



Chapter 6 solutions

Automata Theory (University of the Punjab)

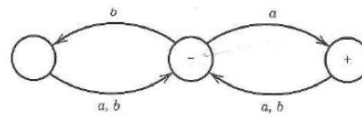


For each of the five FAs pictured in Problems 17, 19, and 20 in Chapter 5, build a transition graph that accepts the same language but has fewer states.

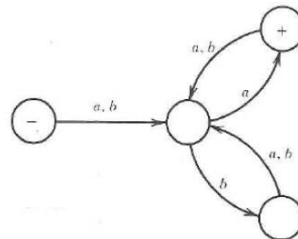
Problem 17

Describe in English the languages accepted by the following FAs:

(i)



(ii)

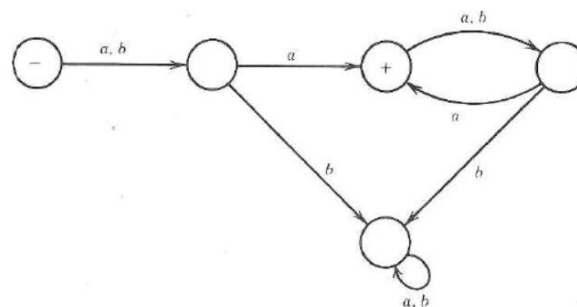


(iii)

Connecting...



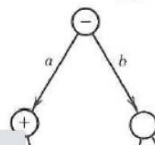
(iii)



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

Problems 19

Consider the following FA:



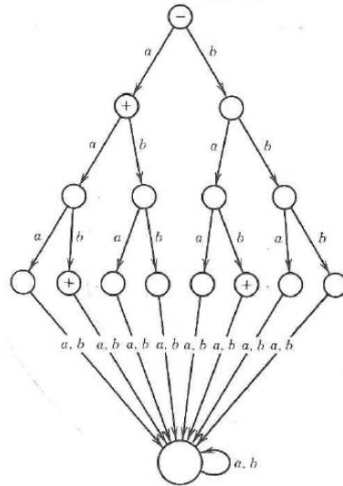
Connecting...



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

Problems 19

Consider the following FA:



Connecting...



Problems

(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 .

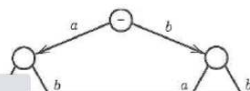
(iii) Show that by changing the location of $+$ signs alone, we can make this FA accept the language $\{bb\ aba\ bba\}$.

(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 extending the binary-tree part of this machine several more layers if necessary.

Problem 20

Let us consider the possibility of an infinite automaton that starts with this infinite binary tree:



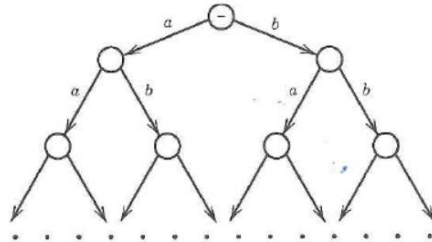
Connecting...



tree part of this machine several more layers if necessary.

Problem 20

Let us consider the possibility of an infinite automaton that starts with this infinite binary tree:



Let L be any infinite language of strings of a 's and b 's whatsoever. Show that by the judicious placement of $+$'s, we can turn the picture above into an infinite automaton to accept the language L . Show that for any given finite string, we can determine from this machine, in a finite time, whether it is a word in L . Discuss why this machine would not be a satisfactory language-definer for L .

Step-by-step solution

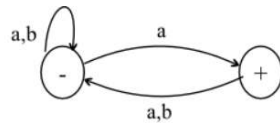
Step 1 of 5 ^

**Step-by-step solution**

Step 1 of 5 ^

1.

The given Finite Automata accepts the string containing a 's and b 's and the string ending with a . The transition graph with reduced number of states is as follows:



The initial state accepts the string with either a or b . A loop from initial state to final state for any number of a 's or b 's with string finally ending with a .

[Comment](#)

Step 2 of 5 ^

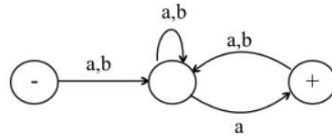
2.

[Comment](#)

Step 2 of 5 ^

2.

The given Finite Automata accepts the string starting with either a or b following any number of a 's or b 's and finally ending with a . The transition graph with reduced number of states is as follows:



The initial state accepts the string starting with either a or b which follows any number of a 's or b 's and finally ending with a .

[Comments \(1\)](#)

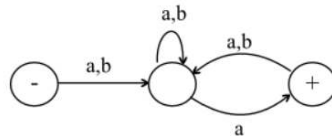
Step 3 of 5 ^

[Comments \(1\)](#)

Step 3 of 5 ^

3.

The given Finite Automata accepts the string starting with either a or b following any number of a 's or b 's and finally ending with a . The transition graph with reduced number of states is as follows:



The initial state accepts the string starting with either a or b which follows any number of a 's or b 's and finally ending with a .

[Comment](#)

Step 4 of 5 ^

4



Problem



For each of the next 10 words, decide which of the six machines on the next page accept the given word.



(i) Λ

(ii) a

(iii) b

(iv) aa

(v) ab

(vi) aba

(vii) $abba$



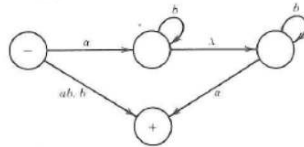
(vii) $abba$

(viii) bab

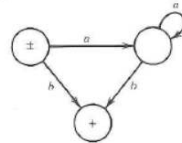
(ix) $baab$

(x) $abbb$

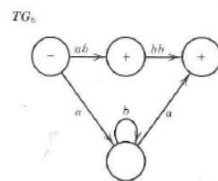
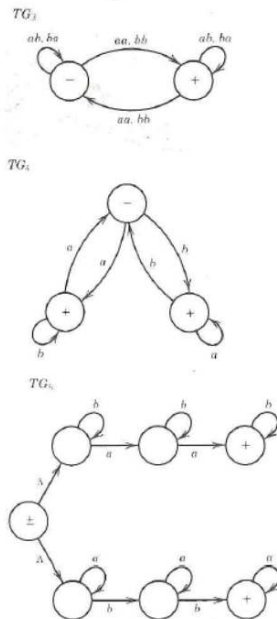
TG_1



TG_2



TG_3




Step-by-step solution


Step 1 of 10 ^

The machines that accept these words are


i) $\Lambda = TG2, TG5$ [Comment](#)

Step 2 of 10 ^


Chapter 6, Problem 2P

 2 Bookmarks

Show all steps: ☒ ON



Step 2 of 10 ^

ii) $a = TG4$

Comments (2)

Step 3 of 10 ^


iii) $b = TG1, TG2, TG4$


Comment

Step 4 of 10 ^


iv) $aa = TG1, TG3, TG5, TG6$

Comment


Chapter 6, Problem 2P

 2 Bookmarks

Show all steps: ☒ ON



Step 5 of 10 ^

v) $ab = TG1, TG2, TG4, TG6$

Comment

Step 6 of 10 ^

vi) $aba = TG1, TG5, TG6$

Comment

Step 7 of 10 ^

vii) $abba = TG1, TG5, TG6$

Comment

Chapter 6, Problem 2P

2 Bookmarks

Show all steps: ☒ ON

Step 8 of 10 ^

viii) bab = TG5

Comment

Step 9 of 10 ^

ix) baab = TG5

Comment

Step 10 of 10 ^

x) abbb = TG3, TG4, TG6

Comments (1)

Chapter 6, Problem 3P

Bookmark

Show all steps: ☒ ON

<

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

>

Step-by-step solution

Step 1 of 2 ^

Any language that can be accepted by a TG can be accepted by a TG with an even number of states. It can be shown with the help of following TG:

```

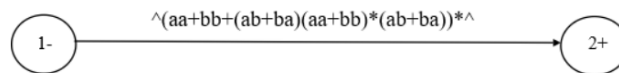
graph LR
    S((+  
-)) -- "aa, bb" --> S
    S -- "ab, ba" --> R(( ))
    R -- "aa, bb" --> R
    R -- "ab, ba" --> S

```

Step 2 of 2

Explanation:

- The given Transition Graph contains an even pair of letters at every edge. So, for a string to be accepted, the TG must have even number of states.
- In this TG, even number of a's and b's are accepted at the loops followed by the even pairs 'ab' and then again, the loop of 'aa' and 'bb'.
- Here, when a pair of unmatched letters (ab or ba) is read, it is known as unbalanced state, which means that an odd number of a's and b's are read.
- Since, there are two unmatched pairs, so the total number of a's and b's read from the input string become even again.
- This language is also known as **EVEN-EVEN**.
- This TG can be converted to another TG with single initial and final state as follows:



Comment

Was this solution helpful?  13  0

Problem



How many different TGs are there over the alphabet $\{a, b\}$ that have two states?



Step-by-step solution

Step 1 of 3 ^

The two different Transition graphs (TGs) over the alphabet $\{a, b\}$ that have two states are as follows:

- 1) Transitions graphs.
- 2) General Transition Graphs.

Comment

Step 2 of 3

Transition Graphs:

Chapter 6, Problem 4P

1 Bookmark

Show all steps: ☒ ON

Step 2 of 3 ^

Transition Graphs:

- A finite set of states, at least one of which is selected as the start state (-) and some (maybe none) of which are selected as final states (+).
- An alphabet Σ of possible input letters from which input strings are formed.
- A finite set of transitions (edges labels) that indicate how to move from one state to other state, based on scanning specified substrings of input letter (possibly even the null string ϵ).
- Every finite automaton can be represented as Transition Graph.

[Comment](#)

Step 3 of 3 ^

General Transition Graphs:

- A finite set of states over alphabet $\{a, b\}$, of which at least one is start state and some (may be none) are final states.
- An alphabet contains input letters.
- Directed edges linking some pairs of states, each labeled with a regular expression.
- Every non-finite automaton can be represented as Generalized Transition Graph.

Chapter 6, Problem 5P

3 Bookmarks

Show all steps: ☒ ON

<

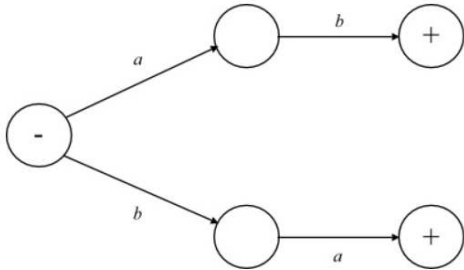
Prove that for every TG there is another TG that accepts the same language but has only one + state.

>

Step-by-step solution

Step 1 of 3 ^

Consider a transition graph TG over the alphabet $\{a, b\}$. The following is the transition graph TG:



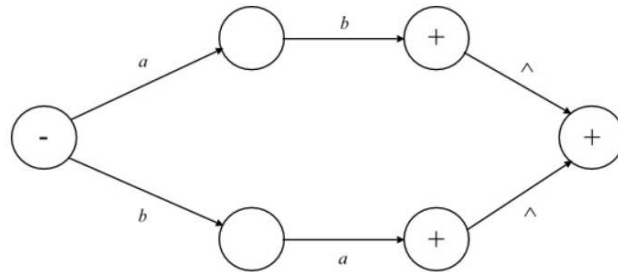
```

graph LR
    Start(( - )) -- a --> Middle(( ))
    Start -- b --> Bottom(( ))
    Middle -- b --> Final1(( + ))
    Bottom -- a --> Final2(( + ))

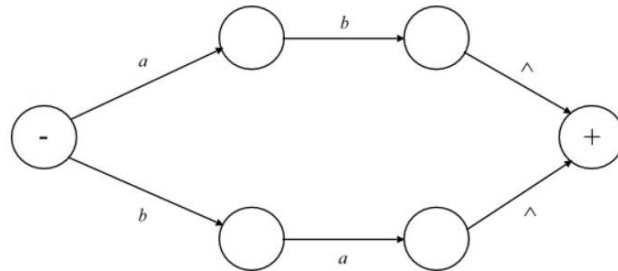
```



Create a new TG, say TG2, that has all the states and transitions of TG and a new accepting state. Connect each + state to this new accepting state via \wedge transitions.



Remove the +'s from each original accepting state.



[Comment](#)

Step 3 of 3 ^

If a string is accepted by the TG then it will leave the 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 the TG then it will leave the 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.

Therefore, it is proven that for every TG there is another TG that accepts the same language but has only one + state.

[Comment](#)

Was this solution helpful?

11

0



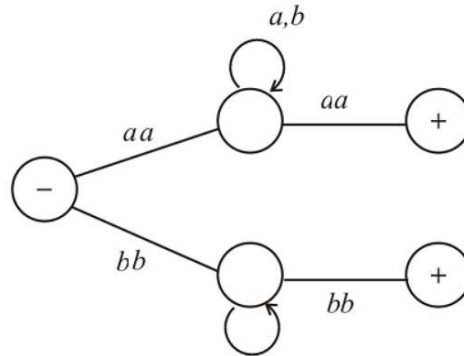
Build a TG that accepts the language L_1 of all words that begin and end with the same double letter, either of the form $aa \dots aa$ or $bb \dots bb$.



Step-by-step solution

Step 1 of 1 ^

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



Problem



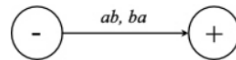
If OURSPONSOR is a language that is accepted by a TG called Henry, prove that there is a TG that accepts the language of all strings of a 's and b 's that end in a word from OURSPONSOR.



Step-by-step solution

Step 1 of 2 ^

Consider the language OURSPONSOR that is accepted by a Transition graph named Henry. Here, the OURSPONSOR contains any strings over $\{a, b\}$. Consider the strings accepted by the OURSPONSOR are $\{ab, ba\}$. The Transition Graph named Henry is as follows:



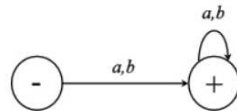
[Comment](#)

Step 2 of 2 ^



Step 2 of 2 ^

The Transition graph TG that accepts the language of all the strings over the alphabet $\{a, b\}$ is as follows:



The transition graph TG accepts the strings like $\{a, b, ab, ba, bab, baab, \dots\}$. The transition graph Henry accepts $\{ab, ba\}$. In order to prove the TG that accepts the language of all strings of a 's and b 's that end in a word from OURSPONSOR, take one string that is accepted by the TG and then append the word from OURSPONSOR at the end.

Take the string $baab$ from the language accepted by TG and then take the string ab from OURSPONSOR. Append the string ab to $baab$ at the end. Now the string becomes $baabab$. The string $baabab$ belongs to the language accepted by TG.

Therefore, Transition Graph TG that accepts the language of all strings of a 's and b 's that end in a word from OURSPONSOR.

[Comment](#)



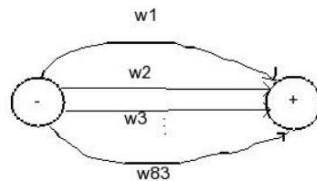
- (i) Suppose that L is a finite language whose words are $w_1, w_2, w_3, \dots, w_{83}$. Prove that there is a TG that accepts exactly the language L .
- (ii) Of all TGs that accept exactly the language L , what is the one with the fewest number of states?



Step-by-step solution

Step 1 of 1 ^

For a given list of words w_1, w_2, \dots, w_{83} we can construct a TG that accepts all these words:



Chapter 6, Problem 8P

1 Bookmark

Show all steps: ☒ ON

w3

w83

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

w1 + w2 + ... + w83

Comment

Was this solution helpful?

2

0

Chapter 6, Problem 9P

Bookmark

Show all steps: ☒ ON

Problem

<

Given a TG1 called TG₁ that accepts the language L₁, and a TG, called TG₂, that accepts the language L₂ show how to build a new TG (called TG₃) that accepts exactly the language L₁+L₂

>

Step-by-step solution

Step 1 of 1 ^

Let TG₁ and TG₂ are two transition graphs.

The L₁ and L₂ 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,

This document is available free of charge on

studocu

Downloaded by Adeel Cheema (adeel.cheema@gmail.com)

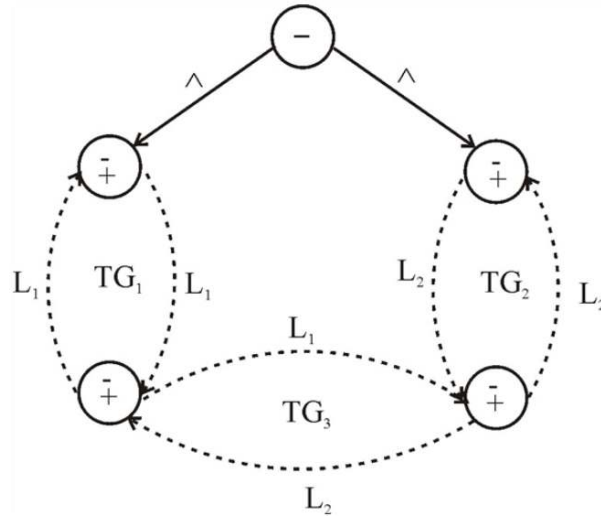


Let TG_1 and TG_2 are two transition graphs.

The L_1 and L_2 are two languages that are accepted by TG_1 and TG_2 .

Now, let us consider a transition graph TG_3 .

In order to accept language $L_1 + L_2$, TG_3 is constructed as follows,



Problem



Given TG_1 and TG_2 as described in Problem 9, show how to build TG_4 that accepts exactly the language $L_1 L_2$.

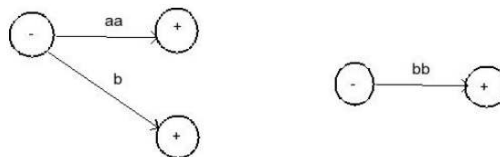
**Problem 9**

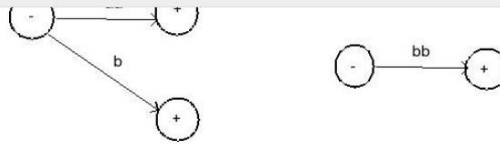
Given a TG_1 called TG_2 that accepts the language L_1 , and a TG , called TG_2 , that accepts the language L_2 show how to build a new TG (called TG_3) that accepts exactly the language $L_1 + L_2$

Step-by-step solution

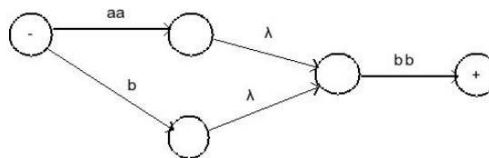
Step 1 of 1

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



[Comment](#)

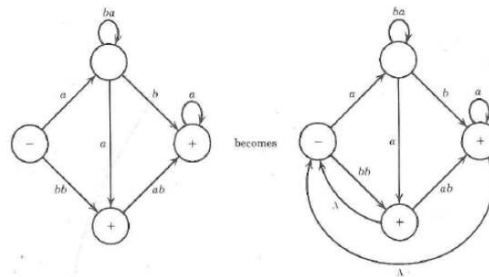
Was this solution helpful? 2 0



Problem



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 λ -edges? For example, by this method,



Step-by-step solution

Step 1 of 2

Consider a transition graph (TG) for any language L . If every + (final) state of the transition graph (TG) is connected back to the - (start) state, then the TG accepts the concatenated strings that are made of the set of strings that are accepted by the language L .



Step-by-step solution

Step 1 of 2 ^

Consider a transition graph (TG) for any language L . If every + (final) state of the transition graph (TG) is connected back to the - (start) state, then the TG accepts the concatenated strings that are made of the set of strings that are accepted by the language L .

[Comment](#)

Step 2 of 2 ^

For example, if the language accepts the strings $\{ab, ba\}$ then, the modified language accepts the strings such as $\{ab, ba, abba, baba, ababba, babaab, \dots\}$.

Therefore, 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.

[Comment](#)

Was this solution helpful? 9 0



Problem



- (i) Let the language L be accepted by the transition graph T and let L not contain the word Λ . Show how to build a new TG that accepts exactly all the words in L and the word Λ .
- (ii) Given TG, that accepts the language L show how to build a TG that accepts the language L^* .

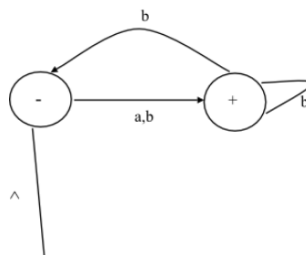


Step-by-step solution

Step 1 of 2 ^

(i)

A transition graph that accepts exactly all the words in L and does not contain the word Λ is as follows:



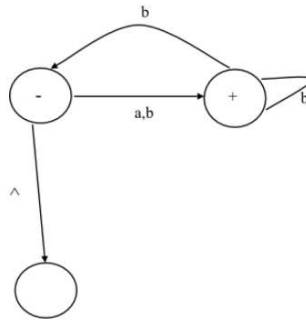


Step-by-step solution

Step 1 of 2 ^

(i)

A transition graph that accepts exactly all the words in L and does not contain the word \wedge is as follows:



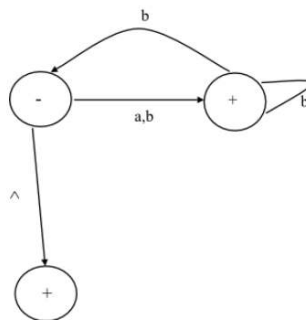
In the above transition graph, the symbol \wedge is redirecting to dead-end state so, \wedge is not accepted.

This transition graph that will accept all the words in the language L and the word \wedge is as follows:



In the above transition graph, the symbol \wedge is redirecting to dead-end state so, \wedge is not accepted.

This transition graph that will accept all the words in the language L and the word \wedge is as follows:



Here, the \wedge is redirecting to the final state, so it is accepted.

[Comment](#)

Step 2 of 2 ^

(ii)



Problem



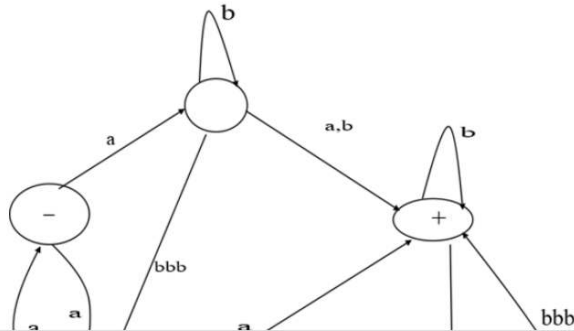
Verify that there are indeed three and only three ways for the TG on p. 84 to accept the word *abbabbbabba*.



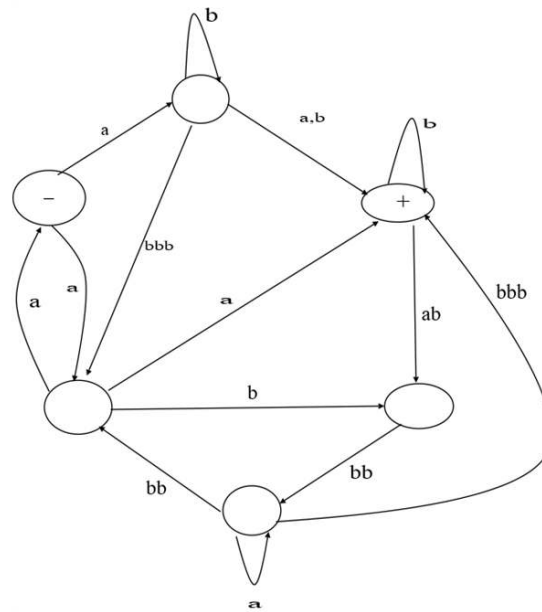
Step-by-step solution

Step 1 of 6

There are indeed 3 ways for the given TG to accept the word 'abbabbbabba'



There are indeed 3 ways for the given TG to accept the word 'abbabbbabba'

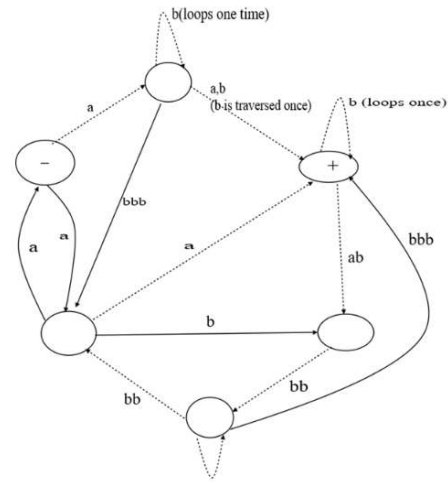


The three ways are as follows:

- First way:

Start reading the string from the start state which is denoted by '1'. Read the letter 'a' and then read 'b' once on the self-loop, once while going to the next state and again once on the loop which is on the final state. Then read the edge 'ab', then 'bb'. After that read the letter 'a' once on the self-loop and then 'bb' and finally read 'a' which ends the string and reaches the final state.

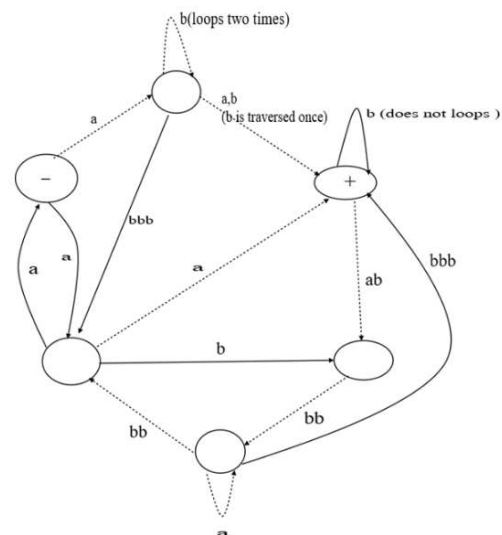
The above procedure can be illustrated through the following TG:



The path traversed is shown with dotted lines

- Second way:

Start reading the string from the start state which is denoted by '-'. Read the letter 'a', then the letter 'b' loops twice and then 'b' is read by the forthcoming edge but on the further transition which is a loop, 'b' is not read. After that 'ab' is read and then 'bb', then a loop is traversed reading the letter 'a' and then 'bb' is read. Finally, the letter 'a' is read which completes the string and reaches the final state.

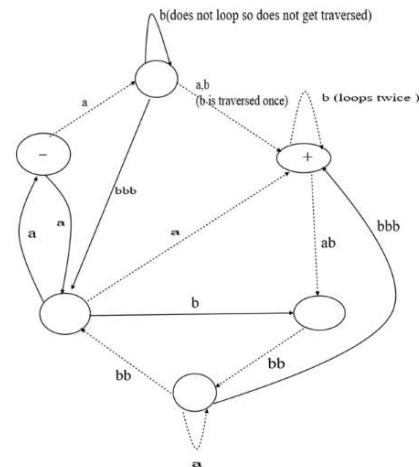


forthcoming edge but on the further transition which is a loop, 'b' is read twice. After that 'ab' is read and then 'bb', then a loop is traversed reading the letter 'a' and then

Comment

Step 6 of 6 ^

'bb' is read. Finally, the letter 'a' is read which completes the string and reaches the final state. Like in above cases, the dotted lines show the path which is traversed.



Problem

An FA with four states was sitting unguarded one night when vandals came and stole an edge labeled a . What resulted was a TG that accepted exactly the language b^* . In the morning the FA was repaired, but the next night vandals stole an edge labeled b and what resulted was a TG that accepted a^* . The FA was again repaired, but this time the vandals stole two edges, one labeled a and one labeled b , and the resultant TG accepted the language $a^* + b^*$. What was the original FA?

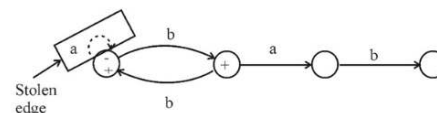
Step-by-step solution

Step 1 of 6

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

Comment

Step 2 of 6 ^



Comment

Step 3 of 6 ^

The above TG accepts only b^* .

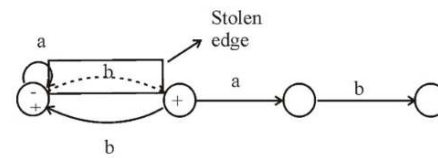
Step 3 of 6 ^

The above TG accepts only b^* .

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

[Comment](#)

Step 4 of 6 ^

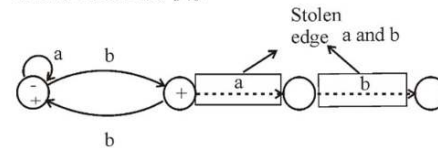


[Comment](#)

Step 5 of 6 ^

The above TG accepts only a^* . The next time vandals come and stole both edges a and b.

The resulted TG accepted exactly $a^* + b^*$.



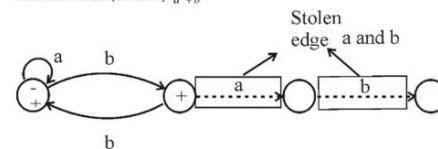
The above TG accepts only $a^* + b^*$.

[Comment](#)

Step 5 of 6 ^

The above TG accepts only a^* . The next time vandals come and stole both edges a and b.

The resulted TG accepted exactly $a^* + b^*$.

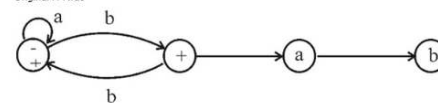


The above TG accepts only $a^* + b^*$.

[Comment](#)

Step 6 of 6 ^

Original FA was



[Comment](#)

Was this solution helpful?

Chapter 6, Problem 17P
4 Bookmarks
Show all steps: ☒ ON

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 T_G accepts the L of words, then the T_G accepts the transpose of L

[Comment](#)

Step 4 of 5

II)

T_G

[Comment](#)

Chapter 6, Problem 17P
4 Bookmarks
Show all steps: ☒ ON

T_G

[Comment](#)

Step 5 of 5

The T_G accepts language L of any words and accepts the language of transpose of L of any words.
 $L = \{a\ ab\ bbaab\ bbbba\}$
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).

[Comment](#)

Was this solution helpful?

7

3

Chapter 6, Problem 19P
Bookmark
Show all steps: ON

Problem

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

Step-by-step solution

Step 1 of 2

The original TG has the following characteristics:

- The TG has two states.
- The TG accepts only Λ .

The TG is as follows:

```

graph LR
    1((1-))
    2((2+))
    2 -- a --> 1
    2 -- a --> 2
  
```

Here, the transition is taking place from the final state (state 2) to the initial state (state 1), so it means that the TG makes transition without reading any symbol in the input.

[Comment](#)

Step 2 of 2

The new TG has the following characteristics:

- It has two states.

Chapter 6, Problem 19P
Bookmark
Show all steps: ON

The TG is as follows:

```

graph LR
    1((1-))
    2((2+))
    2 -- a --> 1
    2 -- a --> 2
  
```

Here, the transition is taking place from the final state (state 2) to the initial state (state 1), so it means that the TG makes transition without reading any symbol in the input.

[Comment](#)

Step 2 of 2

The new TG has the following characteristics:

- It has two states.
- It will accept the language a^* .
- The direction of edges is reversed, the start state (-) and the final state will remain the same.

```

graph LR
    1((1-))
    2((2+))
    1 -- a --> 2
    2 -- a --> 2
  
```

In this TG, the transition is taking place from the initial state (state 1) to the final state (state 2), so the symbol 'a' will be accepted by the TG and it will recur multiple times accepting a^* .

[Comment](#)

Was this solution helpful?

2 0

Chapter 6, Problem 20P
1 Bookmark
Show all steps: ☒ ON

Problem

Let us now consider an algorithm for determining whether a specific TG that has no \wedge -edges accepts a given word.

Step 1 Number each edge in the TG in any order with the integers $1, 2, 3, \dots, x$, where x is the number of edges in the TG.

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 $\leq x$. 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 whether 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 that this algorithm does the job.

(ii) Why is it necessary to assume that the TG has no \wedge -edges.

Step-by-step solution

Step 1 of 3 ^

Suppose

```

graph LR
    n1(( )) -- a --> n1
    n1 -- b --> n2(( ))
    n2 -- c --> n3(( ))
    n3 -- e --> n2
    n3 -- d --> n3
  
```

Chapter 6, Problem 20P
1 Bookmark
Show all steps: ☒ ON

Comment

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 $aahc$

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).

$aahc \rightarrow$ make a path

$abce \rightarrow$ in doesn't make a path.

Step 5: $aahc$, this string make a path from initial to final.

The word accepted

It not rejected.

Comment

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.

Comment