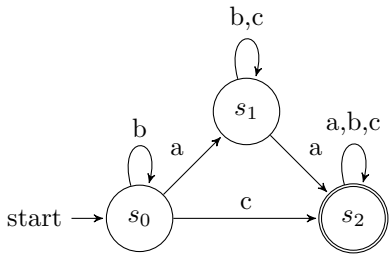
Assignment 2 by Lucas Karlsson

1. Give an informal, intuitive description of the language accepted by the DFA defined in the assignment question.

Solution:
Below follows the automata you can design from the given DFA. Using this you can informally and intuitively describe what is going on.



Basically, you can describe the language as following, every word either start by one or more b's or one c, if you start on one or more b's one a is directly followed then you can have zero or more b's or c's followed by another a and then in the end you can have zero or more a's b's or c's. If you started with a c it will be followed by zero or more a's b's or c's. We could write it more formally as a regular expression $b^*a(b+c)^*a(a+b+c)^* + b^*c(a+b+c)^*$

Example of a word that would be accpeted by this language would be "bbabca" or just plain "c".

2. In this exercise the alphabet $\Sigma = \{a, b\}$

Solution 1:
Here we want to define a DFA A over Σ such that $L(A) = \{w \in \Sigma^* \mid \nexists u, v \in \Sigma^*.w = ubbav\}$. The solution for this is in the file "2point1ass2.jff". I'll explain what my automata does below.

"The words that is accepted by our language can start with either one b or zero or more a's if we start with a b it has to be followed by another a or one or more b's and then followed by an a into a unaccepted state that you cannot escape. This translates into as soon as we get two or more b's we will get stuck and the words is not accepted by our language"

Solution 2:
Here we want to define a DFA B over Σ that accepts exactly those string that have an even number of occurrences of a. The solution for this is in the file "2point2ass2.jff" I'll explain what my automata does below.

"The words that is accepted by our language can start with either zero or more b's or one a followed by zero or more b's followed by another a and we are back where we started. This way we will have a even number of a's everytime we reach our final state."

Solution 3:
Here we want to construct $A \cdot B$ i.e use the product construction to build an automaton that accepts the language $L(A) \cap L(B)$. The solution for this is in the file "2point3ass2.jff" I wont explain what the automata does here because it heavily relies on my previous answers but I can explain the process of combining the two DFAs. To combine two DFAs we calculate $A \cdot B$ by doing $\{A_0, A_1, A_2, A_3\} \cdot \{B_0, B_1\}$ the new states will be:

$$\{A_0B_0, A_0B_1, A_1B_0, A_1B_1, A_2B_0, A_2B_1, A_3B_0, A_3B_1\}$$

To calculate the states and their transition functions we use the new states above and combine what a does to A_0 and B_0 and same for b and then the next state in the set. Below is the table that you can create by doing the method above. The calculations looks as following:

$\delta(A_0B_0, a) = \delta(A_0, a) \cup \delta(B_0, a) = \{A_0, B_1\}$
$\delta(A_0B_0, b) = \delta(A_0, b) \cup \delta(B_0, b) = \{A_1, B_0\}$

$\delta(A_0B_1, a) = \delta(A_0, a) \cup \delta(B_1, a) = \{A_0, B_0\}$
$\delta(A_0B_1, b) = \delta(A_0, b) \cup \delta(B_1, b) = \{A_1, B_0\}$.
.
.
.
.
$\delta(A_3B_1, a) = \delta(A_3, a) \cup \delta(B_1, a) = \{A_3, B_0\}$
$\delta(A_3B_1, b) = \delta(A_3, b) \cup \delta(B_1, b) = \{A_3, B_1\}$

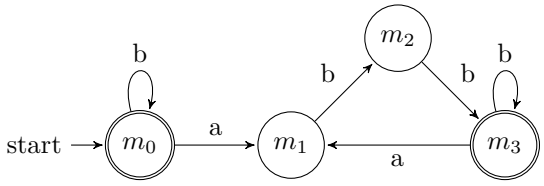
	State	a	b
$\rightarrow *$ (q0)	A_0B_0	A_0B_1	A_1B_0
(q1)	A_0B_1	A_0B_0	A_1B_1
$*$ (q2)	A_1B_0	A_0B_1	A_2B_0
(q3)	A_1B_1	A_0B_0	A_2B_1
$*$ (q4)	A_2B_0	A_3B_1	A_2B_0
(q5)	A_2B_1	A_3B_0	A_2B_1
(q6)	A_3B_0	A_3B_1	A_3B_0
(q7)	A_3B_1	A_3B_0	A_3B_1

We can then use this table to construct our automata using JFLAP, which is what we have done in the file mentioned above and it will hold for $L(A) \cap L(B)$.

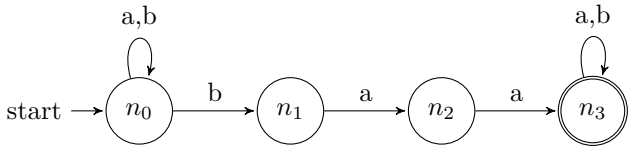
3. In this exercise the alphabet $\Sigma = \{a, b\}$

Solution 1:
We want to define a NFA A without ϵ-transitions over Σ such that $L(A) = M \cup N$ where $M = \{w \in \Sigma^* \mid \exists u, v \in \Sigma^*.w = ubaav\}$ And N contains exactly those strings in Σ^* where every occurence of a is directly followed by two or more occurences of b.

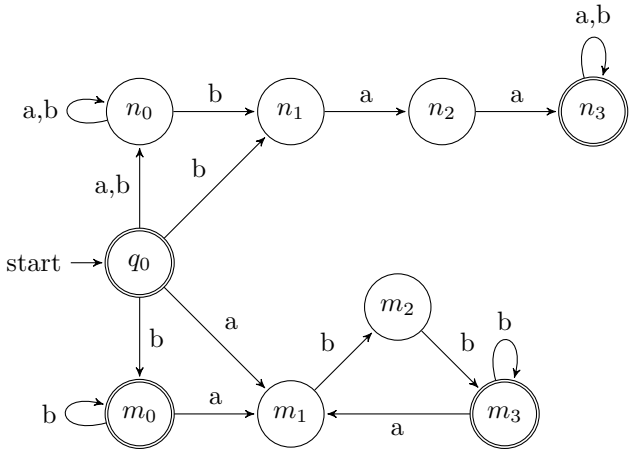
Defining M, we want M to only contain words that contain the substring "baa". We can do this using the following automata:



Defining N, we want N where every time we have a occurence of a is followed by two occurences of b. We can do this using the following automata:



Now we have to combine M and N by creating a new initial state that acts as both the M initial state and N initial state this can be done by adding the q_0 state below



You can now use q_0 as the initial state both for M and N and such we have solved the problem. This is also the automata found in the file "3point1ass2.jff".

Solution 2:
We want to use the subset construction to build a corresponding DFA and we are allowed to omit inaccesible states. This can be done starting at q_0 and doing $\delta(q_0, a)$ and $\delta(q_0, b)$ and using the states that we get from that calculation doing the same δ function for a and b for them. I will do the calculations here in latex and filling in my table as I go along. I will also construct a DFA in JFLIP that will go under the name "3point2ass2.jff".

$\delta(q_0, a) = \delta_n(q_0, a) = \{n_0, m_1\}$
$\delta(q_0, b) = \delta_n(q_0, b) = \{n_0, n_1, m_0\}$

$\delta(n_0m_1, a) = \delta_n(n_0, a) \cup \delta_n(m_1, a) = \{n_0\}$
$\delta(n_0m_1, b) = \delta_n(n_0, b) \cup \delta_n(m_1, b) = \{n_0, n_1, m_2\}$
$\delta(n_0n_1m_0, a) = \delta_n(n_0, a) \cup \delta_n(n_1, a) \cup \delta_n(m_0, a) = \{n_0, n_2, m_1\}$
$\delta(n_0n_1m_0, b) = \delta_n(n_0, b) \cup \delta_n(n_1, b) \cup \delta_n(m_0, b) = \{n_0, n_1, m_0\}$

$\delta(n_0, a) = \delta_n(n_0, a) = \{n_0\}$
$\delta(n_0, b) = \delta_n(n_0, b) = \{n_0, n_1\}$
$\delta(n_0n_1m_2, a) = \delta_n(n_0, a) \cup \delta_n(n_1, a) \cup \delta_n(m_2, a) = \{n_0, n_2\}$
$\delta(n_0n_1m_2, b) = \delta_n(n_0, b) \cup \delta_n(n_1, b) \cup \delta_n(m_2, b) = \{n_0, n_1, m_3\}$
$\delta(n_0n_2m_1, a) = \delta_n(n_0, a) \cup \delta_n(n_2, a) \cup \delta_n(m_1, a) = \{n_0, n_3\}$
$\delta(n_0n_2m_1, b) = \delta_n(n_0, b) \cup \delta_n(n_2, b) \cup \delta_n(m_1, b) = \{n_0, n_1, m_2\}$

$\delta(n_0n_1, a) = \delta_n(n_0, a) \cup \delta_n(n_1, a) = \{n_0, n_2\}$
$\delta(n_0n_1, b) = \delta_n(n_0, b) \cup \delta_n(n_1, b) = \{n_0, n_1\}$
$\delta(n_0n_2, a) = \delta_n(n_0, a) \cup \delta_n(n_2, b) = \{n_0, n_3\}$
$\delta(n_0n_2, b) = \delta_n(n_0, b) \cup \delta_n(n_2, b) = \{n_0, n_1\}$
$\delta(n_0n_1m_3, a) = \delta_n(n_0, a) \cup \delta_n(n_1, a) \cup \delta_n(m_3, a) = \{n_0, n_2, m_1\}$
$\delta(n_0n_1m_3, b) = \delta_n(n_0, b) \cup \delta_n(n_1, b) \cup \delta_n(m_3, b) = \{n_0, n_1, m_3\}$
$\delta(n_0n_3, a) = \delta_n(n_0, a) \cup \delta_n(n_3, a) = \{n_0, n_3\}$
$\delta(n_0n_3, b) = \delta_n(n_0, b) \cup \delta_n(n_3, b) = \{n_0, n_1, n_3\}$

$\delta(n_0n_1n_3, a) = \delta_n(n_0, a) \cup \delta_n(n_1, a) \cup \delta_n(n_3, a) = \{n_0, n_2, n_3\}$
$\delta(n_0n_1n_3, b) = \delta_n(n_0, b) \cup \delta_n(n_1, b) \cup \delta_n(n_3, b) = \{n_0, n_1, n_3\}$

$\delta(n_0n_2n_3, a) = \delta_n(n_0, a) \cup \delta_n(n_2, a) \cup \delta_n(n_3, a) = \{n_0, n_3\}$
$\delta(n_0n_2n_3, b) = \delta_n(n_0, b) \cup \delta_n(n_2, b) \cup \delta_n(n_3, b) = \{n_0, n_1, n_3\}$

What I've done here is calculated the new set for every transitionsfunction starting from q_0 and expanding to q_0:s next of kin. The structure of the calculations above is every new block of functions is a "sub" set of the previous block.

		$State$	a	b
$\rightarrow$	$*$	q_0	n_0m_1	$n_0n_1m_0$
		n_0m_1	n_0	$n_0n_1m_2$
	$*$	$n_0n_1m_0$	$n_0n_2m_1$	$n_0n_1m_0$
		n_0	n_0	n_0n_1
		$n_0n_1m_2$	n_0n_2	$n_0n_1m_3$
		$n_0n_2m_1$	n_0n_3	$n_0n_1m_2$
		n_0n_1	n_0n_2	n_0n_1
		n_0n_2	n_0n_3	n_0n_1
	$*$	$n_0n_1m_3$	$n_0n_2m_1$	$n_0n_1m_3$
	$*$	n_0n_3	n_0n_3	$n_0n_1m_3$
	$*$	$n_0n_1n_3$	$n_0n_2n_3$	$n_0n_1n_3$
	$*$	$n_0n_2n_3$	n_0n_3	$n_0n_1n_3$