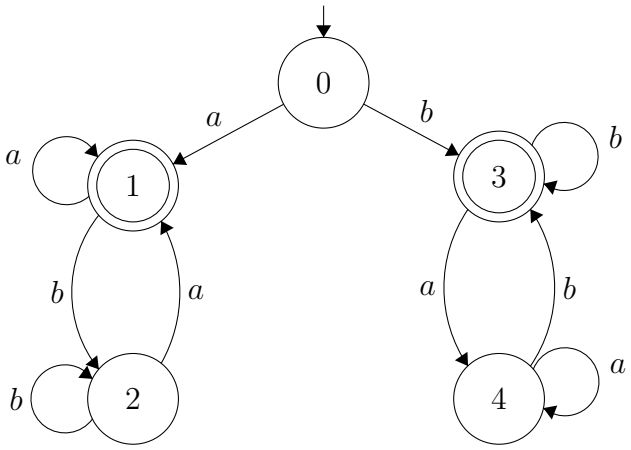
CITS2211 Discrete Structures
Week 10 Exercises – Finite State Machines and Regular Expressions

October 2022

1 Finite State Machines

1. For the following FSM, M, document its parts by listing each of the following:
   a) alphabet b) states c) starting state d) accepting states e) transitions.
   List three strings (from Σ^*) that M accepts.
   List three strings (from Σ^*) that M does not accept.



SOLUTION: a) alphabet $\Sigma = \{a, b\}$

b) states $Q = \{0, 1, 2, 3, 4\}$

c) starting state $q_0 = 0$

d) accepting states $F = \{1, 3\}$

e) transitions: (hint: be systematic in how you make your list either order by states or by inputs

$\delta(0, a) = 1, \delta(1, a) = 1, \delta(2, a) = 1, \delta(3, a) = 4, \delta(4, a) = 4$

$\delta(0, b) = 3, \delta(1, b) = 2, \delta(2, b) = 2, \delta(3, b) = 3, \delta(4, b) = 3$

Three accepted strings: *aaaa*, *abbbba*, *bababbb*

Three non-accepted strings: *ab*, *baba*, *baaaaa* (another example could be the empty string ϵ and many more)

2. Design deterministic FSMs to recognise each of the following languages:

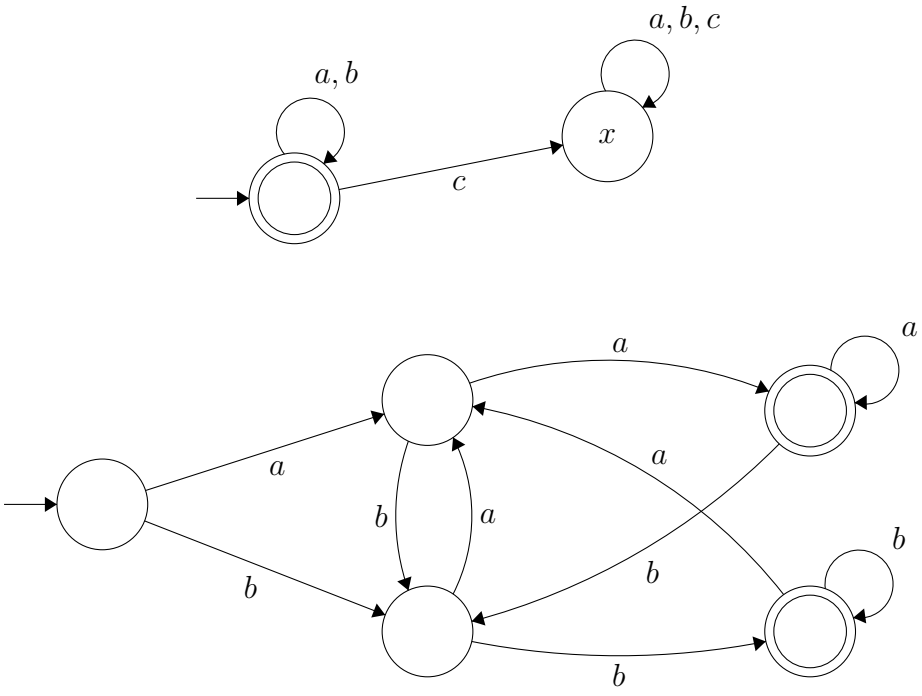
   (a) Words over the alphabet $\{a, b, c\}$ that contain no c.

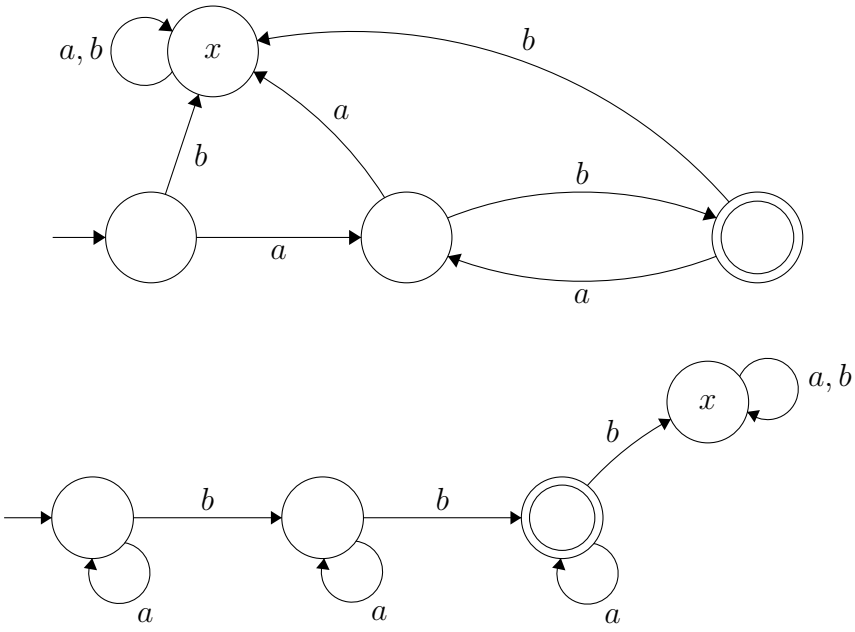
   (b) Words over the alphabet $\{a, b\}$ such that the last two symbols are the same.

   (c) Words over the alphabet $\{a, b\}$ of the form $(ab)^n$ for $n > 0$ (that is ab repeated n times).

   (d) Words over the alphabet $\{a, b\}$ that contain exactly 2 bs.

SOLUTION:

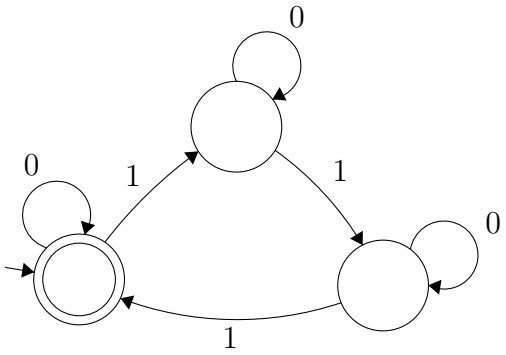




Note: x denotes the error state that is included in the diagrams above for completeness.

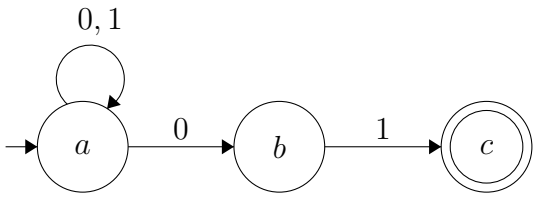
3. Define a divisible by 3 checker as a machine that accepts a string from the alphabet $\{0, 1\}$ if and only if the *sum of its digits* is divisible by 3. (Hint: The sum of the empty string ϵ is zero.)
   Design an FSM to implement a divisible by 3 checker.

SOLUTION: This definition implies the string contains exactly 0, 3, 6, 9, ... 1s and any number of 0s.



4. Design a *nondeterministic* FSM (NFSM) to recognise any binary string that ends in 01. Hint: you only need 3 states.

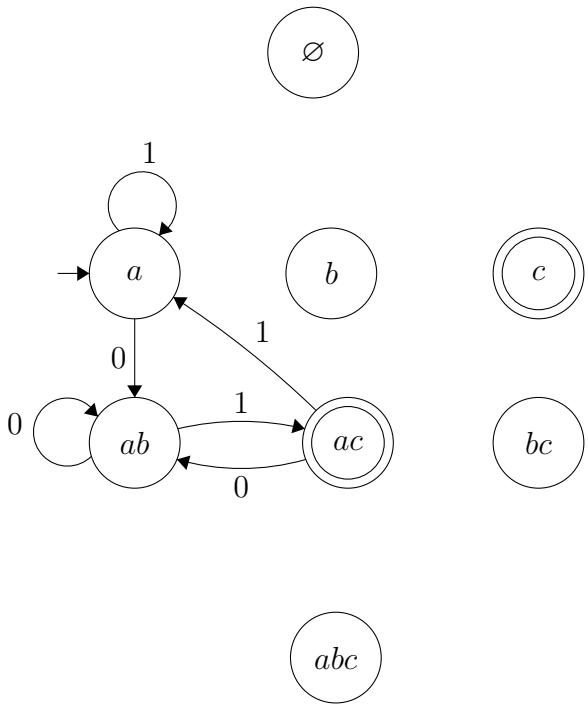
SOLUTION:



5. Convert your NFSM from the previous question into a deterministic FSM for the same language.

SOLUTION:

Note that many of these states can not be reached from the inital state. I have left them in to remind you of the construction. See Sipser for details of the construction algorithm.



6. Devise an FSM to solve the farmer, wolf, goat and cabbage problem and find the solutions. This problem involves a farmer, a wolf, a goat and a cabbage all on one side of a river. There is a boat but the farmer can carry only one passenger at a time. The farmer wants to get them all to the other side of the river. However, left alone, the wolf will eat the goat, and the goat will eat the cabbage. How do they all get to the other side?

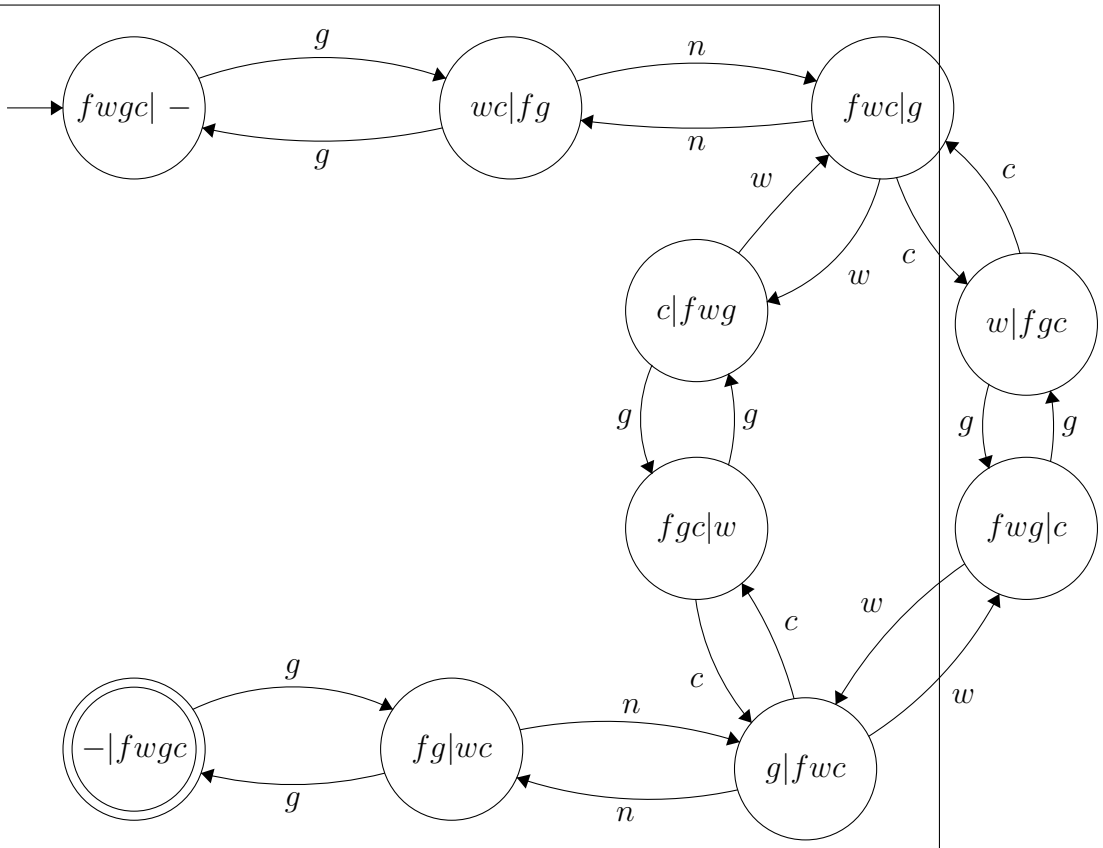
   (The cabbage counts as a passenger!)

SOLUTION: In order to design an FSM for the given problem we need to think about the possible states and inputs. Let the states be the locations of the farmer (f), the wolf (w), the goat (g) and the cabbage (c), where for example

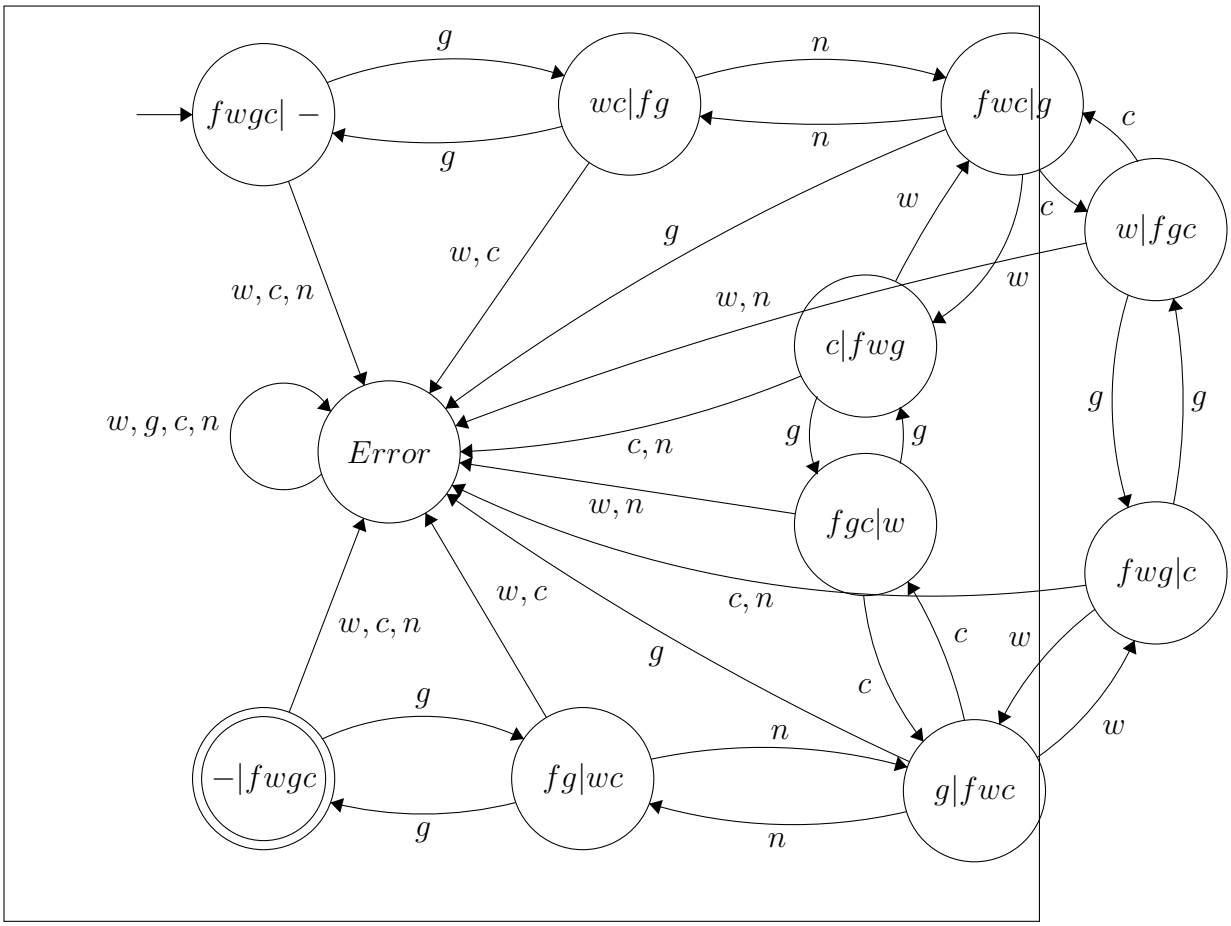
$$fw|cg$$

means that the farmer and wolf are on one side of the river while the goat and the cabbage are on the other side. The possible input strings are $\Sigma = \{w, g, c, n\}$, which mean:

- The farmer crosses the river with the wolf (w)
- The farmer crosses the river with the goat (g)
- The farmer crosses the river with the cabbage (c)
- The farmer crosses the river with nothing (n)



The diagram above shows the FSM without the error state. All transitions not shown in the diagram are assumed to go to the error state. The diagram below has the error state included which makes the diagram a bit crowded.



7. *Source: Sipser 1.12*

   Let $D = \{w \mid w$ contains an even number of as and
   an odd number of bs and
   does not contain the substring ab $\}$

   Give an FSM over the alphabet $\Sigma = \{a, b\}$ with 5 states that recognises D.

   Hint: describe D more simply.

SOLUTION:

The reasoning is really more important than the answer here:

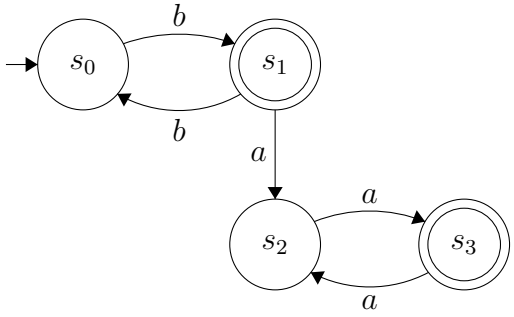
First, think about the machines for an even number of as (2 states) and an odd number of bs (3 states).

Remember that 0 is even, but not odd. So you must have 1 or more bs, but can have 0 as.

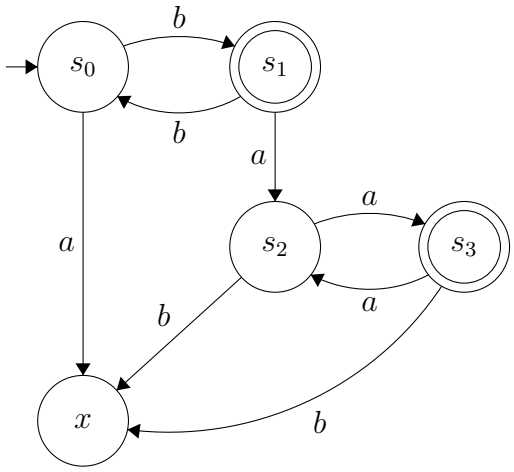
Next, notice that since D can not ever contain the substring ab that all the bs have to occur before the as. So we can concatenate (that is, join in sequence) our machines for the odd bs and even as.

The only addition that is needed is that you need an extra state to move into the even a counting part, so that the machine will not accept any more bs.

Here is a machine that does it, but not showing the non-accepting transitions.



Here is the same machine but with the transitions to the non-accept state added.



2 Regular expressions and languages

1. Give a regular expression for the following sets:

   a) the set of all binary strings (i.e. strings of 0s and 1s) beginning with 1 and ending with 1
   b) the set of all strings of 0s and 1s containing exactly two 1s
   c) the set of all strings of 0s and 1s having an odd number of 1s
   d) the set of all strings of 0s and 1s containing at least one 0
   e) the set of all strings of 0s and 1s where each 0 is followed by two 1s
   f) the set of all strings of 0s and 1s containing exactly three 0s

SOLUTION:

(a) $1(0+1)^*1+1$
That is, a 1 followed by any number of 0s or 1s, finishing with a 1, or just a single 1

(b) $0^*10^*10^*$
That is, two 0s with 0,1 or more 1s before, after and between them.

(c) $0^*1(0^*10^*1)^*0^*$
That is, at least 1 occurrence of 1 and then sequences with 2 1s repeated, with 0 or more 0s in between any of these.

(d) $(0+1)^*0(0+1)^*$
That is, any binary string, a 0 and then any binary string.

(e) $(011+1)^*$ or $(1^*(011)^*1^*)^*$
Allow for multiple occurrences of 011 which may be separated by any length sequences of 0 or more 1s. For example test this string: 111 011 1 011 011 111 (the spaces are just for readability - not part of the string). remember that the * operator means 0 or more repetitions. So 1*(011) can be 011 or 1011 or 1111111011 etc. Also, make sure that sequences with *no* 0s are accepted too.

(f) $1^*01^*01^*01^*$
That is three 1s with 0,1 or more 0s before, after and between them.

Other answers are possible.

2. Find a regular expression for the language L consisting of all strings over $\{0, 1\}$ with no consecutive zeros (that is, any string containing 00 is *not* in the language).

> SOLUTION: This means every 0 must be followed by a 1 or it can be the last 0 of the string (followed by nothing). Possible regular expressions are for example
>
> $1^*(011^*)^*(\epsilon + 0)$
> or
> $(01 + 1)^*(\epsilon + 0)$

3. Does the string 01110111 belong to the regular set $(1^*01)^*(11 + 0^*)$? Justify your answer.

> SOLUTION:
>
> Yes. The first part 011101 matches $(1^*01)^*$ by looping twice. The second part 11 matches $(11 + 0^*)$ by selecting the first branch.

4. Does the string 011100101 belong to the regular set $01^*10^*(11^*0)^*$? Justify your answer.

> SOLUTION:
>
> No. Going through the string backwards we know that in order for the string to belong to the regular set $01^*10^*(11^*0)^*$ the last 1 cannot be part of the set expressed by $0^*(11^*0)^*$ as this part can only end in 0 (or the empty string). This means the whole string must match 01^*1. That does not match because 01^*1 means the string must contain exactly one 0 at the start but no further 0s.