# **University of Calgary**

# 

# **CPSC 313: Introduction to Computability, Winter 2018**

# 

# **Assignment #1**

**For**

**Samin Sabokrohiyeh, Dr. Wayne Michael Eberly**

**By**

**Jesus Cuadra (ID: 30002376 | Email:** [**jesus.cuadrakaram@ucalgary.ca**](mailto:jesus.cuadrakaram@ucalgary.ca)**)**

**BenKun Chen (ID: 30005337 | Email:** [**benkun.chen@ucalgary.ca**](mailto:benkun.chen@ucalgary.ca)**)**

**Adrienne Lee (ID: 10087046 | Email:** [**adrienne.lee1@ucalgary.ca**](mailto:adrienne.lee1@ucalgary.ca)**)**

**1. Let Σ = {a, b, c} and consider the language**

**L = {ω ∈ Σ ⋆ | ω has length at least two and the second-to-last and last symbols in and last symbols in and last symbols in and last symbols in ω are different}.**

In order to recognize this language, it is necessary for a DFA to remember which one of the following cases holds for the string that has been processed, so far.

* The string has length of zero, so that it belongs to the set

;

this will corresponding to a state in the DFA being constructed.

* The string in which has only one symbol or the last symbol is same as the second-to-last symbol, so that it belongs to the sets

ω in which the only symbol is an ‘a’ or the last symbol is an ‘a’ and the second-to-last symbol is an ‘a’} ;

ω in which the only symbol is a ‘b’ or the last symbol is an ‘b’ and the second-to-last symbol is a ‘b’} ;

ω in which the only symbol is a ‘c’ or the last symbol is a ‘c’ and the second-to-last symbol is a ‘c’} ;

this will corresponding to a state in the DFA being constructed.

* The string has length at least two, in which the last symbol is different than the second-to-last symbol, so that it belongs to the sets

in which the last symbol is an ‘a’ and the second-to-last symbol is a ‘b’ or a ‘c’};

in which the last symbol is a ‘b’ and the second-to-last symbol is a ‘a’ or ‘c’};

in which the last symbol is a ‘c’ and the second-to-last symbol is ‘a’ or ‘b’};

this will corresponding to a state in the DFA being constructed.

The first “sanity check” has been passed, because only a finite number of sets (and states)

have been identified.

* The second “sanity check” has been passed, because every string include some fixed lengths and symbol(s). Length must either be zero, one or two. Each string that has length > 0 will contain unique symbol(s) and it cannot be more than one of these things at the same time. That is,

Thus, every string belongs to exactly one of the sets which are shown above.

Now, since , the corresponding state called will be the start state

* The third “sanity check” has also been checked:

If are as described above, then

while . We can now define the set of accepting states for the DFA being designed:

* The fourth “sanity check” has also been checked:

, so that ; , so that; , so that .

, so that ; , so that; , so that .

, so that ; , so that; , so that .

, so that ; , so that; , so that .

, so that ; , so that; , so that .

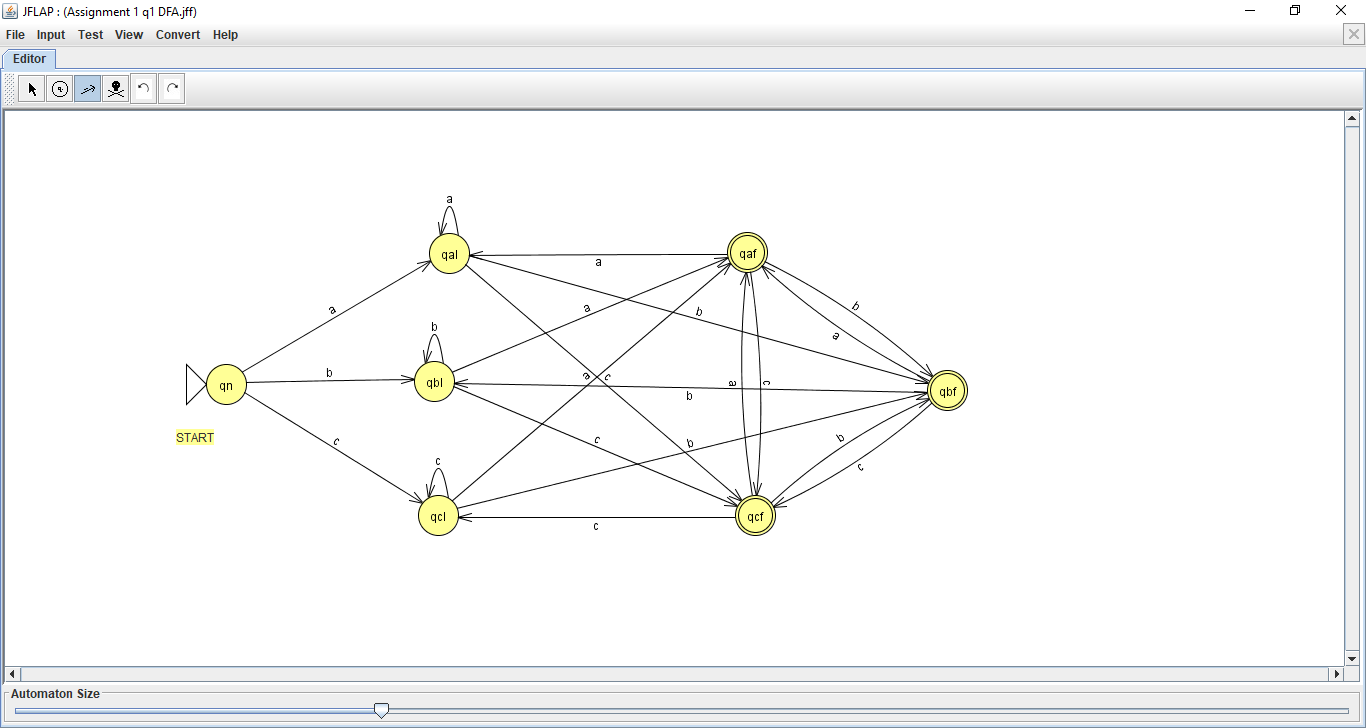
, so that ; , so that; , so that .

, so that ; , so that; , so that .

The “design process” has now been completed and a table for a transition function can now be given as follows:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **a** | **b** | **c** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Together with the above information, this corresponds to a DFA that looks like the following:



Therefore, as was proven with the above sanities, This DFA defines the language L.

**2. Describe (but do not fully specify or write down) a deterministic finite automaton whose language is**

**= {ω ∈ Σ ⋆ | ω has length at least three and the third-to-last and second-to-last symbols in ω are different}.**

The DFA that defines the language above will include the previously proven DFA from the above question. The new DFA also includes the addition of six new states, which replace the a, b, and c-transitions out of the old final states of the DFA for L, as they now lead to the new six states. These six new states, which are all final states, are further divided into three pairs, one for each input. The first state of each pair is transitioned into by a state whose final symbol and new symbol are the same. This state will then transition to the looping state of its own symbol if the symbol input while in this state is the same as the last symbol of the string. However if the input symbol is different from the last symbol of the string, then the transition will be to what was previously the final state for the respective input symbol. The second state of each pair will be transitioned into if the input symbol is different to the last symbol of the string. These states will then transition to one of the other second states of the pairs if the input is different from the last symbol. If it is the same, it will instead transition to the first state of the pair if the symbol is the same as the last symbol of the string, where it will follow the logic of the first state of the pair, as defined above. This leads to a machine that remembers the previous two inputs, and will only finish if the second-to-last and third-to-last symbols are different, and a is therefore a DFA that defines the language .

**3. Finally, give a much simpler nondeterministic finite automaton whose language is the language defined in the previous question and explain (somewhat informally, and reasonably briefly) your answer is correct.**

This NFA is correct due to the fact that it will not lead to a final state if the string cannot be accepted. Using a looping state for each of the three possible inputs, we can restrict the third-to-last and second-to-last inputs to be different, and in case the string continues, it allows for the machine while in q4 to return to a looping state for further inputs. Finally, when the string is finished, the machine has the ability to end at the Final/Closing state, and therefore fulfills the requirements of an NFA that describes the language **.**