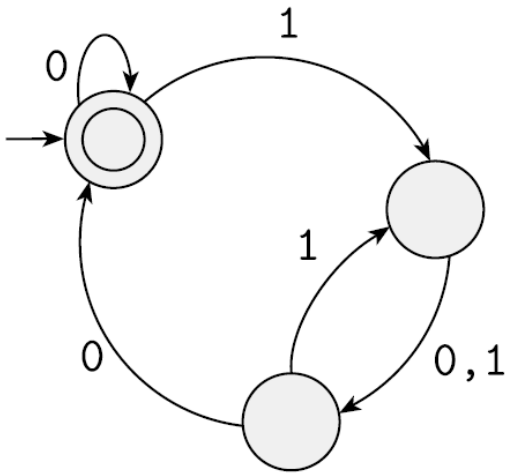
Problem

Answer all parts for the following DFA M and give reasons for your answers.



a. Is $\langle M, 0100 \rangle \in A_{\text{DFA}}$?

b. Is $\langle M, 011 \rangle \in A_{\text{DFA}}$?

c. Is $\langle M \rangle \in A_{\text{DFA}}$?

d. Is $\langle M, 0100 \rangle \in A_{\text{REX}}$?

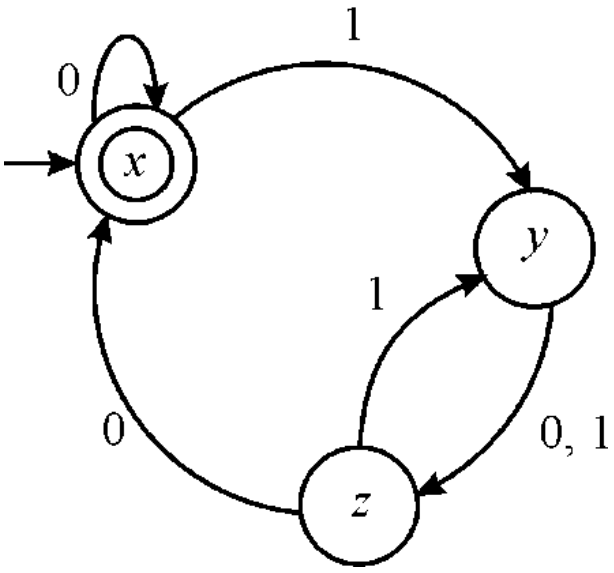
e. Is $\langle M \rangle \in E_{\text{DFA}}$?

f. Is $\langle M, M \rangle \in EQ_{\text{DFA}}$?

Step-by-step solution

Step 1 of 7

Consider the following figure:



As it is already given *M* is a DFA.

Theorem: A_{DFA} is a language which test whether given DFA *M* accepts provided string *w* or not. Use the following steps to check DFA accepted given string or not.

Consider a Turing machine *W* which decides A_{DFA}.

$W = \text{On input } \langle M, w \rangle, \text{ where } M \text{ is a DFA and } w \text{ is string}$

1. On *w* input simulate *M*.

2. When simulation ends at accept states then it is **accepted.** When simulation does not end at accept states then it is **rejected**.

Comment

Step 2 of 7

Use above theorem to check whether *M* accepts string 0100 or not. Simulate, DFA *M* on 0100. Consider the figure shown above for the production of *M*.

• As shown in above figure the starting and final (accepting) state of *M* is *x*. After taking the first input 0 from string 0100 the next state is still *x*.

• After taking the second input 1 from string 0100 the next state is *y*. similarly after taking fourth '0' input from 0100 the next state becomes *x* which is an acceptable state.

Hence, *M* accepts input string 0100.

Comment

Step 3 of 7

Use above theorem to check whether *M* accepts string 011 or not. Simulate, DFA *M* on 011. Consider the figure shown above for the production of *M*.

• As shown in above figure the starting and final (accepting) state of *M* is *x*. After taking the first input 0 from string 011 the next state is still *x*.

• After taking the second input 1 from string 011 the next state is *y*. similarly after taking third '0' input from 011 the next state becomes *z* which is not an acceptable state.

Hence, *M* does not accept input string 011.

Comments (1)

Step 4 of 7

Use above theorem to check whether *M* accepts given string or not. Here input string *w* is not given in A_{DFA}. Here A_{DFA} is not in proper format.

Hence, *M* does not accept input string.

Comment

Step 5 of 7

Consider a language A_{REX} which is used to determine whether particular regular expression can generate provided string or not. Consider a Turing machine *W* which decides whether regular expression can generate provided string or not.

$W = \text{On input } \langle R, w \rangle, \text{ where R is a Regular expression and } w \text{ is string}$

Here, in this part *M* is a DFA instead of regular expression but for A_{REX} it should be regular expression to check whether it can generate string 0100 or not.

Hence, *M* cannot generate string 0100.

Comment

Step 6 of 7

Use marking algorithm E_{DFA} to check, whether DFA accept any string or not. Use the following steps to check DFA accepted any string or not.

Consider a Turing machine *W* which decides E_{DFA}.

W = On input $\langle M \rangle$, where M is a DFA

1. Marks the initial state of *M* hat is *x*.
2. Repeat the state till new state is not marked.
3. Mark the next state of transition which comes from any other marked states.
4. If accept state is not marked then **accept**, else **reject**.

Consider the transition diagram shown above. In that diagram the initial and final state are same that is *x*. so when applying the first step of E_{DFA} theorem then initial as well as final states both get marked. According to step 4 of above theorem it is not acceptable when final states is marked.

Hence, *M* cannot accept any string. M is not an empty language.

Comments (5)

Step 7 of 7

Consider theorem EQ_{DFA} to determine whether two DFA identify the same language.

Consider a Turing machine *W* which decides EQ_{DFA}.

Here, in this part $L(M) = L(M)$ therefore derived language $L(C)$ becomes empty.

Therefore Turing machine W accepts the derived language $L(C)$.

Hence, yes the language of *M* accepts itself.

Comment