1. Determinate Finite Automata (DFA)

Given the specifications, the DFA had to be able to read and understand the following language:

{w | w contains at least two 0's and at most one 1} where ∑ = {0,1}

There were two variables that had to be monitored, the number of 1s and 0s. While the PDA can't store this information into variables like the programming language C can, the states themselves can represent the storing of that information. By laying out the possible number of 0s on the y-axis and the 1s on the x, the states can be displayed accordingly.

		Number of 1s		
		0	1	2 +
Number of 0s	0	S0	S1	S2
	1	S3	S4	S5
	2 +	S6	S7	S8
Green - Accept States Red - Reject States Orange - Currently Reject, but leads to Accept				

When there are 2 or more 0s, the 0 requirement is fulfilled, so they can be combined together into one column, instead of having columns 3, 4, +. On a similar aspect, when there are 2 or more 1s, the 1 requirement has not been fulfilled and it is impossible for the language to remove those 1s (once the system has read it, there's no going back). Therefore the 3rd column is completely red to represent failed states. The green squares represent accepted states, so grammar ending in these states are accepted by the language. The orange states can be seen as uncertain states, if the grammar ends in these states, it fails, but if the following characters are fortunate, then the grammar can reach States 6 or 7.

What is important to note is that States 2, 5 and 8 all have the same properties (When reached, every input correlates to the same state). This means that these 3 states can be combined into one state, which will be State 2. Therefore S5 and S8 in the table can be changed to S3, and the numbering of the other states can be rearranged.

By following the above table and the changes, the following State Diagram and 5-Tuple Specification can be generated:

Q = {q0, q1, q2, q3, q4, q5, q6}
∑ = {0,1}
q0 = {q0}
F = {q5, q6}
□ = See Transition Table

Start State	Input	Finish State
q0	0	q3
q0	1	q1
q1	0	q4
q1	1	q2
q2	0	q2
q2	1	q2
q3	0	q6
q3	1	q4
q4	0	q7
q4	1	q2
q5	0	q5
q5	1	q6
q6	0	q6
q6	1	q2

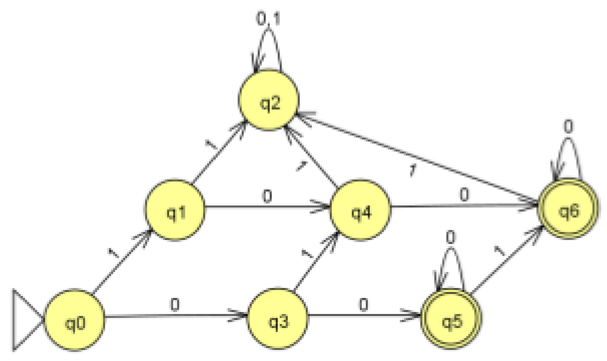

Fig 1) Deterministic Finite Automaton

2. Push Down Automata (PDA)

The Push Down Automata required more forethought than the DFA. While the DFA only requires jumping from one state to another, the PDA needs to initialize a stack, and have the ability to manipulate it (as mentioned previously). This results in a PDA being able to recognise the following language.

PDA : {$a^i b^j c^k$ | i, j, k >= 0 and i = j or i = k }

This language is in the form of a series of As, followed by a series of Bs, followed by a series of Cs. What's important to note is that either the number of As and Bs have to match or the number of As and Cs have to match. This is possible to monitor by using indeterminism.

The previous Automaton was deterministic, as every state had a set output for each input. This PDA will end up being indeterministic, as the PDA is monitoring whether i and j are equal or i and k are equal. This will be conducted by iterating over the possible indeterminate branches, with the first iteration monitoring the state of i and j. By adding 'x' to the stack whenever an A appears, the stack will represent i (the number of x's). Upon reading that first B, the first branch appears, an 'x' is removed and the next digit is read. By removing 'x's every time a B is read, the stack will represent the number of As minus the number of Bs. If there are more As than Bs, the stack's top character will be an x. If there are more Bs than As, the stack will be empty, with even the '$' removed. On an even stack, the '$' remains and a C begins to be read.It proceeds to read and ignore all Cs until the string is finished, and if the remaining character in the stack is '$', the PDA accepts the grammar.

If the grammar isn't accepted, the second iteration starts. This iteration is exactly the same as the first, however this time the Bs are ignored and the Cs pop characters from the stack. This is also continued until either the stack has nothing to remove or if there are no more Cs to read. A successful Grammar will only contain '$' in the stack.

By following the rules, the following State Diagram and 6-Tuple Specification can be generated:

6-Tuple Specification
Q = {q0, q1, q2, q3, q4, q5, q6}
∑ = {a, b, c}
Γ = {\$, γ, x}
q0 = {q0}
F = {q6}
□ = See Transition Table

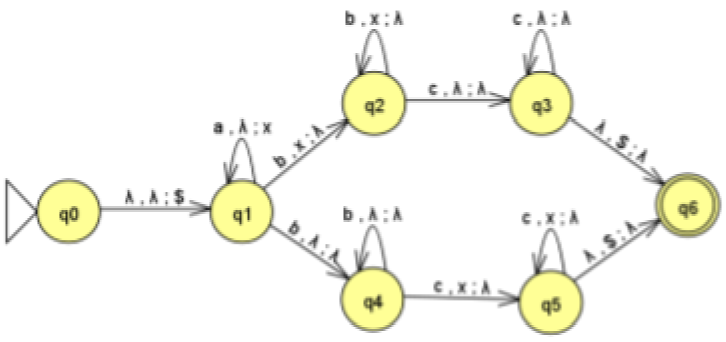

Fig 2) Push Down Automaton

Start State	Input	Finish State
q0	$\gamma, \gamma \rightarrow$ \$	q3
q1	a, $\gamma \rightarrow$ x	q1
q1	b, x $\rightarrow \gamma$	q2
q1	b, $\gamma \rightarrow \gamma$	q4
q2	b, x $\rightarrow \gamma$	q2
q2	c, $\gamma \rightarrow \gamma$	q3
q3	c, $\gamma \rightarrow \gamma$	q3
q3	γ, \$ $\rightarrow \gamma$	q6
q4	b, $\gamma \rightarrow \gamma$	q7
q4	c, x $\rightarrow \gamma$	q2
q5	c, x $\rightarrow \gamma$	q5
q5	γ, \$ $\rightarrow \gamma$	q6