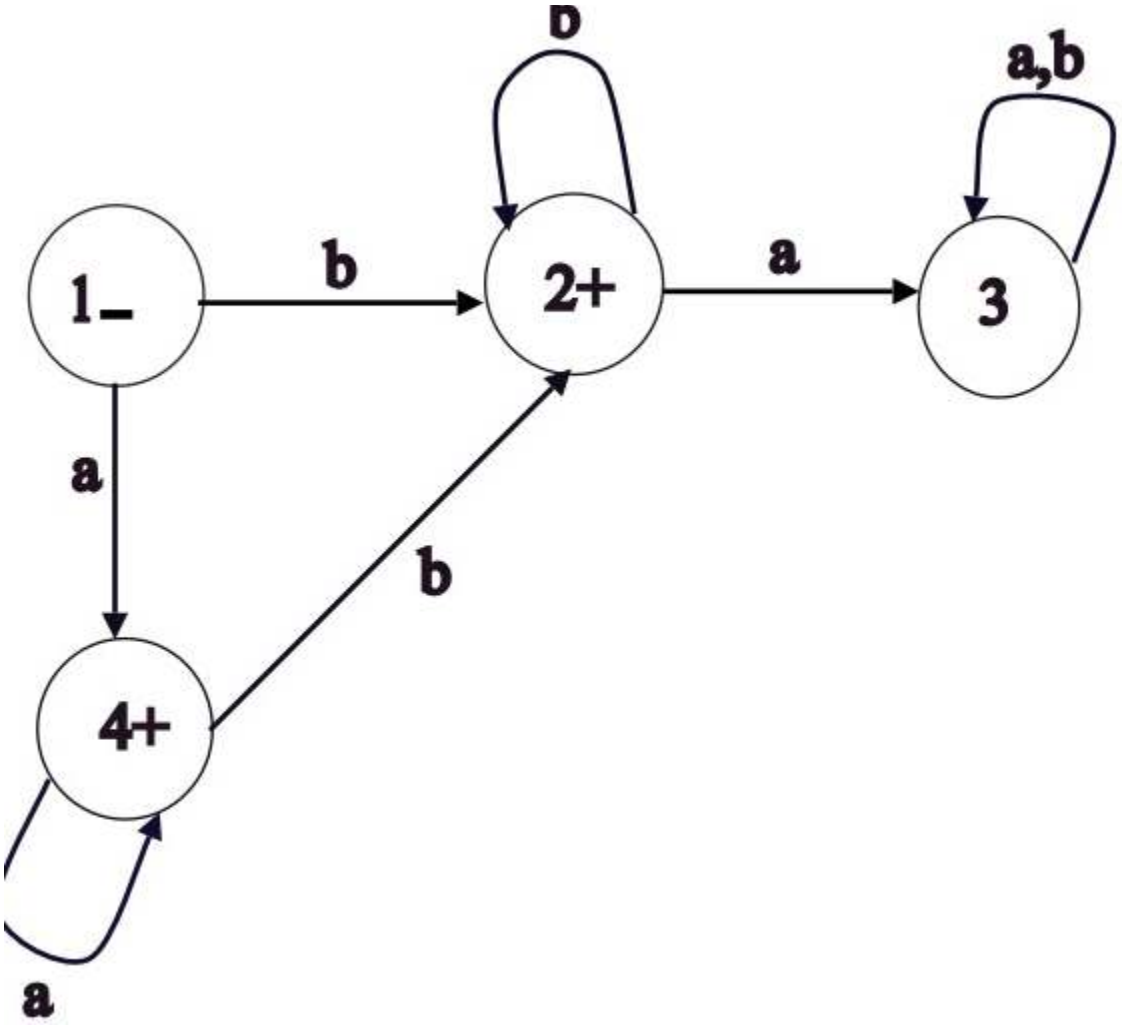
Step 3 of 3

III)



The above FA accepts exactly four letters as, on reaching q4, the inputs to the FA are going to be terminated.

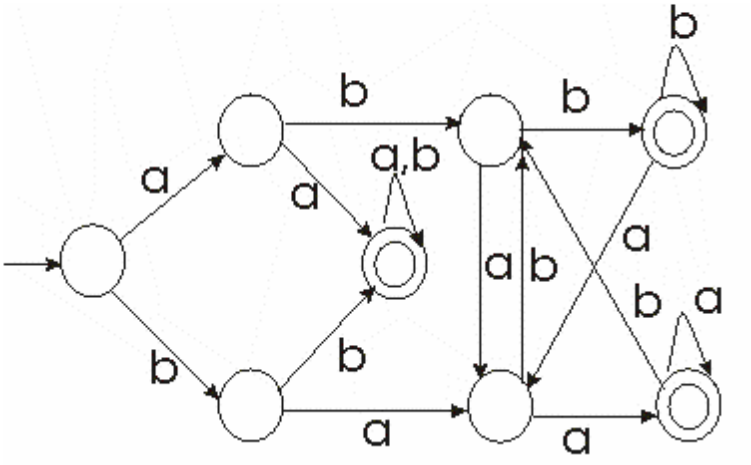
6. Build an FA that accepts only those words that do not end with *ba*.



7. Build an FA that accepts only those words that begin or end with a double letter.

Step 1 of 1

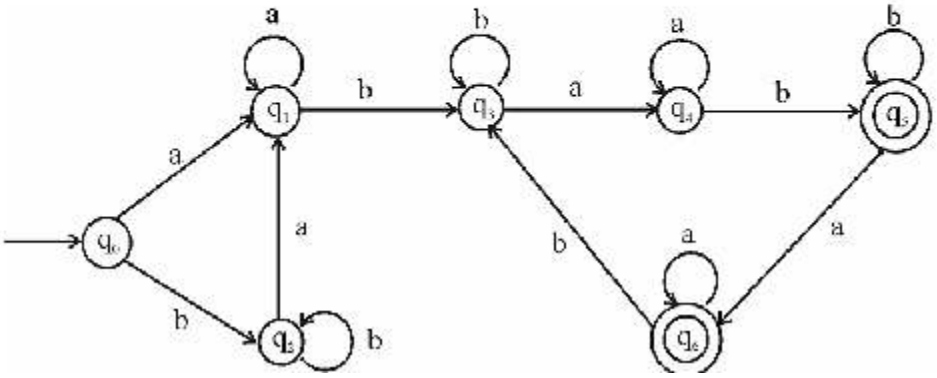
Many possibilities are there, we are giving one example



8. (i) Build an FA that accepts only those words that have more than four letters.
(ii) Build an FA that accepts only those words that have fewer than four letters.

Step 1 of 1

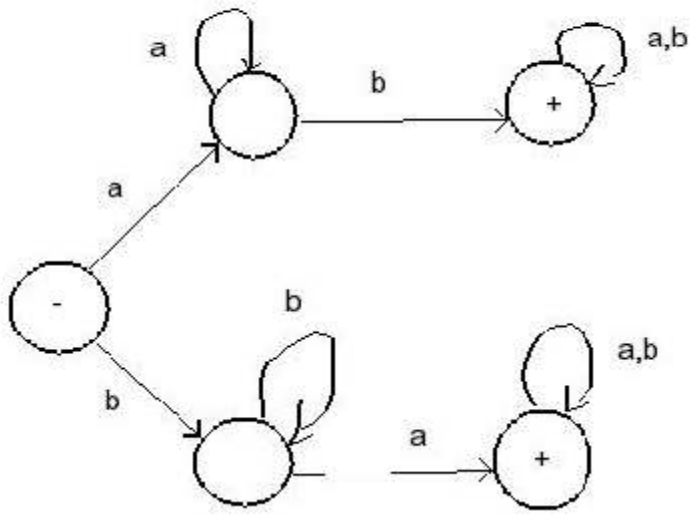
Many possibilities are there, we are giving one example. The FA that accepts only those words that have an even number of substrings 'ab'



9. Problems 2 through 12 of Chapter 4 include 14 languages that could be represented by regular expressions. For each of these find an FA that accepts exactly it.

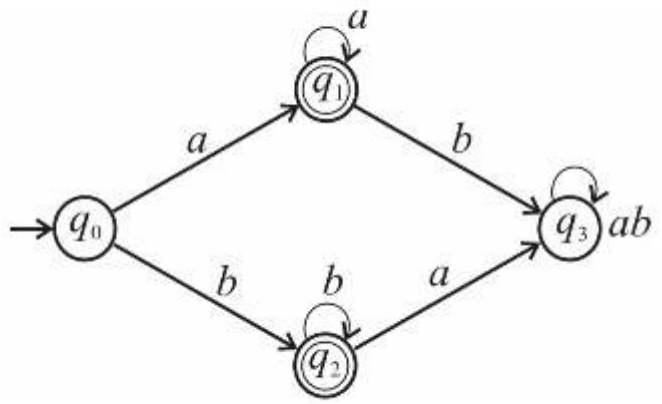
Step 1 of 2

I) The proper FA should be: All the words that have both the letter a and b. Where + denotes the final state and – denotes the start state.



Step 2 of 2

II) Words contain only a's or b's



Regular expression: $(aa^* + bb^*)$

10. So far we have been dealing with Fa's over the alphabet. {a, *b}*. Let us consider for the moment the alphabet **I** - {a, *b*, *c}*.
(i) If we had an FA over this alphabet with five states, how many entries would there be in the transition table?
(ii) In the picture of this five-state machine, how many edges would need to be drawn in total (counting an edge with two labels double and an edge with three labels triple)?
(iii) Build an FA that accepts all the words in this alphabet that have an a in them somewhere that is followed later in the word by some b that is followed later in the word by some c (the three being not necessarily in a row but in that order, as in *abaac).*
(iv) Write a regular expression for the language accepted by this machine.

11. Recall from Chapter 4 the language of all words over the alphabet {a, *b}* that have both the letter a and the letter b in them, but not necessarily in that order. Build an FA that accepts this language.

Step 1 of 1

$2^3 \times 3^6 = 5832$ different automata on three states acting on two letters. Obvious symmetries such as permutation of states, permutation of letters, and inversion of states, together with minimization, reduces the number of automata that needs to checked to 194.

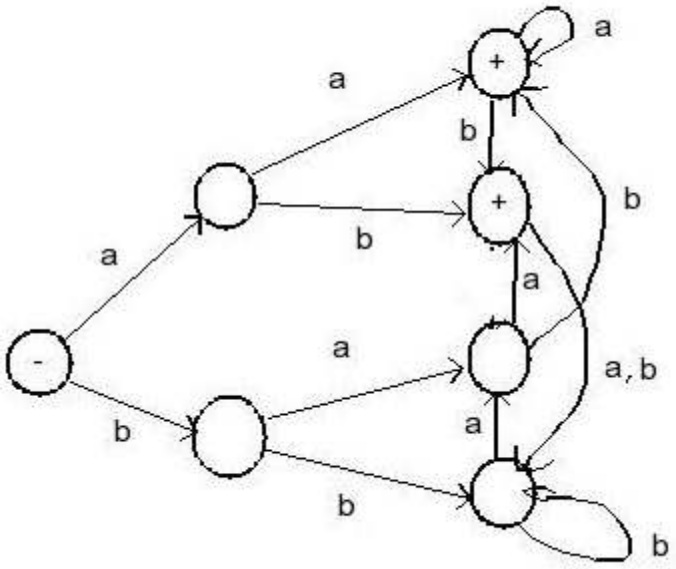
12. Build an FA that accepts the language of all words with only a's or

only b's in them. Give a regular expression for this language.

13. Draw pictures for all the FA's over the alphabet {a, *b}* that have exactly two states. Be careful to put the +'s in in all possible ways. *(Hint:* There are 48 different machines.)
14. (i) Write out the transition tables for all the FA's in Problem 13.
(ii) Write out regular expressions to represent all the languages defined by the machines in Problem 13.

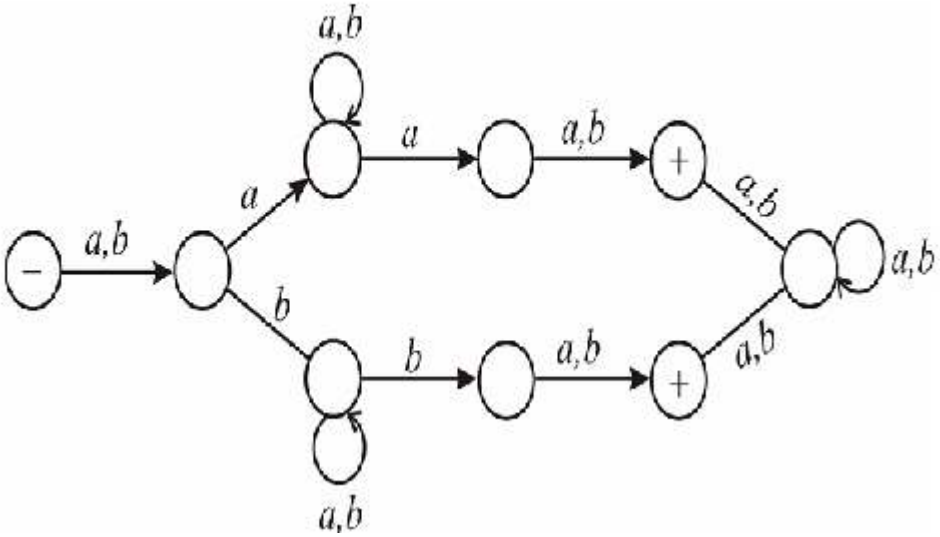
Step 1 of 2

I)



Step 2 of 2

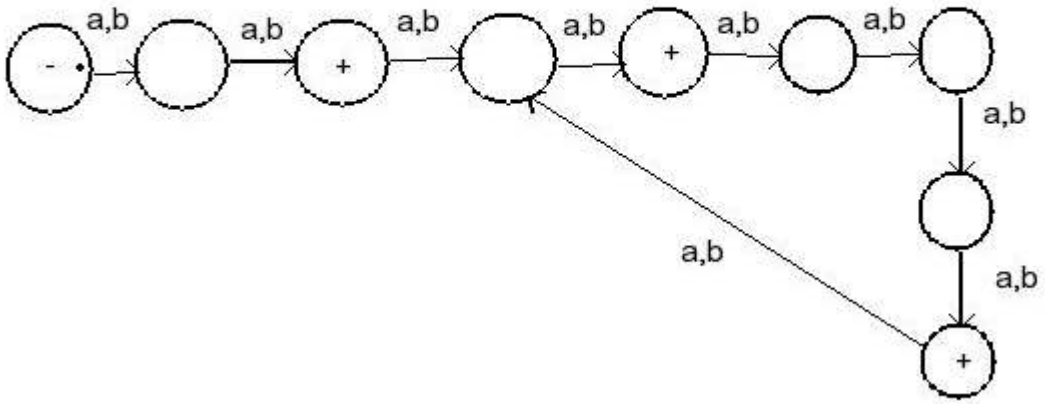
II)



15. Let us call two FA's *different* if their pictures are not the same but *equivalent* if they accept the language. How many different languages are represented by the 48 machines of Problem 13.

Step 1 of 1

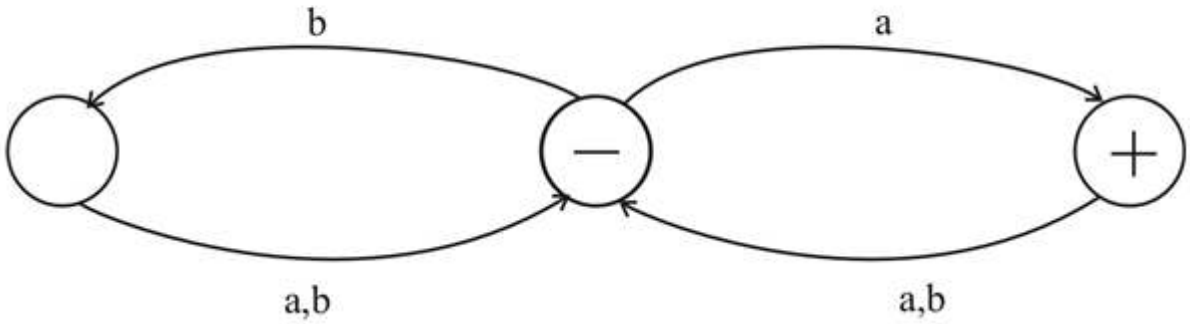
We must build a FA that accepts strings of even length, not divisible by 6:



16. Show that there are exactly
36(8) = 5832
different finite automata with three states x, y, *z* over the alphabet *{a,b}* where x is always the start state.

Step 1 of 3

Old FA:



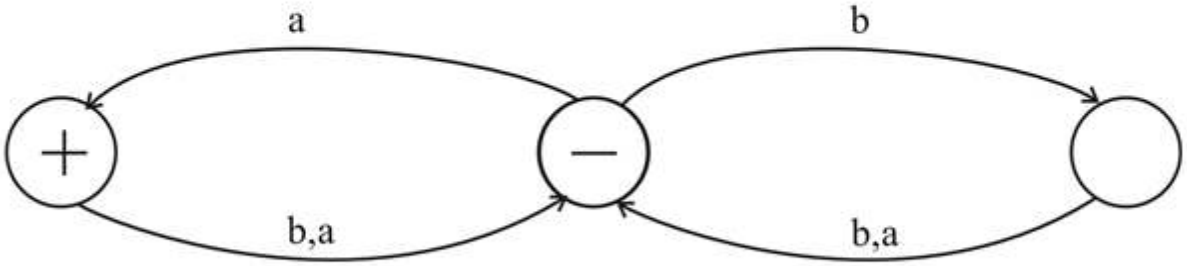
Step 2 of 3

Language generated by this FA is, L = { a, aba, aaa, bba, ababa, ... }

Provide feedback (0)

Step 3 of 3

New FA (after swapping the labels and also the states):



The machine has been changed by changing the positions of states and labels. But, the output of the new FA is same as the output of the old FA.

Language generated by this new FA is, L = { a, aba, aaa, bba,.....}

Therefore the two FA's generate the same language

FINITE AUTOMATA 83

17. Find two FA's that satisfy these conditions: Between them they accept all words in (a + b)*, but there is no word accepted by both machines.

Step 1 of 4

i) Second state is initial, the FA accepts the language L of all odd length words over {a, b} that end with a.

Provide feedback (0)

Step 2 of 4

ii) First state is initial, the FA that accepts words of at least length two, even length strings that end with 'a'.

Step 3 of 4

iii) First state is initial, the FA will accept words of at least length two, even length of strings that ends with 'a' and will not accept consecutive b's.

Provide feedback (4)

Step 4 of 4

iv) i. (aa + ab + ba + bb)*a

ii. (a+b) [(b(a+b))*(a(a+b))*]*a.

iii. (a+b)a((a+b)a)*

18. Describe the languages accepted by the following FA's.

iv) Write regular expressions for the languages accepted by these three machines.

Step 1 of 1

Assume 1-, 2, 3, 4+, 5+, 6+ states as q0,q1,q2,q3,q4,q5 sequentially. Where q0 is the start state and q3, q4, q5 are final states.

Passing string abcabcabc

The given FA will accept this string by going

$q0 \to q1 \to q2 \to q3 \to q4 \to q5 \to q0 \to q1 \to q2 \to q3$

Passing string abcabc

The given FA will not accept this string by going

$q0 \to q1 \to q2 \to q3 \to q4 \to q5 \to q0$,

Where $q0$ is not the final state.

19. The following is an FA over the alphabet I = {a, *b*, *c}*. Prove that it accepts all strings that have an odd number of occurrences of the substring *abc*.

Step 1 of 6

I) Taking input of 'abab' will take us to state q15 which is a dummy state and the string is not accepted by the FA

Provide feedback (0)

Step 2 of 6

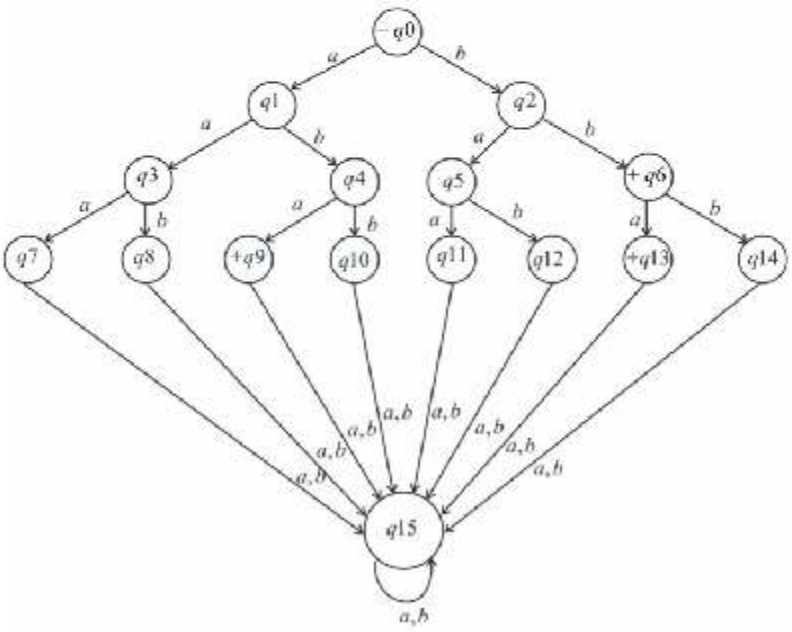
II) Starting from q0 and reading a will we go to q1 which is a final stat so the word 'a' is accepted by this language similarly 'aab' and 'bab' words area also accepted by this language, passing any other word will take us to a state which will be either non-final (or) dummy state, and the word will not be accepted.

Step 3 of 6

III) As this FA is change now it accepts the only words bb, bba, aba,
No other word than these will be accepted by this FA

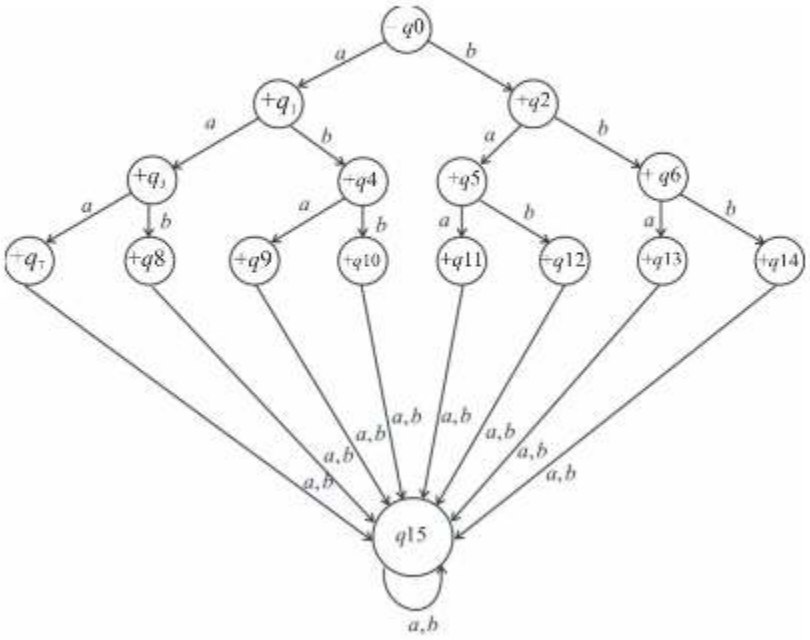
Provide feedback (0)

Step 4 of 6



Step 5 of 6

IV) The FA that accepts any language fewer than four letters is as given below,



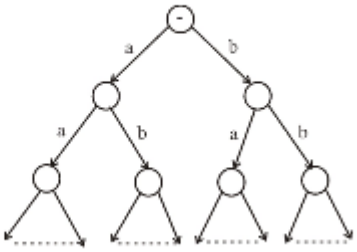
Step 6 of 6

V) As the language accepts finite no. of words in a finite language. So there can be another FA by extending more layers and mentioning length of words accepted (or) just giving words along with.

20. Consider the following FA:

(i) Show that any input string with more than three letters is not accepted by this FA.
(ii) Show that the only words accepted are *a*, *aab*, and *bab*.
(iii) Show that by changing the + signs alone we can make this FA accept the language *{bb*, *aba*, *bba}*
(iv) Show that any language in which the words have fewer than four letters can be accepted by a machine that looks like this one with the + signs in different places.
(v) Prove that if L is a finite language, then there is some FA that accepts *L*.

Step 1 of 4

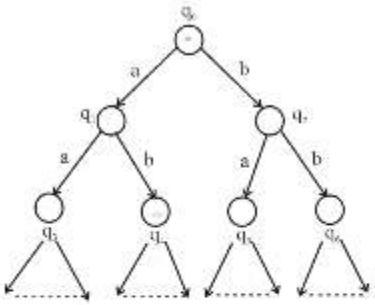


Provide feedback (0)

Step 2 of 4

Above tree energy state contains 2 – edges.
Suppose $q_0, q_1, q_2 \ldots\ldots\ldots\ldots q_n$ all states contains the edges a,b
L be infinite language of strings a's and b's
Let us assume to place '+' final state in q_4 state in the infinite tree

Step 3 of 4



Provide feedback (0)

Step 4 of 4

Now, the infinite tree accepts only the string 'ab'.
Suppose, we place final state in (+) q_3 position the tree accepts only the string 'aa'.
In this infinite tree the edges are going one state other states like.
$q_0 \rightarrow q_1 \rightarrow q_3$ like this way.
So, we can't place final states in all states in the infinite tree.
That the reason, this machine would not be a satisfactory language definer for 'L'

CHAPTER **6**

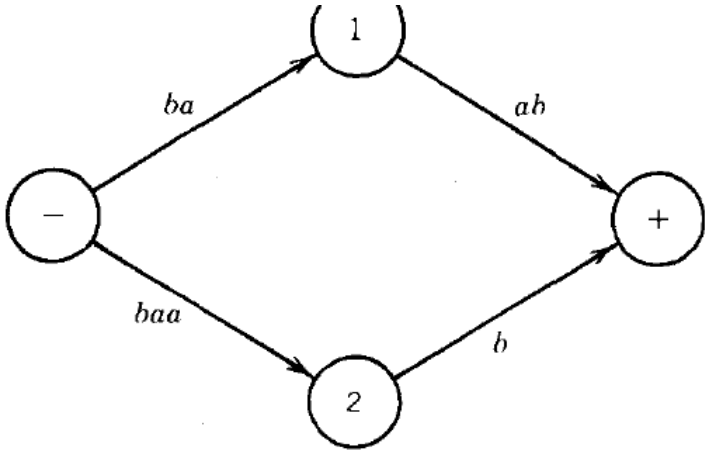
PROBLEMS

1. For the four FA's pictured in Problems 5-18, 5-19 and 5-20 determine whether a TG could be built that can accept the same language but requires fewer states.

Step 1 of 1

The three transition graphs generated languages are equal to the finite automata generated languages.

2. The notion of transition table can be extended to TG's. The rows of the table would be the states of the machine and the columns of the table would be all those strings of alphabet characters that are ever used as the label for any edge in the TG. However, the mere fact that a certain string is the label for an edge coming from state **I** does not mean that it is also the label of an edge coming out of state 2. Therefore, in the transition table some entries are likely to be blank (that is, no new state is reached from the prior state given this input sequence). The TG



discussed in this section has the following transition table:

	b	*ab*	*ba*	*baa*
–			1	2
1		+		
2	+			
+				

Calculate the transition table for the TG's defined by pictures on pages 86, 87, 89 (bottom), 90, 91 (third), 93 (second), and 94.
One advantage of defining a TG by such a table is that in complicated cases it may be easier to read the table than a cluttered picture having many edges with long string labels. (Remember that in cases where not all the states have names it is necessary to give them names to build the table.)

Step 1 of 10

The machines that accept these words are

i) A = TG2, TG5

Provide feedback (0)

Step 2 of 10

ii) a = TG4

Step 3 of 10

iii) b = TG1, TG2, TG4

Provide feedback (0)

Step 4 of 10

iv) aa = TG1, TG3, TG5, TG6

Provide feedback (0)

Step 5 of 10

v) ab = TG1, TG2, TG4, TG6

Provide feedback (0)

Step 6 of 10

vi) aba = TG1, TG5, TG6

Step 7 of 10

vii) abba = TG1, TG5, TG6

Provide feedback (0)

Step 8 of 10

viii) bab = TG5

Provide feedback (0)

Step 9 of 10

ix) baab = TG5

Provide feedback (0)

Step 10 of 10

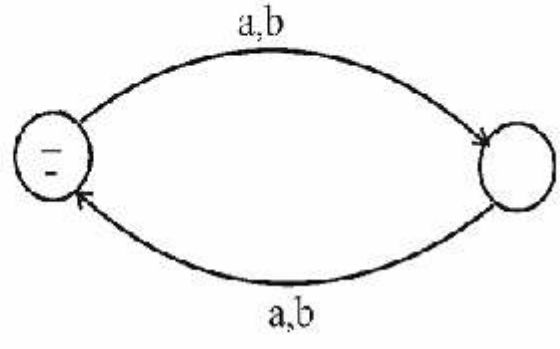
x) abbb = TG3, TG4, TG6

3. Draw a four-state TG that accepts all the input strings from {a, *b}** that

are *not* in EVEN-EVEN. Is there a two-state TG that accepts this language?

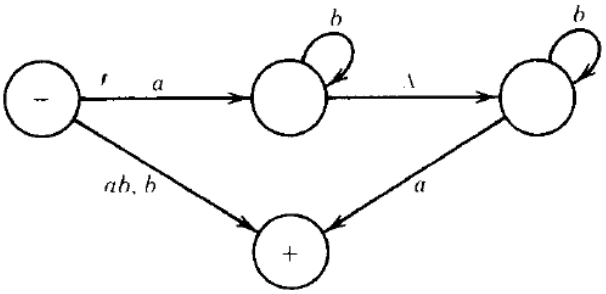
Step 1 of 1

Accept even number of states

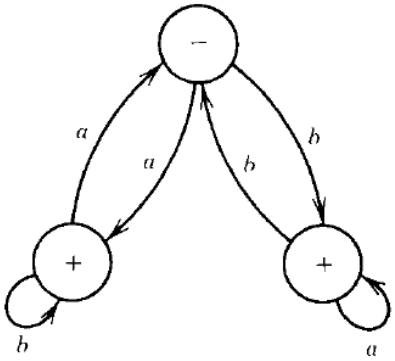


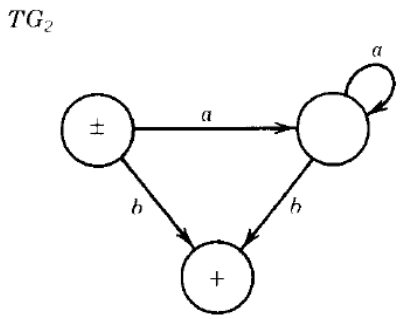
4. Here are six TG's. For each of the next 10 words decide which of these machines accepts the given word.

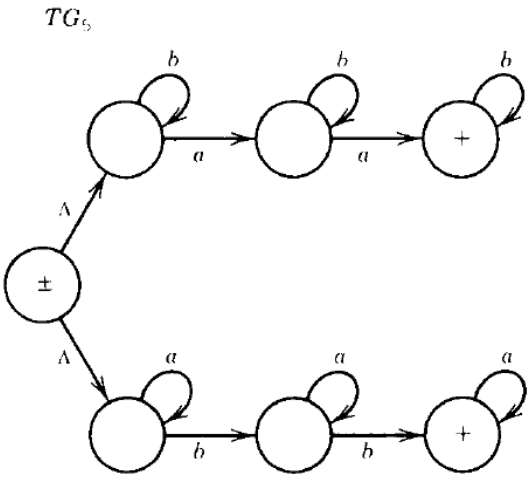
TG_1

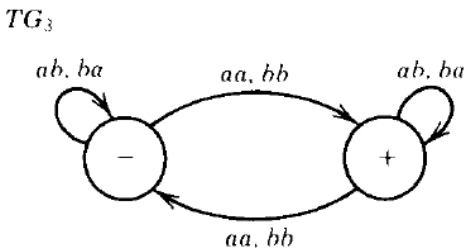


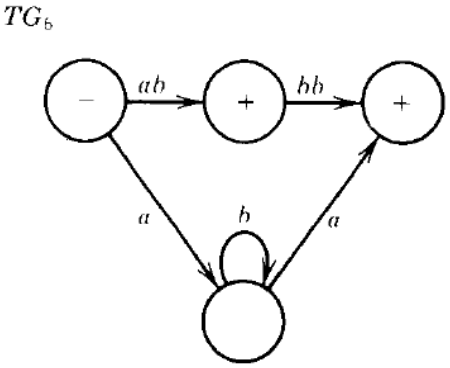
TG_4











(i) A (vi) *aba*
(ii) a (vii) *abba*
(iii) b (viii) *bab*
'(iv) aa (ix) *baab*
(v) *ab* (x) *abbb*

Step 1 of 1

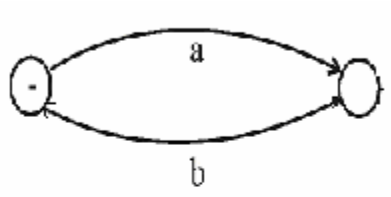
1) Transition Graph

2) Generalized Transition Graph (GTG)

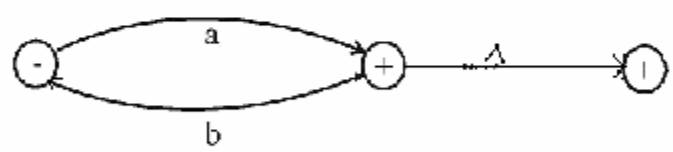
5. Find regular expressions defining the language accepted by each of the six TG's above.

Step 1 of 2

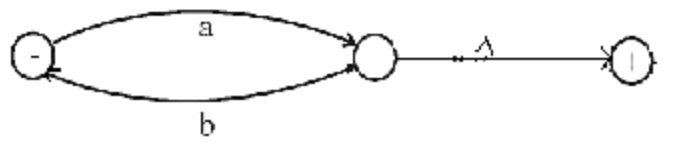
i. Create a new TG, say TG2, that has all the states and transitions of TG and a new accepting state.



ii. Connect each + state to this new accepting state via $\wedge$ transitions.



iii. Remove the +'s from each original accepting state.



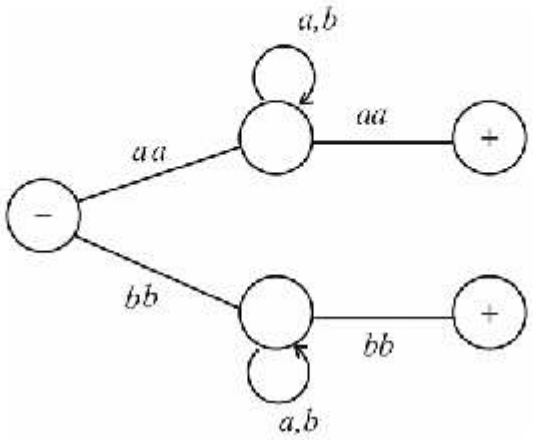
Step 2 of 2

- If a string is accepted by TG then it will leave TG2 in a state that is connected to the new + state via a $\wedge$ transition and so TG2 will accept it as well.
- If a string is rejected by TG then it will leave TG2 in some state that is not accepting and is not connected to the + state, therefore TG2 will also reject it.
- Since TG and TG2 accept the same strings and reject the same strings TG2 accepts the same language as the original TG.

6. Show that any language that can be accepted by a TG can be accepted by a TG with an even number of states.

Step 1 of 1

The TG that accepts the language of all words that begins and ends with double letter



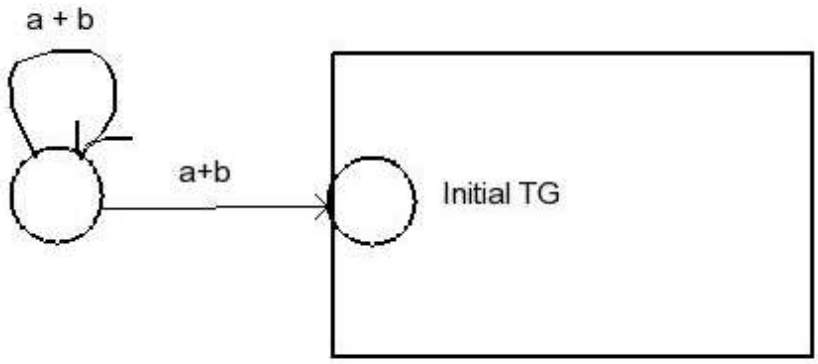
7. How many different TG's are there over the alphabet {a, *b}* that have two states?

Step 1 of 1

Let's pose the problem a bit different to make it a general case:

if OURSPONSOR is a set of strings accepted by a given TG, prove that a new TG over the same alphabet can be created, which will contain all strings which end with words from OURSPONSOR.

The problem might be easily solved by adding the following transition:



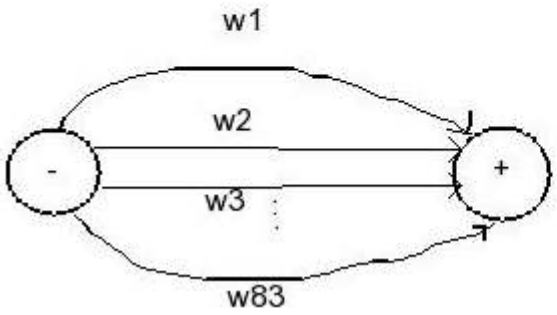
this will add all possible beginnings of strings ending with words from initial TG (OURSPONSOR let's say)

8. Show that for every finite language L there is a TG that accepts exactly

the words in L and no others. Contrast this with Theorem 5.

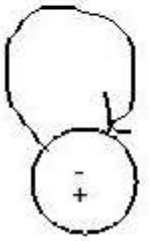
Step 1 of 1

Fo a given list of words w_1 w_2 ... w_{83} we can construct a TG that accepts all these words:



ii) The fewest possibe number states' diagram would be:

w1 + w2 + ... +w83



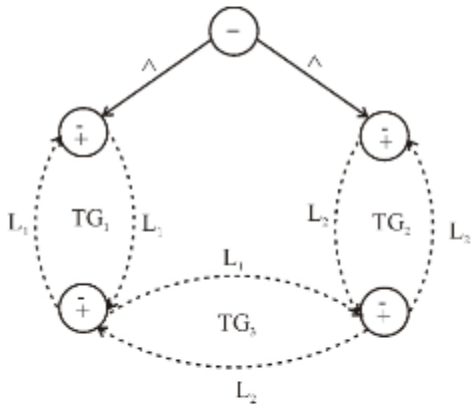
9. Prove that for every TG there is another TG that accepts the same language but has only one + state.

Step 1 of 1

Let TG_1 and TG_2 are two transition graphs.

The L_1 and L_2 are two languages that are accepted by TG1 and TG2.

Now, let us consider a transition graph TG3.
In order to accept language L1 + L2, TG3 is constructed as follows,

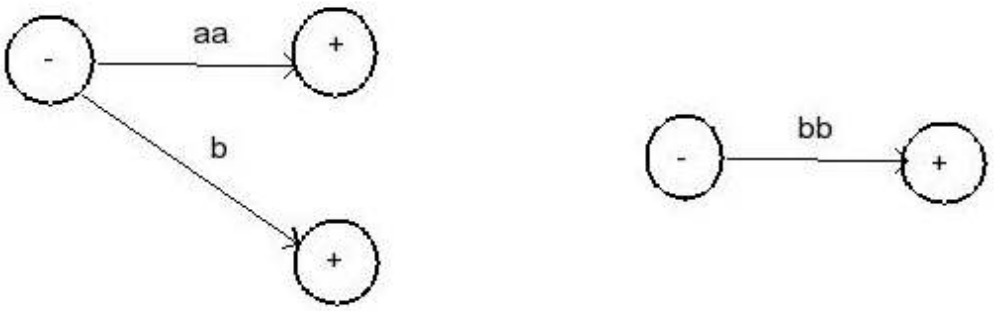


Thus, TG_1 accepts the language L_1, TG_2 accepts the language L_2. And finally the new transition graph TG_3 accepts the language (L_1+L_2).

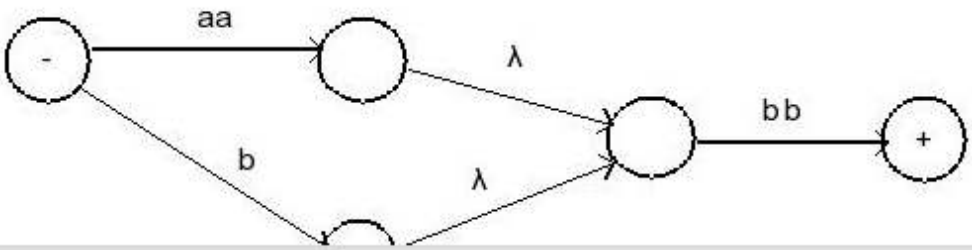
10. Build a TG that accepts the language L, of all words that begin and end with the same doubled letter, either of the form aa . . . aa or *bb* . . . *bb*. Note: aaa and *bbb* are not words in this language.

Step 1 of 1

In order to acquire a product language of two transition graphs, firstly we create lambda transitions from all final states of the first graph and connect them with initial state of the second TG. Then, we remove final states of the first graph and initial state of the other one:



would become



11. Build a TG that accepts the language of all strings that end in a word from *L1* of Problem 10 above.

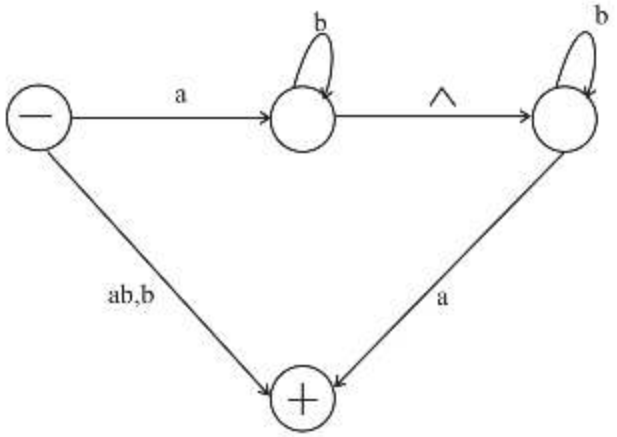
Step 1 of 1

1. The only difference is that there is not a new, non-reenterable start state that is accepting.
2. Similar to how a TG that accepts L is modified to accept L*
3. By connecting every accepting state back to the start state the new machine will accept L+, strings formed by one or more occurrences of strings in L concatenated together.

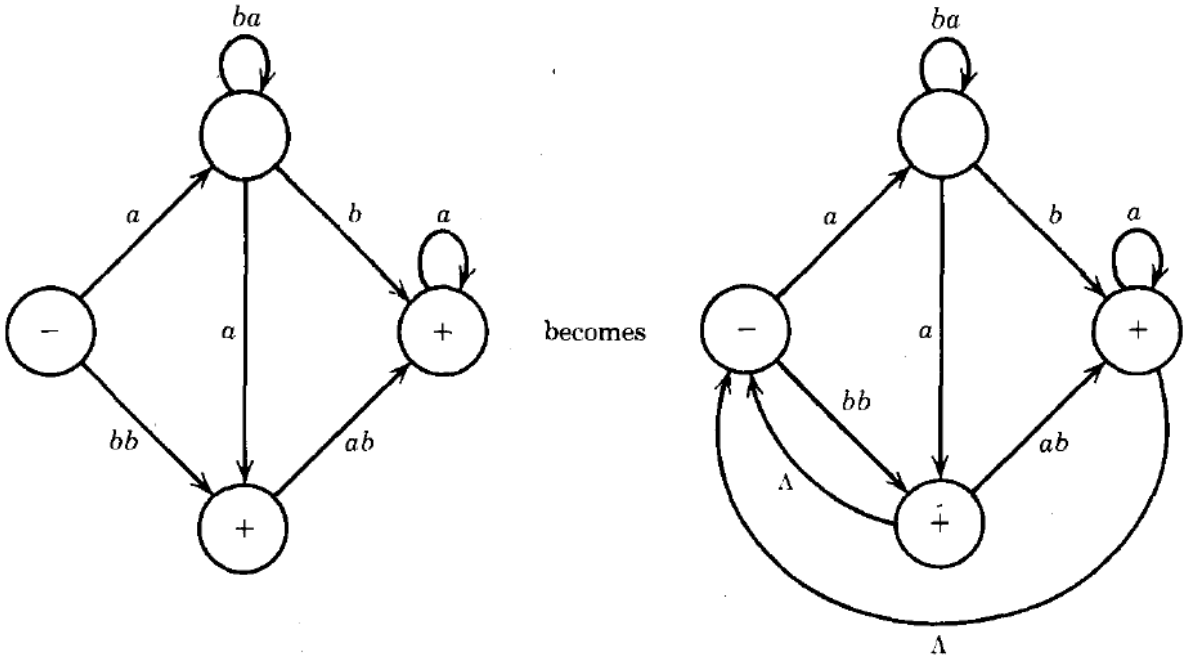
12. If OURSPONSOR is a language that is accepted by a TG called Henry, prove that there is a TG that accepts the language of all strings of a's and b's that end in a word from OURSPONSOR.

Step 1 of 1

D)



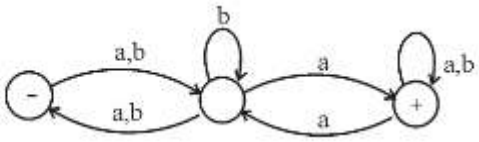
13. Given a TG for some arbitrary language L, what language would it accept if every + state were to be connected back to every - state by Aedges?
For example, by this method:



Hinta Why is the answer not always *L*?*

Step 1 of 1

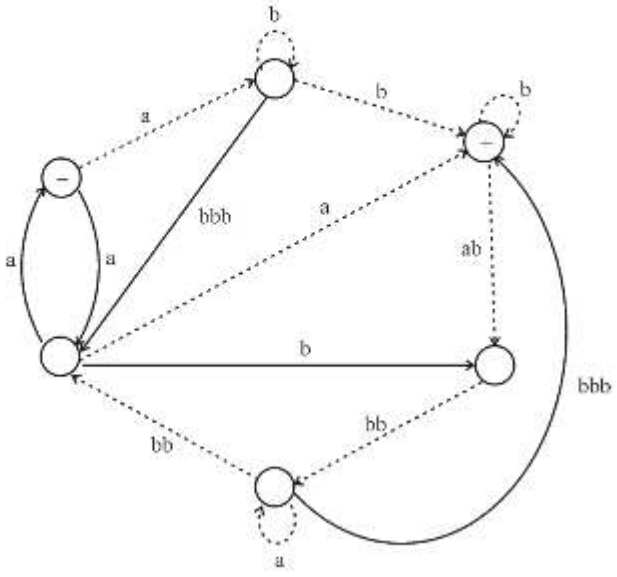
The TG (Transition graph) is generated the L^* language.



14. Let the language *L* be accepted by the finite automaton *F* and let *L* not contain the word A. Show how to build a new finite automaton that accepts exactly all the words in L and the word A.

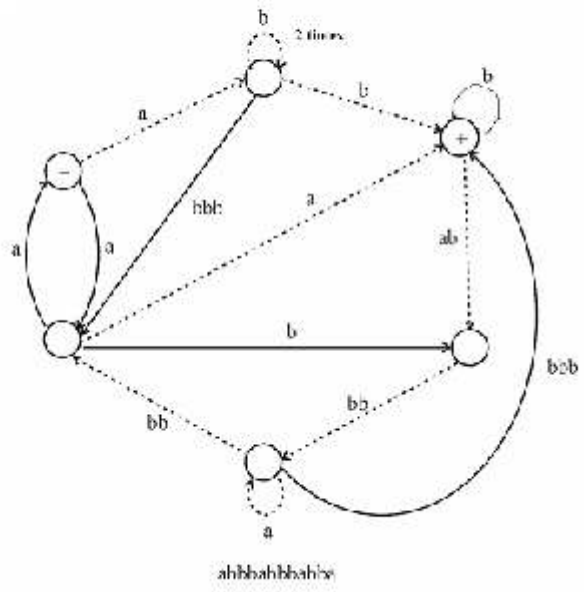
Step 1 of 3

1) Accept the string " abbbabbbabba "



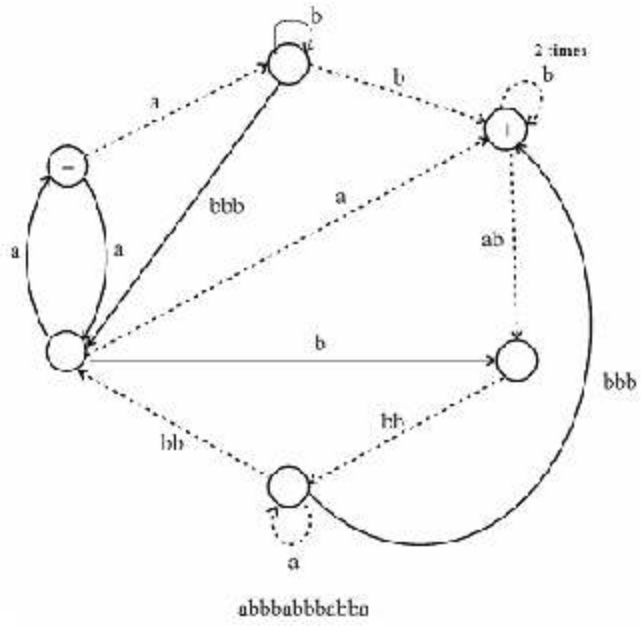
Step 2 of 3

2) Accept the string " abbbabbbabba "



Step 3 of 3

3) Accept the string " abbbabbbabba "



15. Let the language L be accepted by the transition graph T and let L not

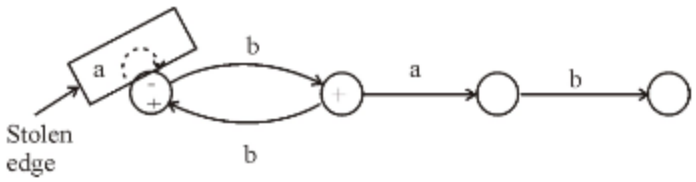
contain the word A. Show how to build a new TG that accepts exactly all the words in L and the word A.

Step 1 of 8

When vandals come and stole an edge labeled 'a'.
The resulted TG was accepted exactly the language b^*.

Provide feedback (3)

Step 2 of 8

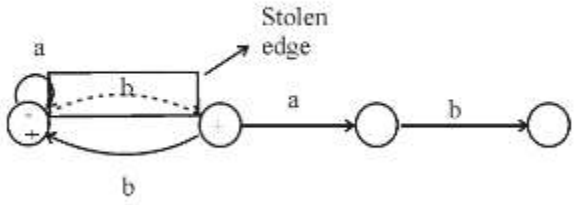


Provide feedback (0)

Step 3 of 8

The above TG accepts only b^*.
Next time vandals come and stole an edge labeled 'b'.
The resulted TG was accepted exactly the language a^*.

Step 4 of 8

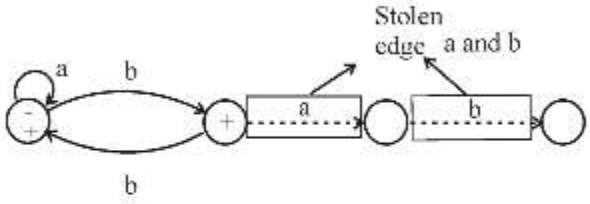


Provide feedback (0)

Step 5 of 8

The above TG accepts only a^*.
next time vandals come and stole both edges a and b.
the resulted TG was accepted exactly a^*+b^*.

Step 6 of 8



Provide feedback (0)

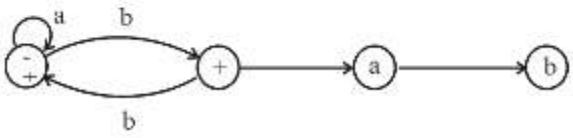
Step 7 of 8

The above TG accepts only $a^* + b^*$

Provide feedback (0)

Step 8 of 8

Original FA was

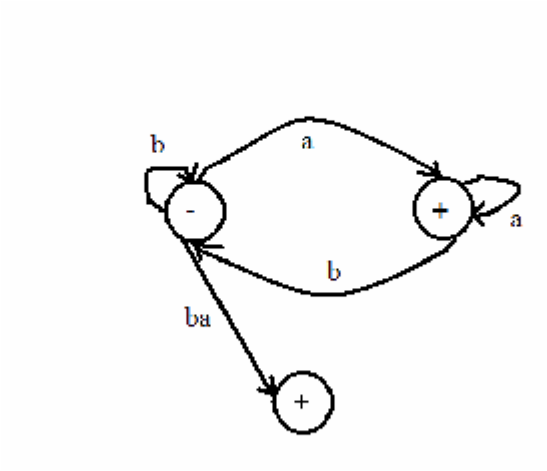


16. Let the language L be accepted by the transition graph T and let L not contain the word *ba*. We want to build a new TG that accepts exactly L and the word *ba*.
(i) One suggestion is to draw an edge from - to + and label it *ba*. Show that this does not always work.
(ii) Another suggestion is to draw a new + state and draw an edge from a - state to it labeled ba. Show that this does not always work.
(iii) What does work?

Step 1 of 1

To built a TG that accepts exactly 'L' and the word 'ba', suggestion (ii) is followed. That means a new + state is considered and an edge labeled 'ba' is drawn from the – state.

Then the transition graph is as shown,



The language accepted by this graph is b* (a+b) a* + ba

In this case the TG accepts either the language b*(a+b) a* or 'ba', every time 'ba' need not be accepted.

17. Let L be any language. Let us define the transpose of *L* to be the language of exactly those words that are the words in L spelled backward. For example, if

L = {a abb bbaab bbbaa}

then

transpose (L) = **{a** *bba baabb aabbb}*

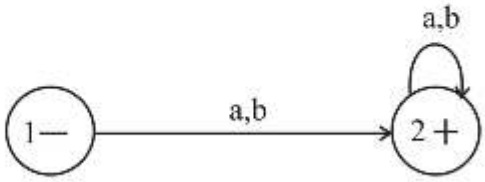
(i) Prove that if there is a FA that accepts L, then there is a TG that accepts the transpose of *L*.

(ii) Prove that if there is a TG that accepts L, then there is a TG that accepts the transpose of *L*.

Note: It is true, but much harder to prove that if an FA accepts L then some FA accepts the transpose of L. However, after Chapter 7 this will be trivial to prove.

Step 1 of 5

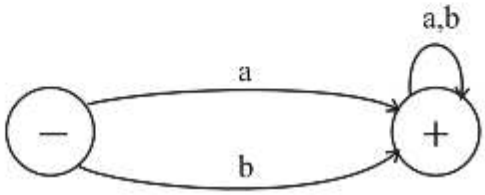
I) Let us consider L = {a abb bbaab bbbaa}
Then transpose (L) = {a bba baabb aabbb}
Finite Automata (FA):



Provide feedback (0)

Step 2 of 5

Then FA accept all words of language L
i.e., a accepted by the FA
bba accepted by the FA and soon
Transition graph (T_G):



Step 3 of 5

The T_G accepts the all words of transpose of L

i.e., the a accepted by the T_G

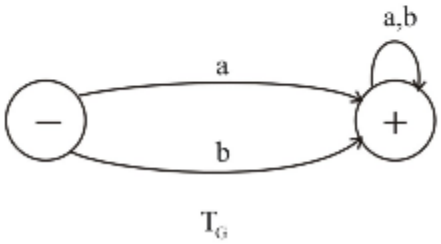
the bba accepted by the T_G and soon

$\therefore$ The F_G accepts the L of words, then the T_G accepts the transpose of L

Provide feedback (0)

Step 4 of 5

II)



T_G

Step 5 of 5

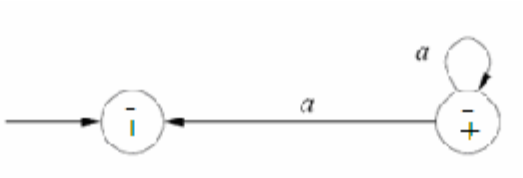
The T_G accepts language L of any words and accepts the language of transpose of L of any words.
L={a abb bbaab bbbaa}
The transition graph accepts all strigs of this language L.
Transpose (L)={a bba baabb aabbb}
The above transition graph accepts all string in transpose(L)
The transition graph allows Language L and transpose(L).

18. Transition graph T accepts language L. Show that if L has a word of odd length, then T has an edge with a label with an odd number of letters.

19. A student walks into a classroom and sees on the blackboard a diagram of a TG with two states that accepts only the word A. The student reverses the direction of exactly one, edge leaving all other edges and all labels and all +'s and -'s the same. But now the new TG accepts the language a*. What was the original machine?

Step 1 of 2

Original automata



Provide feedback (4)

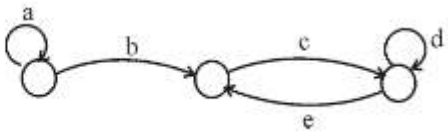
Step 2 of 2

After change of automata

20. Let us now consider an algorithm for determining whether a specific TG that has no A-edges accepts a given word.
Step **I** Number each edge in the TG in any order with the integers 1, 2, 3 . . . x, where x is the number of edges in the TG.
TRANSITION GRAPHS **99**
Step 2 Observe that if the word has y letters and is accepted at all by this machine, it can be accepted by tracing a path of not more than y edges.
Step 3 List all strings of y or fewer integers each of which -- x. This is a finite list.
Step 4 Check each string on the list in Step 3 by concatenating the labels of the edges involved to see if they make a path from a - to a + corresponding to the given word.
Step 5 If there is a string in Step 4 that works, the word is accepted. If none work, the word is not in the language of the machine.
(i) Prove this algorithm does the job.
(ii) Why is it necessary to assume that the TG has no A-edges?

Step 1 of 3

Suppose



Step 2 of 3

The above TG that has no $\wedge$ - edges accepts a given word.
Step 1: X is the number of edges in TG.
$X = 5$ regarding to the example.

Step 2: Y letters and is accepted at all by the above machine, it can be accepted by tracing path more than y edges *aabc*
Suppose $Y = 4$; y doesn't exceed more the 4 edges.
$Y = 4$; on this string tracing path is z.
Step 3: All strings of $y \leq x$
Step 4: check whether given string make a path or not (initial state to final state).
aabc $\rightarrow$ make a path
abce $\rightarrow$ in doesn't make a path.
Step 5: *aabc*, this string make a path from initial to final.
The word accepted
It not rejected.

Provide feedback (0)

Step 3 of 3

(ii) If TG has $\wedge-$ edges the above algorithm didn't accept the steps 2 and z.
That's the reason TG has no $\wedge-$ edges.

CHAPTER **8**

PROBLEMS

1. In the example for NFA3 above, why was it necessary to stipulate that no edges come into the two different start states? What can be done to add two machines when this condition fails?

2. Suppose *NFA1* is a machine for the regular expression r, and NFA2 for the regular expression r 2. Further suppose that *NFA1* has only one + state and that this state has no edges leading out of it (even back to itself). Show how to make a simple NFA for the language rjr 2.

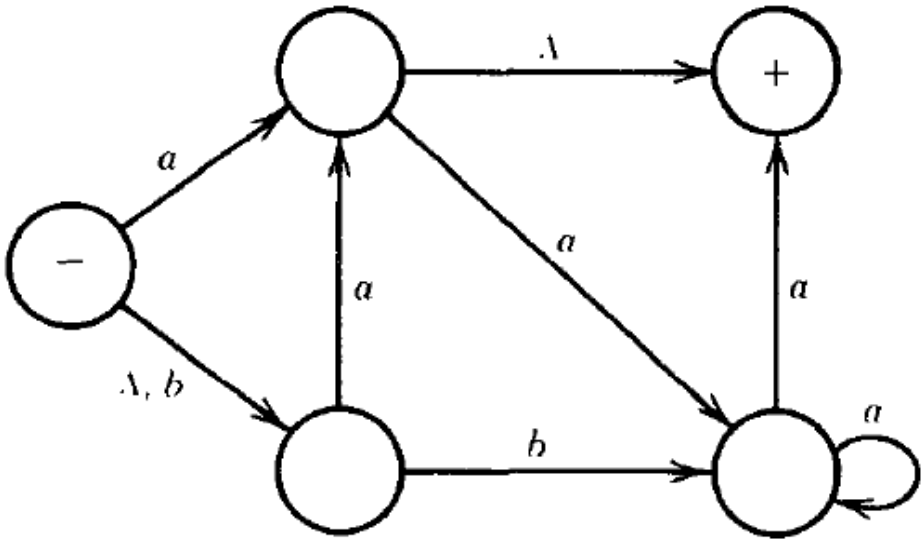
3. Can anything be done, similar to Problem 2, for (rl)*?

4. (i) How many NFA's are there with exactly two states where there can be 0, 1, or 2 final states and where the states may even be disconnected?

(ii) How many are there if the states cannot be disconnected?

(iii) How many different connected NFA's are there with two states that accept at least one word? (Be careful: We are counting machines not languages.)

Let us now introduce a machine called "a nondeterministic finite automaton with null string labels-," abbreviated NFA-A. This machine follows the same rules as an NFA except that we are allowed to have edges labeled with the null string A, as in the example below:



5. What words are accepted by the machine pictured above?

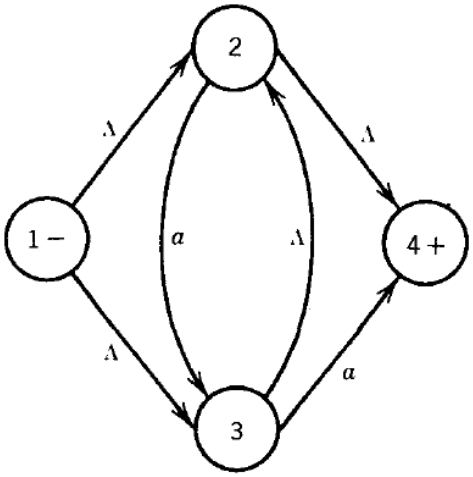
6. Prove that NFA-A's have the same power as FA's, that is, show that any language accepted by a machine of one type can also be accepted by a machine of the other.

7. Show that NFA-A's can be added, concatenated, and starred quite easily (without the extra conditions required for NFA's).

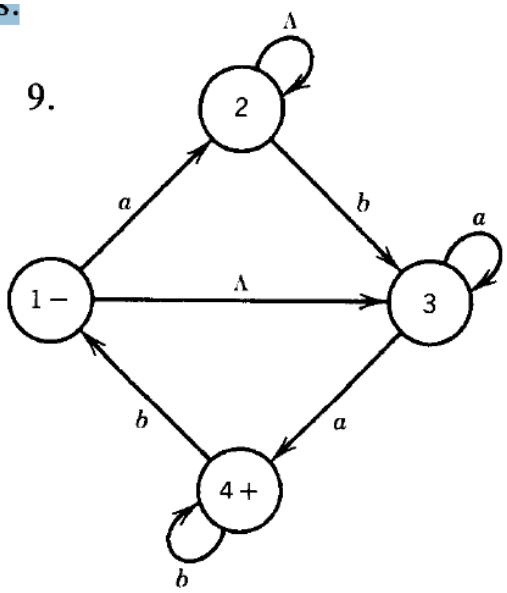
Convert the following NFA-A's into FA's.

Convert the following NFA-Λ's into FA's.

8.

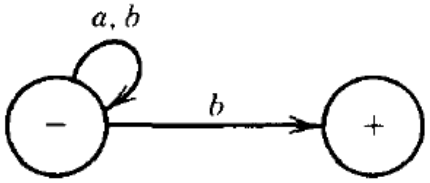


9.

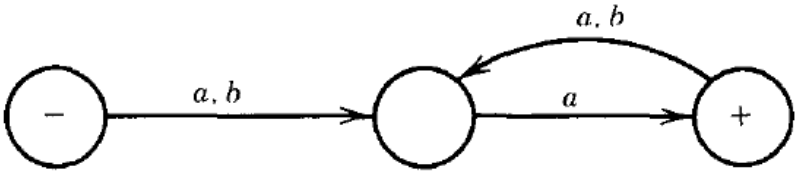


What is the language for each of the NFA's pictured below? Write regular expressions for each.

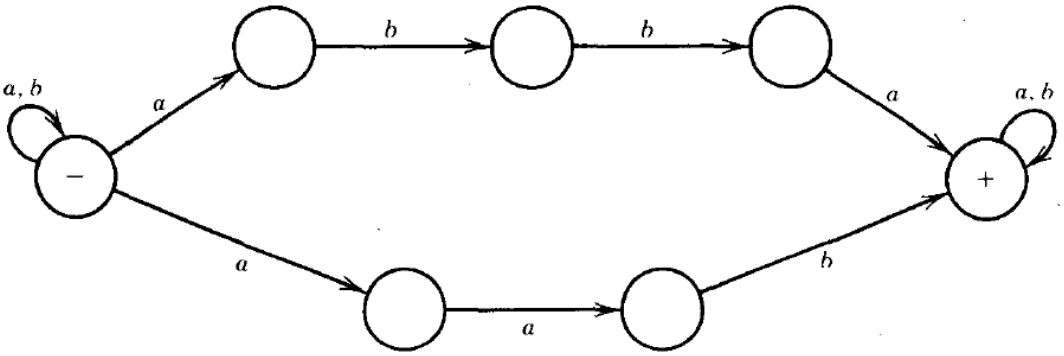
10.



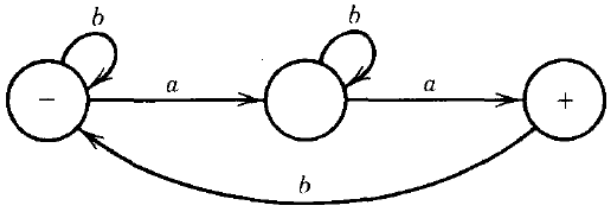
11.



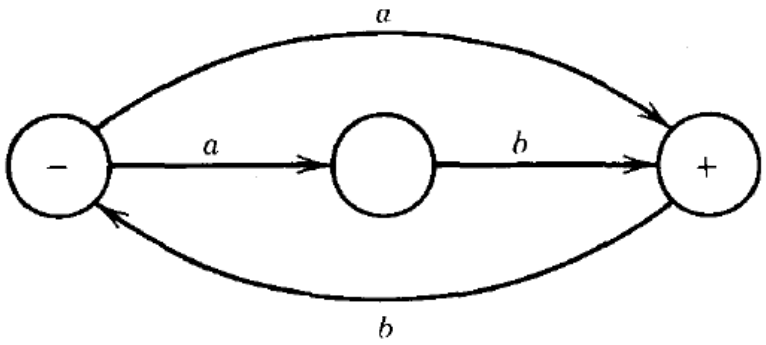
12.



13.

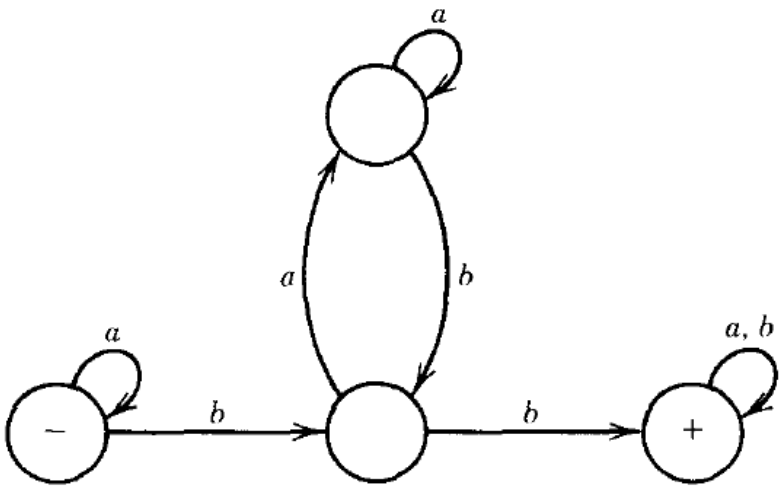


14.

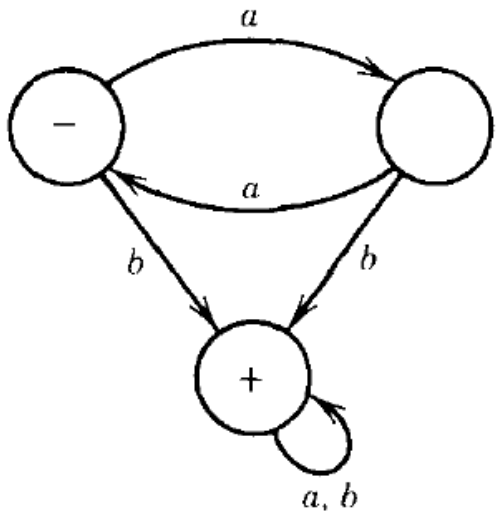


15. Build FA's for the languages in Problems 10 through 14.
For each of the following FA's, find NFA's that have fewer states and accept the same language.

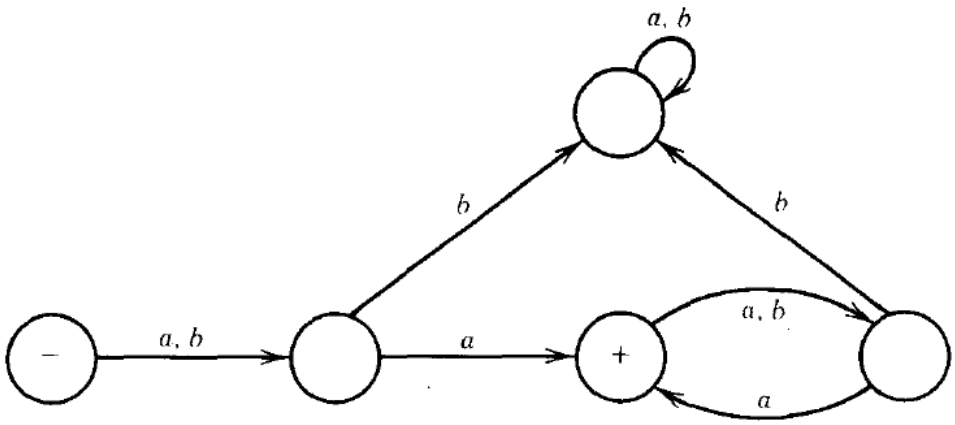
16.



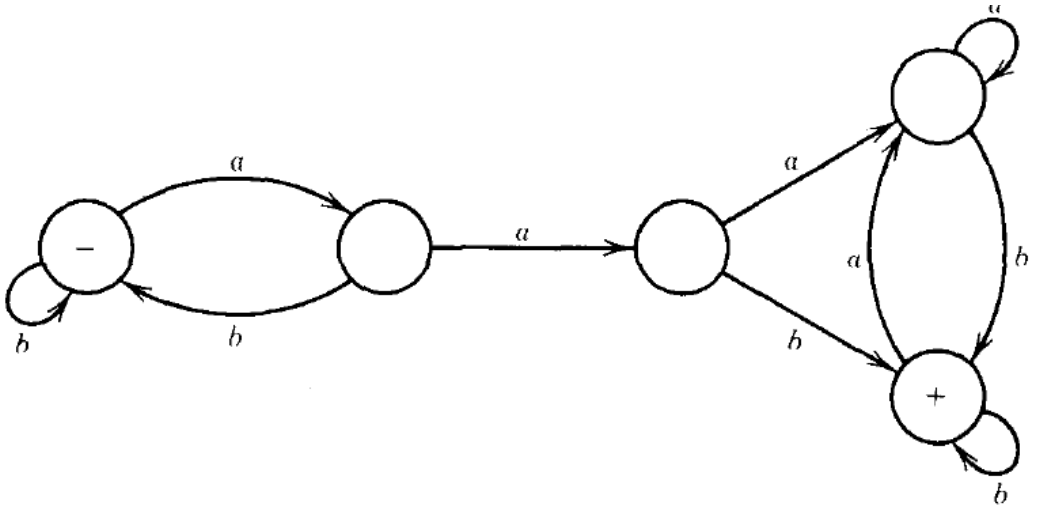
17.



18.



19. For the language accepted by the machine below, find a different FA with four states. Find an NFA that accepts the same language and has only seven edges (where edges with two labels are counted twice).



20. A one-person game can be converted into an NFA as follows. Let every possible board situation be a state. If any move (there may be several types of moves but we are not interested in distinguishing among them) can change some state x into some state y, then draw an edge from x to y and label it *m.* Label the initial position - and the winning positions +. "This game can be won in five moves" is the same as saying *"m5* is accepted by this NFA." Once we have the NFA we use the algorithm of Chapter 7 to convert it into a regular expression. The language it represents tells us how many moves are in each winning sequence.
Let us do this with the following example. The game of Flips is played with three coins. Initially they are all heads. A move consists of flipping two coins simultaneously from whatever they were to the opposite side. For example, flipping the end coins changes THH into HHT. We win when all three coins are tails. There are eight possible states: HHH, HHT **....** TTT. The only - is HHH; the only + is TTT. Draw this NFA, labeling any edge that can flip between states with the letter m. Convert this NFA into a regular expression. Is *m3* or *m'* in the language of this machine? The shortest word in this language is the shortest solution of this puzzle. What is it?

CHAPTER **9**

PROBLEMS

Each of the following is a Moore machine with input alphabet Y = *{a,b}* and output alphabet F = {0,1}. In Problems 1 through 5, draw the machines given the transition and output tables. In Problems 6 through 10, construct the transition

and output tables given the pictorial representations of the machines.

1.

	a	b	Output
q_0	q_1	q_2	1
q_1	q_1	q_1	0
q_2	q_1	q_0	1

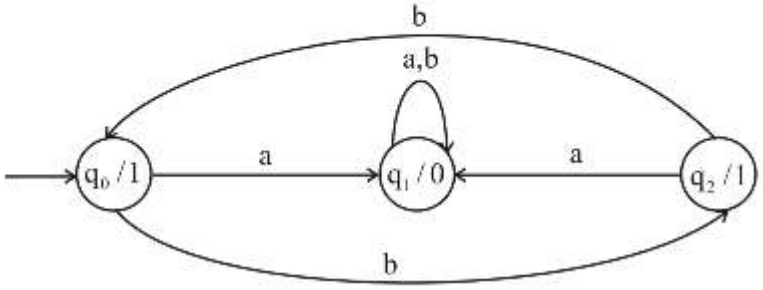
Step 1 of 10

I) Transition table and output table:

Old state	Output by the old state	New state	
		After input a	After input b
$-q_0$	1	q_1	q_2
q_1	0	q_1	q_1
q_2	1	q_1	q_0

Step 2 of 10

More machine:



2.

	a	*b*	Output
q_0	q_0	q_2	0
q_1	q_1	q_0	1
q_2	q_2	q_1	1

Step 3 of 10

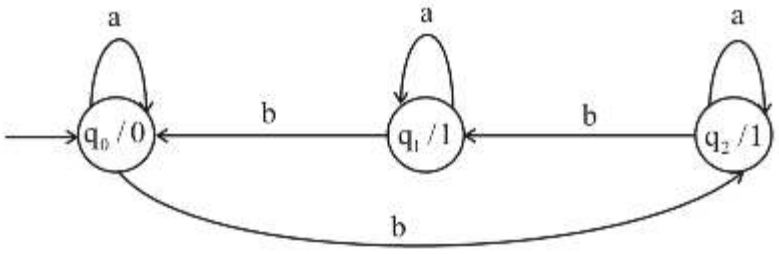
II) Transition table and output table:

Old state	Output by the old state	New state	
		After input a	After input b
q_0	0	q_0	q_2
q_1	1	q_1	q_0
q_2	1	q_2	q_1

Provide feedback (0)

Step 4 of 10

More machine:



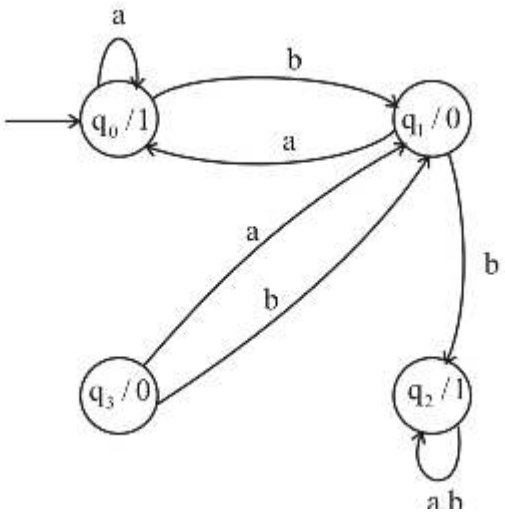
3.

	a	*b*	Output
q_0	q_0	q_1	1
q_1	q_0	q_2	0
q_2	q_2	q_2	1
q_3	q_1	q_1	0

Old state	Output by the old state	New state	
		After input a	After input b
q_0	1	q_0	q_1
q_1	0	q_0	q_2
q_2	1	q_2	q_2
q_3	0	q_1	q_1

ovide feedback (0)

ep 6 of 10



4.

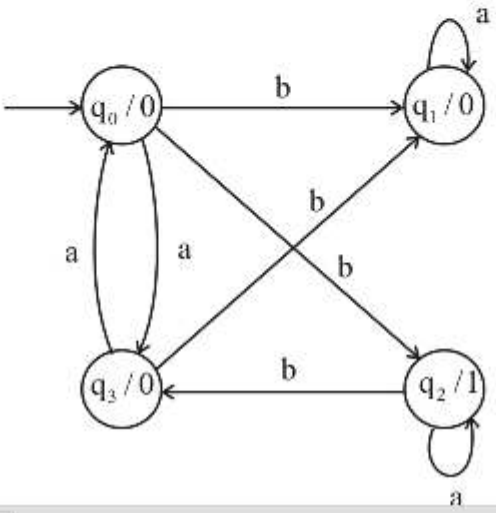
	a	*b*	Output
q_0	q_3	q_2	0
q_1	q_1	q_0	0
q_2	q_2	q_3	1
q_3	q_0	q_1	0

Old state	Output by the old state	New state	
		After input a	After input b
q_0	0	q_3	q_2
q_1	0	q_1	q_0
q_2	1	q_2	q_3
q_3	0	q_0	q_1

feedback (0)

of 10

More machine:



5.

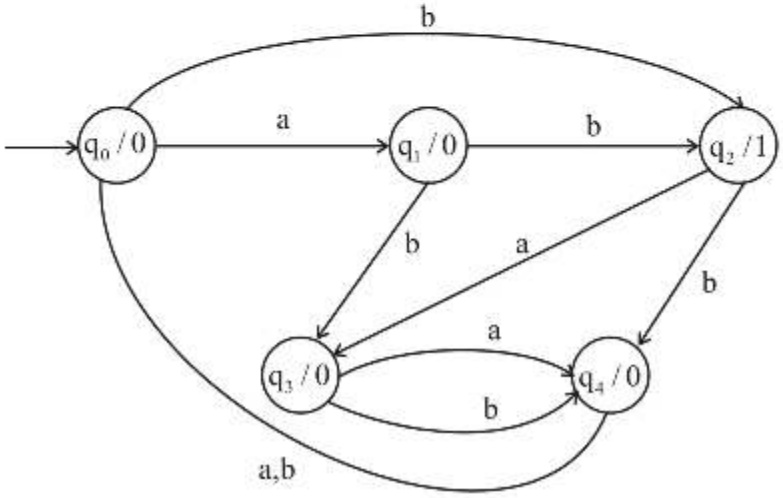
	a	*b*	Output
q_0	q_1	q_2	0
q_1	q_2	q_3	0
q_2	q_3	q_4	1
q_3	q_4	q_4	0
q_4	q_0	q_0	0

q_0	0	q_1	q_2
q_1	0	q_2	q_3
q_2	1	q_3	q_4
q_3	0	q_4	q_4
q_4	0	q_0	q_0

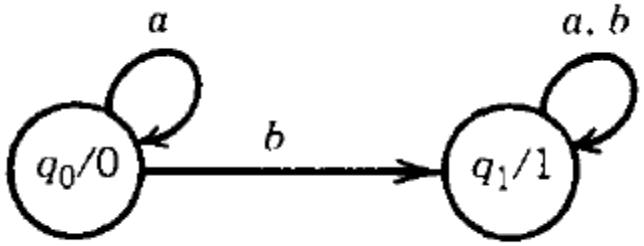
de feedback (0)

10 of 10

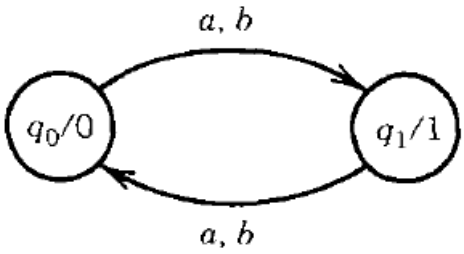
More machine:



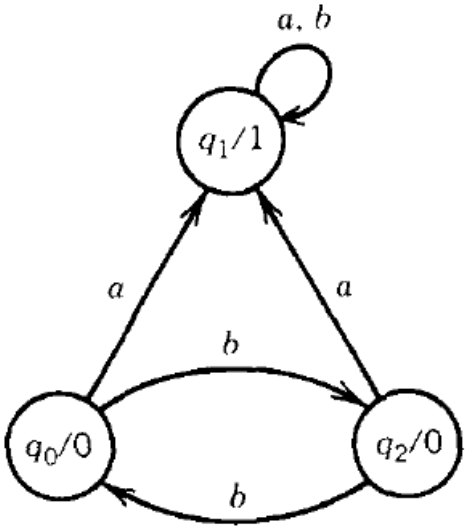
6.



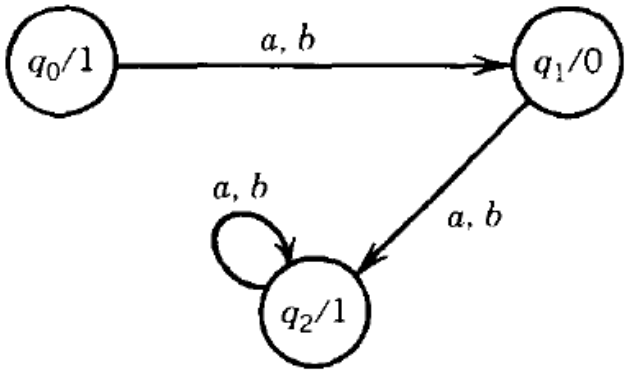
7.



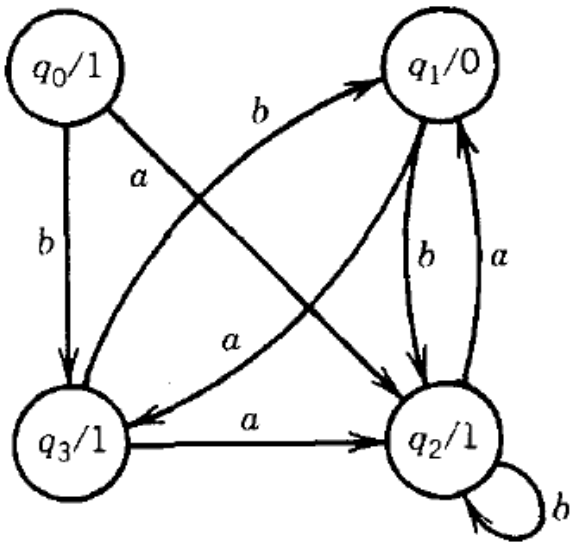
8.



9.



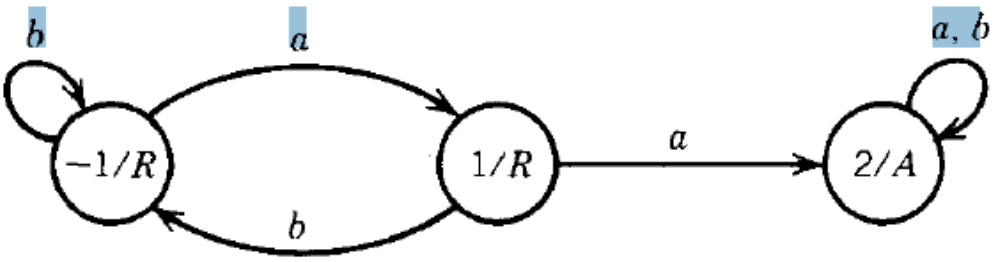
10.



I1. On each of the Moore machines in Problems 1 through 10, run the input sequence *aabab*. What are the respective outputs?

12. Even though a Moore machine does not define a language of the strings that it accepts as input, we can still use it to recogonize a language in the following sense. We can arrange that the last character printed out by the machine is an A for accept or an R for reject. For example, the following Moore machine will recognize the language of all words of

the form (a + b)*aa(a + b)* in the sense that these words and only these words will cause the last character printed by the machine to be A.



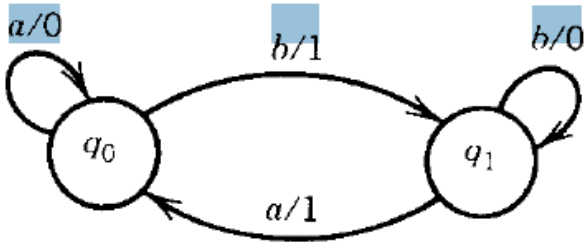
Show that all languages defined by regular expressions can be recognized by Moore machines in this fashion.

13. Suppose we define a Less machine to be a Moore machine that does not automatically print the character of the start state. The first character it prints is the character of the second state it enters. From then on, for every state it enters it prints a character, even when it reenters the start state. In this way the input string gets to have some say in what the first character printed is going to be. Show that these Less machines are equivalent to Mealy machines in the direct sense, that is, for every Less machine there is a Mealy machine that has the same output for every input string.

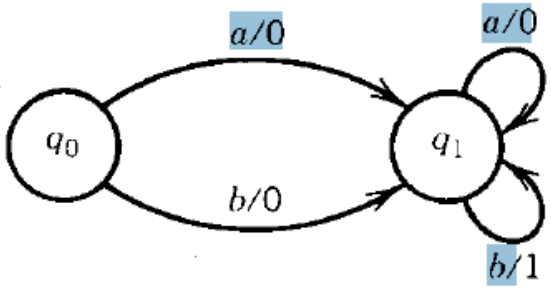
Mealy machines can also be defined by transition tables. The rows and the columns are both labeled with the names of the states. The entry in the table is the label of the edge (or edges) going from the row state to the column state (if there is no such edge, this entry is blank).

Construct the transition table for each of the four Mealy machines shown below.

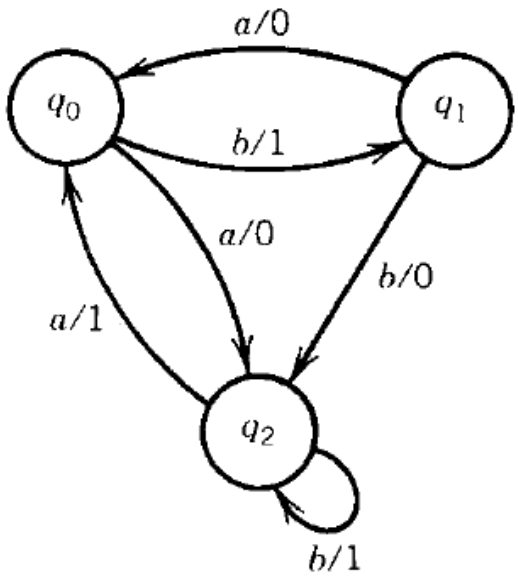
14.



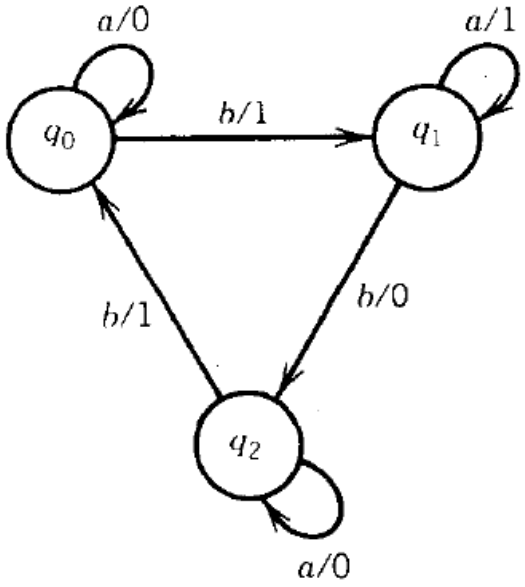
15.



16.



17.



Step 1 of 2

I) 16 MO's

Provide feedback (0)

Step 2 of 2

II) 2n

Step 1 of 5

I) Transition and output table of the first more machine in this problem:

Old state	Output by the old state	New state	
		After input a	After input b
$-q_0$	0	q_0	q_1
q_1	1	q_1	q_1

Provide feedback (0)

Step 2 of 5

II) Transition table and output table of second diagram:

Old state	Output by the old state	New state	
		After input a	After input b
$-q_0$	0	q_1	q_1
q_1	1	q_0	q_0

Step 3 of 5

III) Transition table and output table of third diagram:

Old state	Output by the old state	New state	
		After input a	After input b
$-q_0$	0	q_1	q_2
q_1	1	q_1	q_1
q_2	0	q_1	q_0

Provide feedback (0)

Step 4 of 5

IV)

Old state	Output by the old state	New state	
		After input a	After input b
$-q_0$	1	q_1	q_1
q_1	0	q_2	q_2
q_2	1	q_2	q_2

Step 5 of 5

V)

Old state	Output by the old state	New state	
		After input a	After input b
$-q_0$	1	q_2	q_3
q_1	0	q_3	q_2
q_2	1	q_1	q_2
q_3	1	q_2	q_1

Step 1 of 2

Input sequence = aabab

1.I) Output sequence = 100000
1.II) Output sequence = 000111
1.III) Output sequence = 111010
1.IV) Output sequence = 000110
1.V) Output sequence = 001001

Provide feedback (0)

Step 2 of 2

3.I) Output sequence = 000111
3.II) Output sequence = 010101
3.III) Output sequence = 011111
3.IV) Output sequence = 101111
3.V) Output sequence = 110101

Step 1 of 4

I) Transition table:

state	After input a	After input b
q_0	$q_0, 0$	$q_1, 1$
q_1	$q_0, 1$	$q_1, 0$

Provide feedback (1)

Step 2 of 4

II) Transition table:

State	After input a	After input b
q_0	$q_2, 0$	$q_1, 1$
q_1	$q_0, 0$	$q_2, 0$
q_2	$q_0, 1$	$q_2, 1$

Step 3 of 4

III) Transition table:

State	After input a	After input b
q_0	$q_1, 0$	$q_1, 0$
q_1	$q_1, 0$	$q_1, 1$

Provide feedback (0)

Step 4 of 4

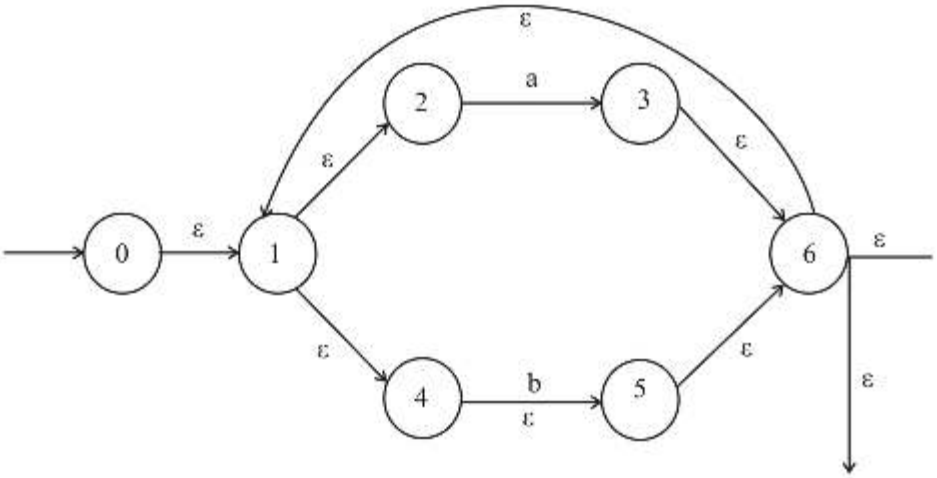
IV) Transition table:

State	After input a	After input b
q_0	$q_0, 0$	$q_1, 1$
q_1	$q_1, 1$	$q_2, 0$
q_2	$q_2, 0$	$q_0, 1$

18. tToh ep eerfxoarmmp le of the increment machine on page 160 used three states its job. Show that two states are all that are needed.

Step 1 of 1

$(a+b)^* aa(a+b)^*$

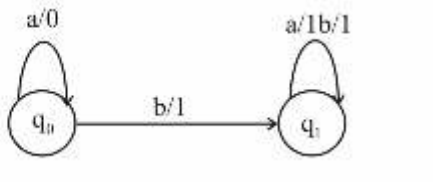


19. (i) Convert the Moore machines in Problems 1 through 10 into Mealy
(ii) Cmoanchvienrte s.the Mealy machines in Problems 14 through 17 into Moore machines.

Step 1 of 5

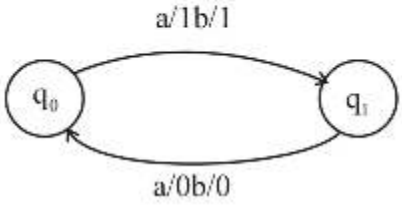
Convert more machine into mealy machine:

I)



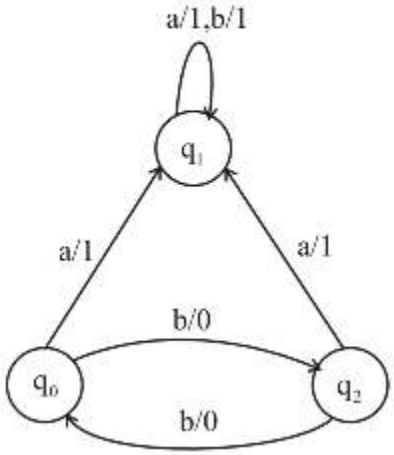
Step 2 of 5

II)



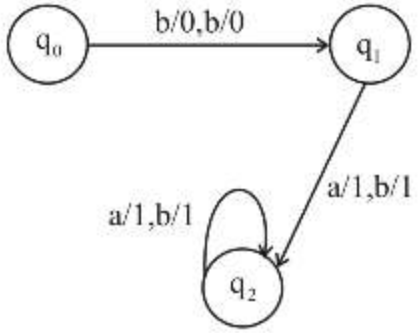
Step 3 of 5

III)



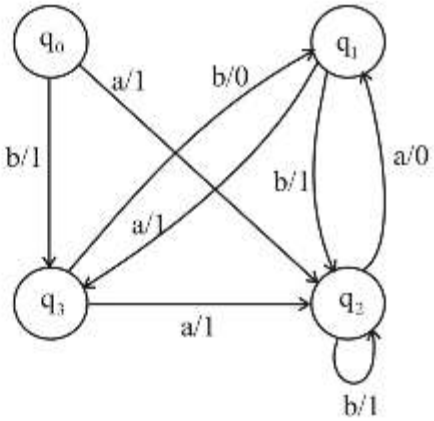
Step 4 of 5

IV)



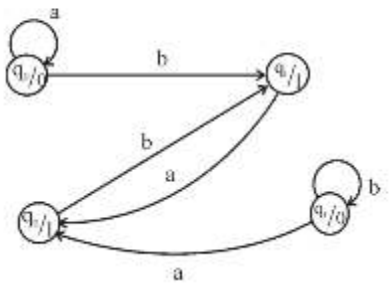
Step 5 of 5

V)



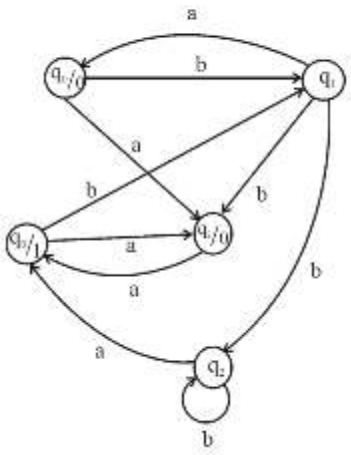
Convert mealy machine equivalent to Moore machine.

(i)



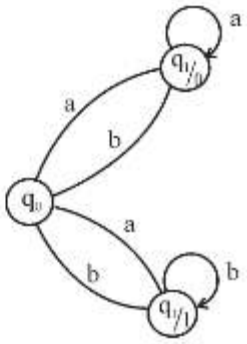
Step 2 of 4

(ii)



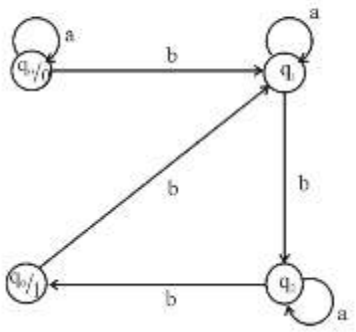
Step 3 of 4

(iii)

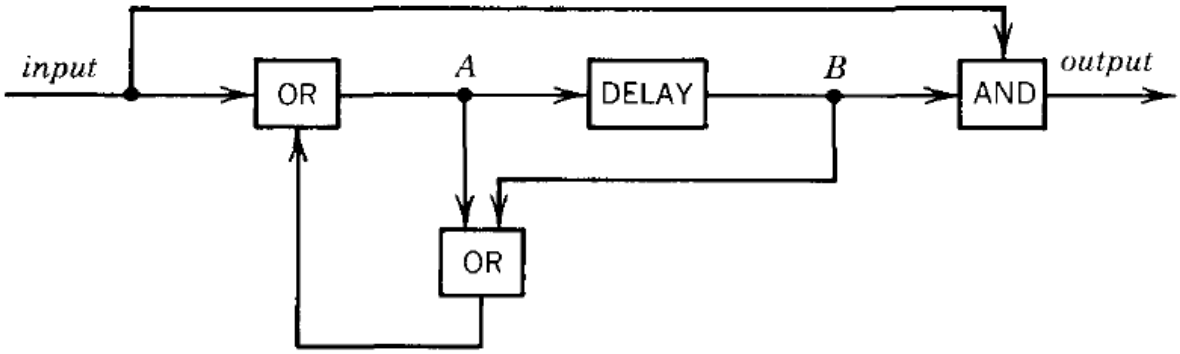


Step 4 of 4

(iv)



20. Draw a Mealy machine equivalent to the following sequential circuit.



Step 1 of 3

Mealy machine equivalent to the following sequential circuit.

Provide feedback (1)

Step 2 of 3

In put	A	B	O/P
0	0	0	00
1	1	1	00

Provide feedback (1)

Step 3 of 3