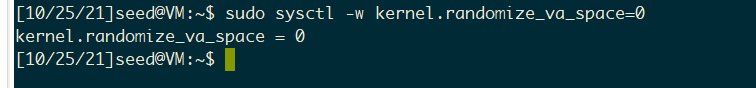
2.1 Turning Off Countermeasures

2.2.1 Address Space Randomization

$ sudo sysctl -w kernel.randomize\_va\_space=0



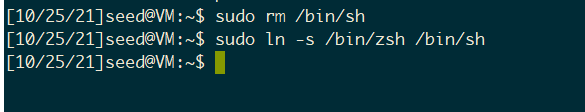
2.2.2 The StackGuard Protection Scheme

2.2.3 Non-Executable Stack

2.2.4 Configuring /bin/sh

$ sudo rm /bin/sh

$ sudo ln -s /bin/zsh /bin/sh



According to 2.2.2 and 2.2.3, we need to use

$ gcc -fno-stack-protector -z execstack example.c -o example

instead of

$ gcc example.c -o example

at our final step.

2.2 Task 1: Running Shellcode

**Answers**

1)

a) What is the language accepted by M?

b) Use the formal definition of Deterministic Finite Automaton to describe M.

=

=

=

=

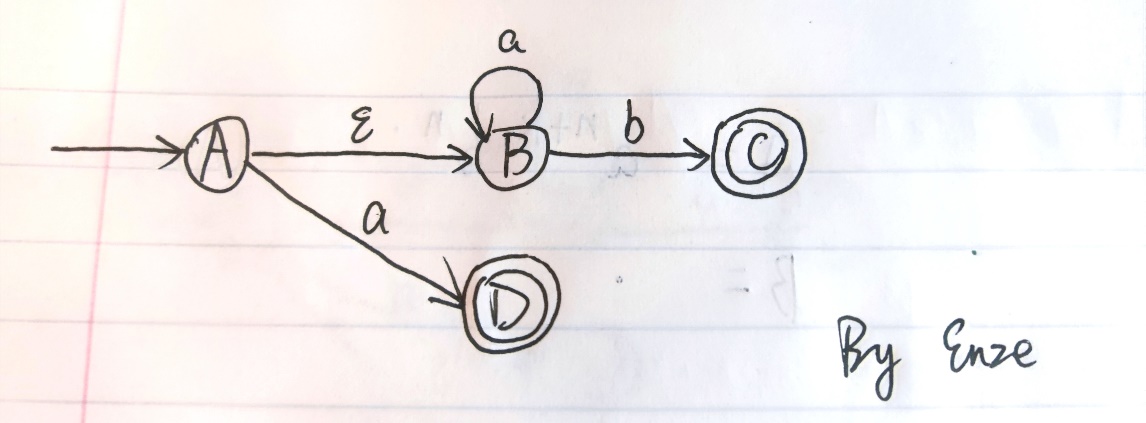
=

=

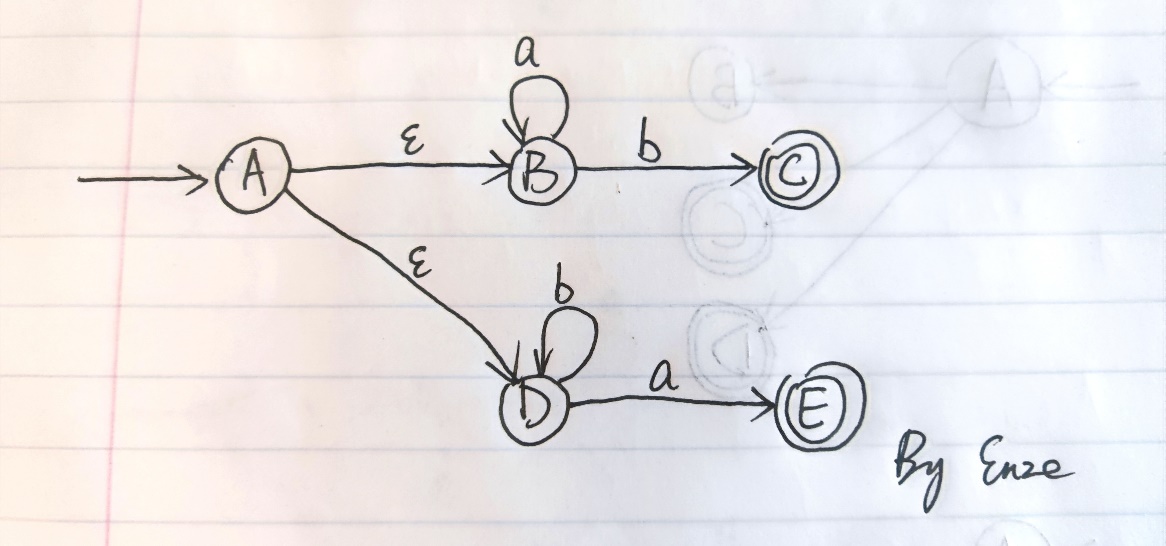
|  |  |  |
| --- | --- | --- |
| States \ Symbols |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

2)

a) L = {a\*b and a}

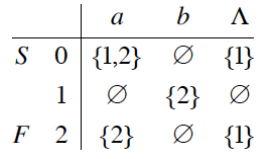


b) L = {a\*b and b\*a}



3)

a) Use the formal definition of NFA’s to describe the following NFA

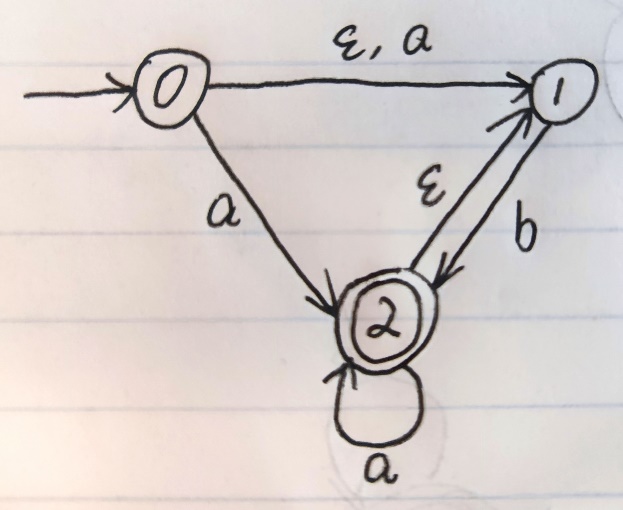


=

|  |  |  |  |
| --- | --- | --- | --- |
| States \ Symbols |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

b) Find a DFA that is equivalent to the given NFA

NFA:

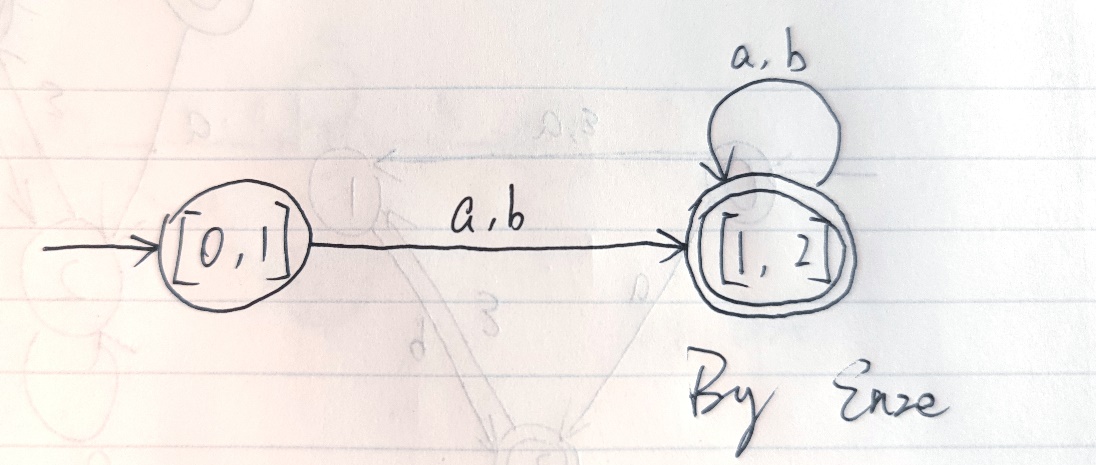


|  |  |  |
| --- | --- | --- |
| States \ Symbols |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

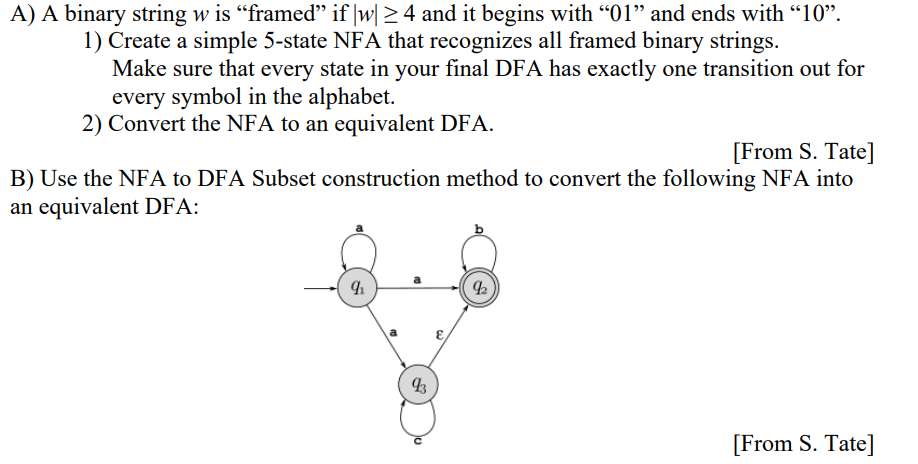
Transition table for the equivalent DFA:

|  |  |  |
| --- | --- | --- |
| States |  |  |
|  |  |  |
|  |  |  |

The equivalent DFA:



Problem Two



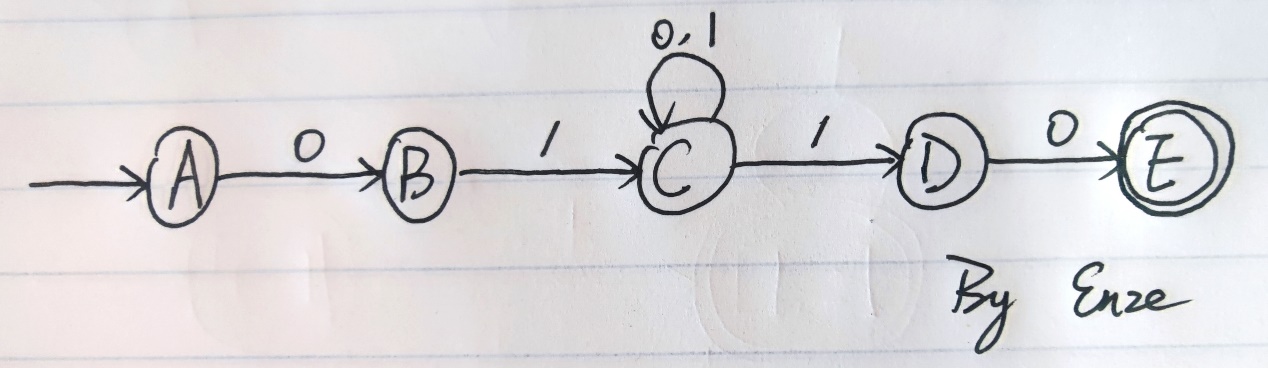
**Answers**

A) A binary string w is “framed” if |w| ≥ 4 and it begins with “01” and ends with “10”

1) Create a simple 5-state NFA that recognizes all framed binary strings

=

|  |  |  |  |
| --- | --- | --- | --- |
| States \ Symbols |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |



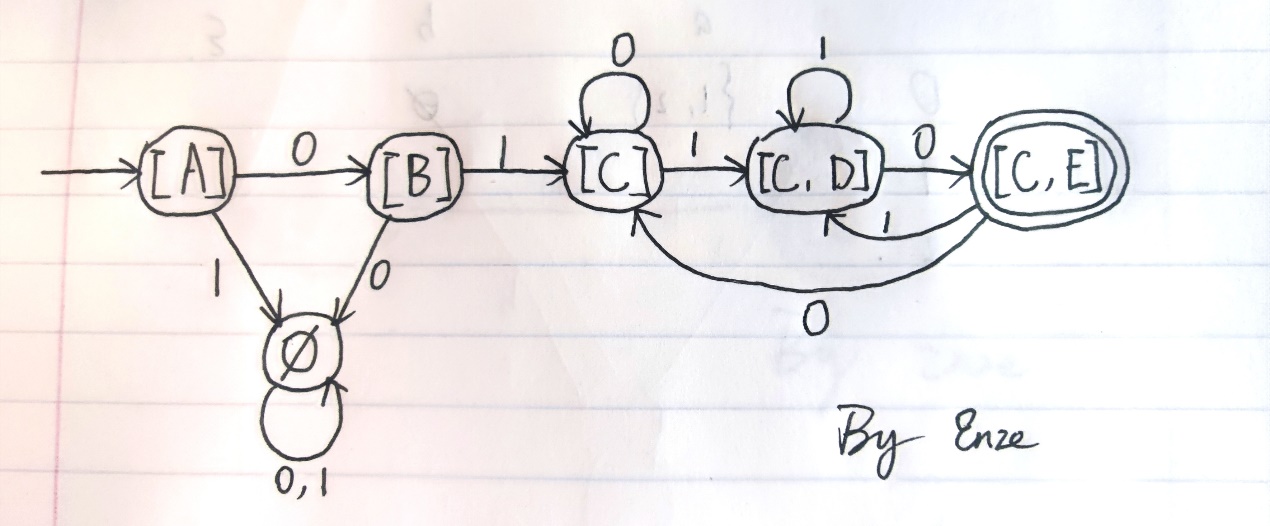
2) Convert the NFA to an equivalent DFA

|  |  |  |
| --- | --- | --- |
| States \ Symbols |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

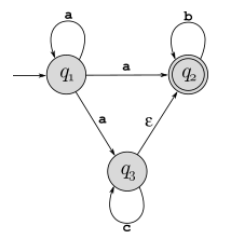
Transition table for the equivalent DFA:

|  |  |  |
| --- | --- | --- |
| States |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

The equivalent DFA:



B) Use the NFA to DFA Subset construction method to convert the following NFA into an equivalent DFA:

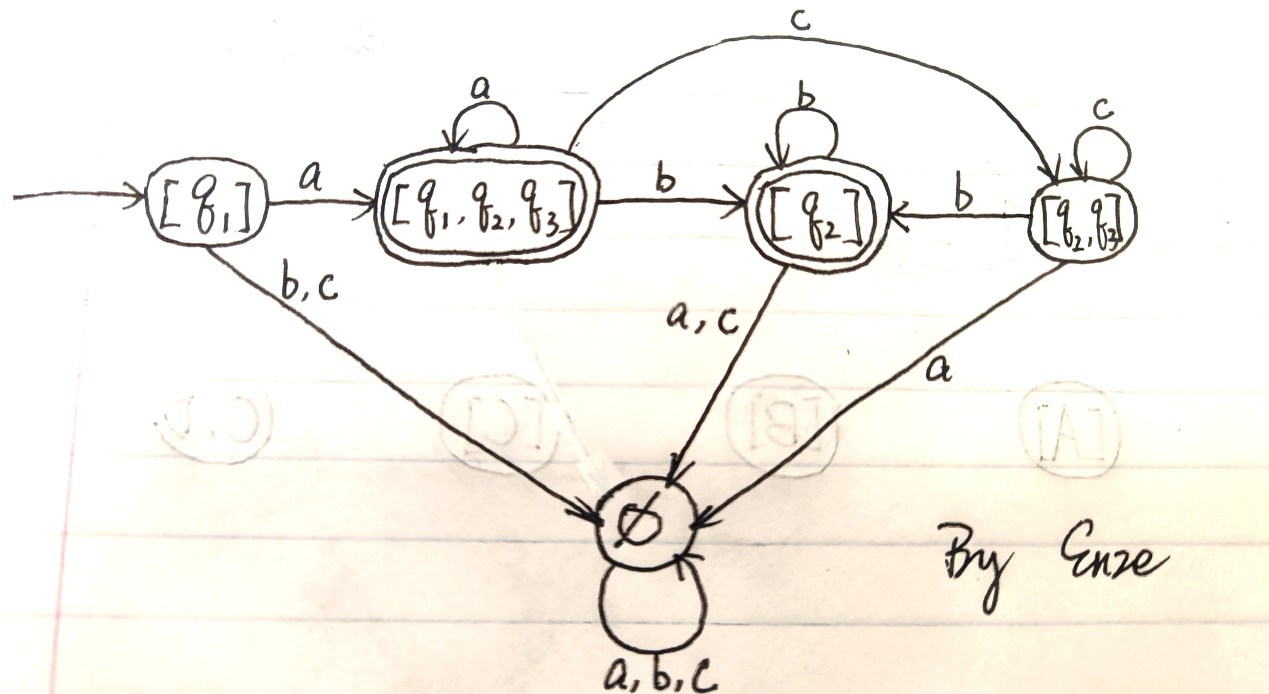


|  |  |  |  |
| --- | --- | --- | --- |
| States \ Symbols |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

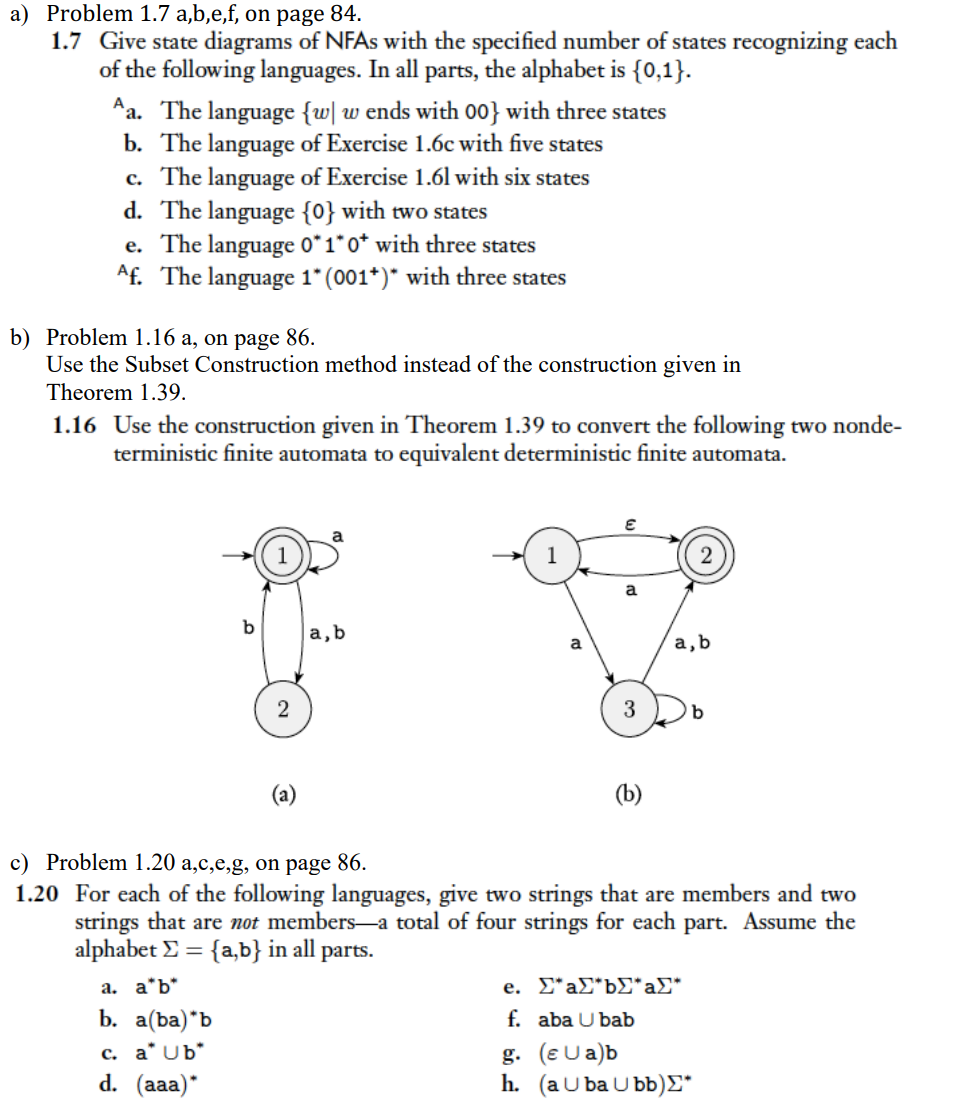
Transition table for the equivalent DFA:

|  |  |  |  |
| --- | --- | --- | --- |
| States |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

The equivalent DFA:

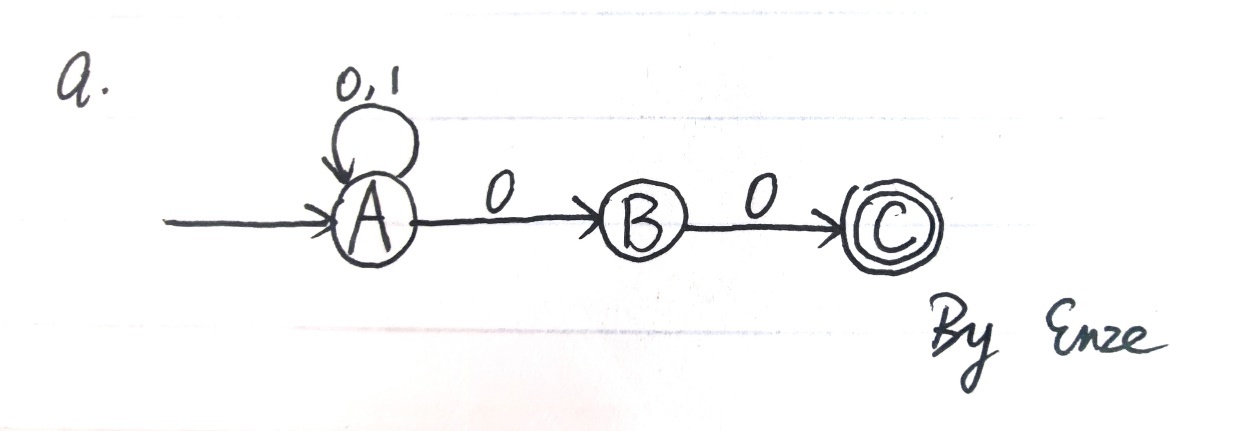


Problem Three



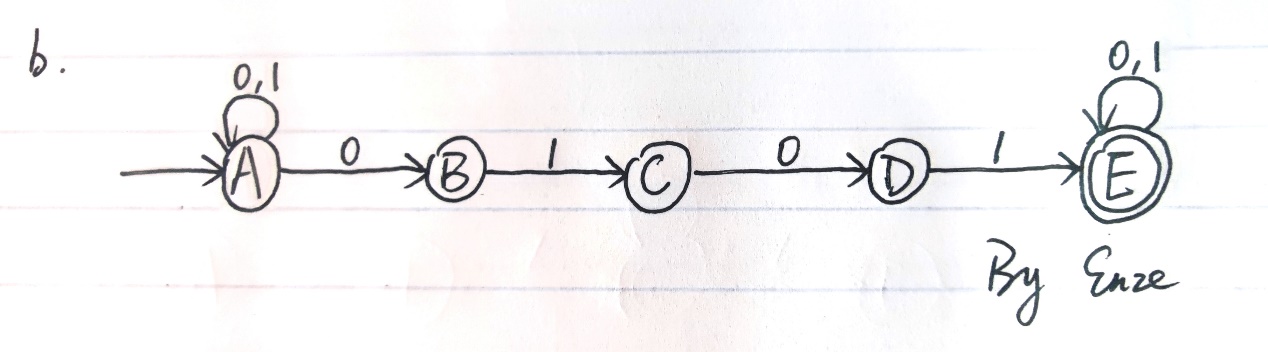
a) Problem 1.7 a,b,e,f, on page 84



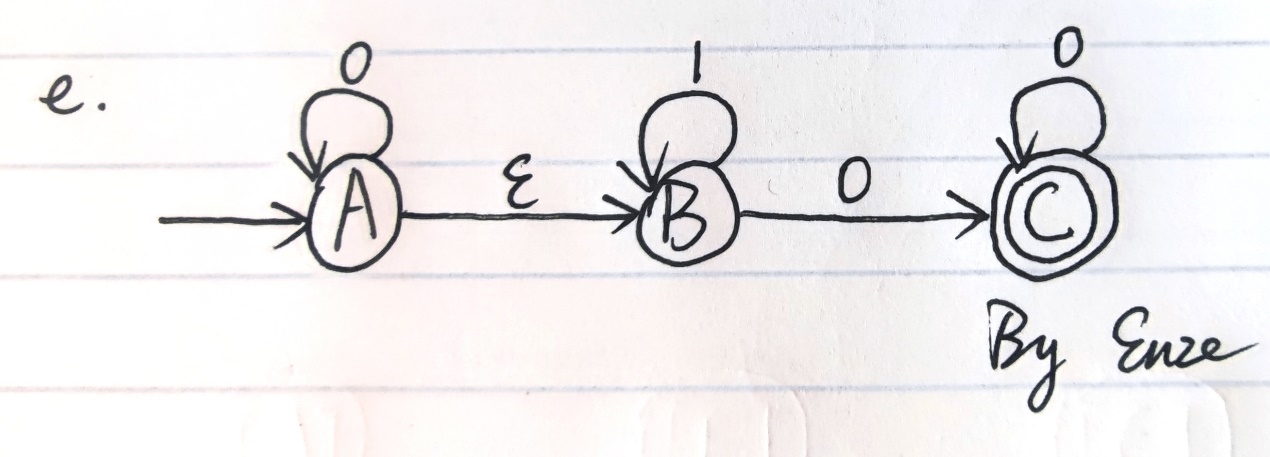




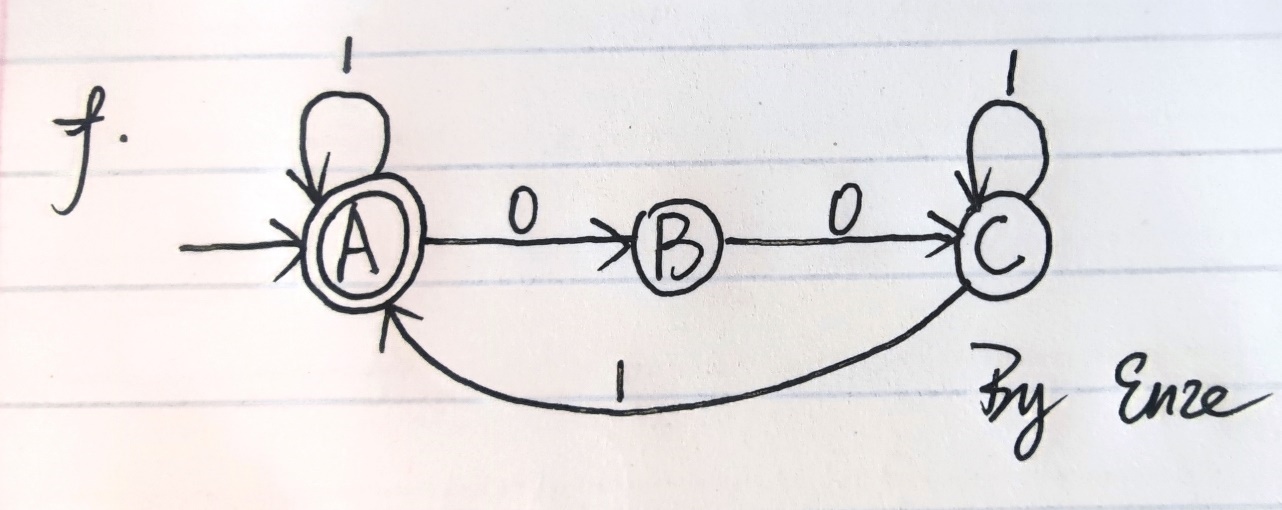
(1.6c: {w| w contains the substring 0101 (i.e., w = x0101y for some x and y)}







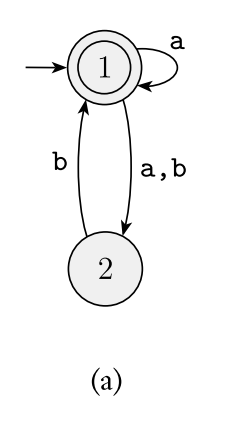




b) Problem 1.16 a, on page 86.

Use the Subset Construction method instead of the construction given in

Theorem 1.39.

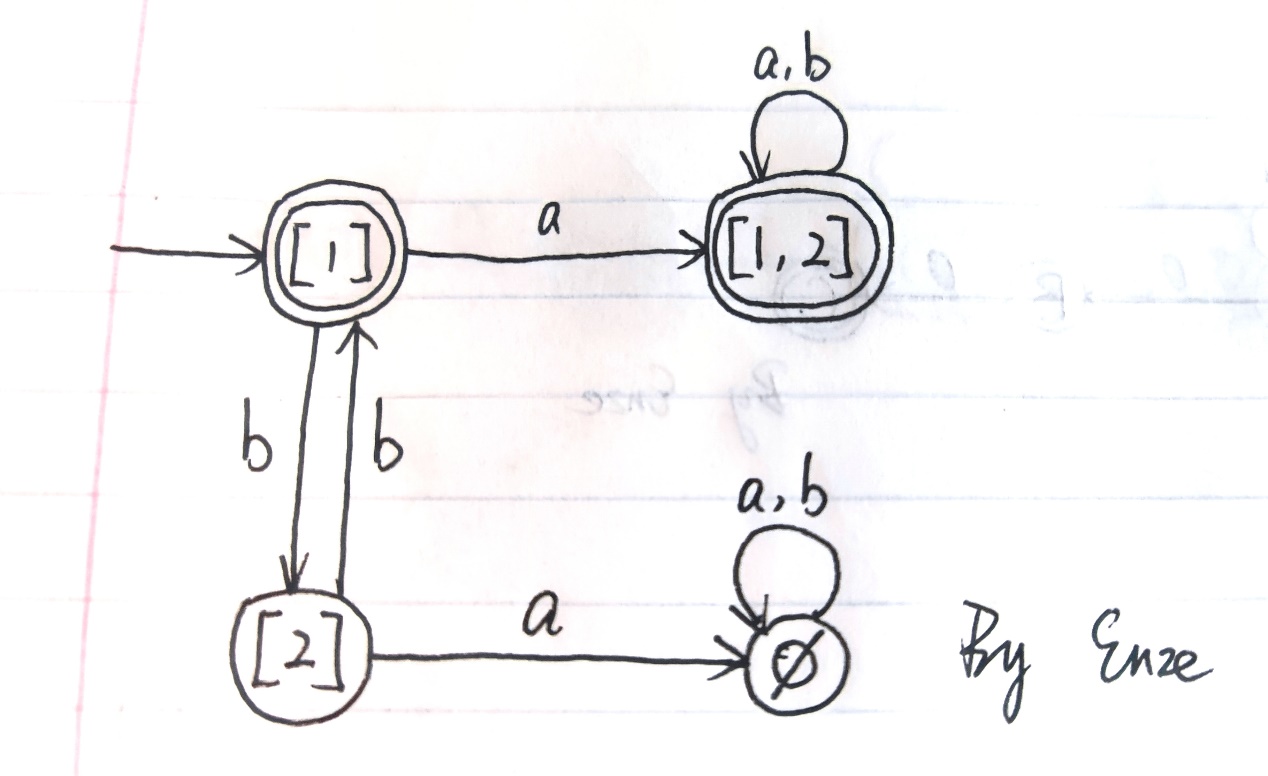


|  |  |  |
| --- | --- | --- |
| States \ Symbols |  |  |
|  |  |  |
|  |  |  |

Transition table for the equivalent DFA:

|  |  |  |
| --- | --- | --- |
| States |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

The equivalent DFA:



c) Problem 1.20 a,c,e,g, on page 86.



members:

(1)

(2)

not members:

(1)

(2)



members:

(1)

(2)

not members:

(1)

(2)



members:

(1)

(2)

not members:

(1)

(2)



members:

(1)

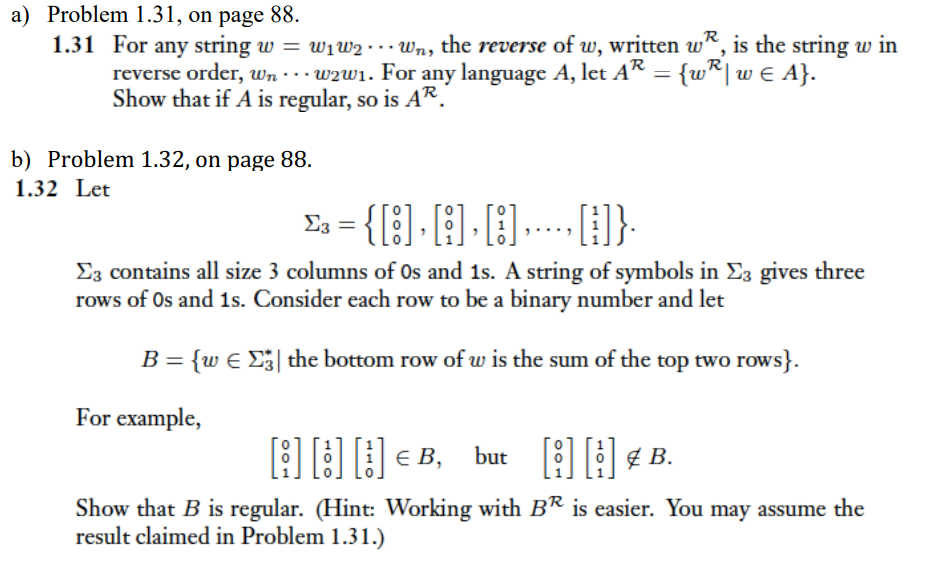
(2)

not members:

(1)

(2)

Problem Four



**Answers**

a)

Assume is regular, so there exists a DFA that recognizes

Then we can make some changes on it to convert it to a NFA that recognizes .

First, in , assume:

Conversion on to be :

(1) Reverse all the arrows

(2) Add a new start state

(3) Add k new arrows from to and label symbol on each of it.

(4) is the only accept state in

So the formal definition of should be:

Prove that accepts any string :

Assume that and accepts

So there exists states that

and

Then in , we can find a path (following defined above) from to

As and , accepts .

So accepts any string , i.e., recognizes

Then, as is a NFA, there exists a DFA that is equivalent to , which also recognizes .

So that is also regular. Q.E.D.

b)

Let’s focus on .

For example,

Assume , Let ,

and create

where represents that on the i-th bit, what’s the value the bottom row “owe” the top two rows that next higher bit needs to “repay”. Therefore, each only needs to inherit half of on the previous bit (the remainder has been “repaid” by immediately).

As , it is obvious that:

For example, in , we see:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Then, as and , by PMI it is easy to prove that

So now we can define 2 states to represent (and a to represent the situation when mismatches formula (1) as introduced above), and then we can give the formal definition of the DFA that recognizes :

So that is regular, and by the result claimed in Problem 1.31, is regular. Q.E.D.