**ABERIN**, Shawn Emmanuel A. S15

**LUCENA**, Renz Erik R. 119

MCO1-1: Knowledge Understanding

# In this activity, you will have the opportunity to demonstrate your understanding of finite state machines, regular expressions, regular languages, and the conversions involving finite state machines with regular expressions. Additionally, the activity will provide you with an opportunity to delve into other relevant concepts related to finite state machines, such as optimization, Mealy and Moore machines.

***Instructions:***

1. You have the option to work on this activity with a partner. You and your partner are expected to learn from each other, clarify, and correct any misconceptions you may have. 2. All answers must

result from the combined efforts of you and your partner. 3. You may conduct research before

answering, but if you include information from another source (e.g., peers, Google, Chat-GPT and other tools) within your answer, please enclose it in quotation marks and provide proper

references/citations.

1. The due date for MCO1-1 is October 21, 2023, no later than 11:59 pm. Late submissions, even by a minute, will incur an appropriate deduction. There will be a 10-point deduction for every 3 hours of delay, including weekends.
2. When submitting your work, ensure that each item, question, and its corresponding answers are included.
3. Perfect score is 80.00, 80/80 = 100%.

**[5pts]** 1. In your own words, explain what a finite state machine is, and provide an example of its application in a real-world scenario.

Answer:

Finite state machines (FSM) are machines made of a finite set of states the machine traverses given an input. Those machines are defined by their finite set of states, a set of inputs, a set of transitions between states, an initial state, and a set of final states. A good example of an FSM is the typical vending machine (Gribkoff, 2013). Its "wait for input", "dispense", and "cancel" actions act as the states for the machine. The buttons for the power and the selection of products are the inputs of this machine. The transition occurs via selection and dispensation based on the input. Of course "wait" is the initial state while "dispense" or "change" acts as the final state. Another good example is locks. The inputs are keys allowing the mechanisms of the lock to transit between states until the final state of being unlocked is achieved or the key is rejected.

**[5pts]** 2. Discuss the strengths and limitations of finite state machines. Provide at least 2 strengths and 2 limitations.

## Answer:

* 1. Strengths
     1. Could simulate a lot of complex systems well
     2. Very easy to build and understand
     3. DFA’s deterministic behavior makes for an extremely tight, strict, and detailed data ﬂow
  2. Limits
     1. It can only operate on whatever it is reading at the present(no persistent memory)
     2. Computational power is not as strong as other machine types; Can only accept and transduce (generation is the transduction of an empty string)

**[5pts]** 3. List and explain (or illustrate) three practical applications of finite state machines.

## Answer:

1. Clocks: How does a clock know the time? We have a thing with a known stable frequency and we have something to count the oscillations. The oscillation count is the input and the internals will transit among various states until it changes the display to the current time.
2. Password generators: If you can accept an input, you know how to make one that is accepted. Password generators know the common tactics for making strong passwords and can make one for you.
3. Behavior models: We can model behaviors as states of being that transits to other behaviors given external stimuli.

**[5pts]** 4. Compare and contrast the differences (provide at least 2) between Nondeterministic Finite Automata (NFA) and Deterministic Finite Automata (DFA).

## Answer:

1. A NFA inputs transition into a set of states while inputs in DFA can only transition into one state.
2. NFA can have ε-transitions. DFA cannot.

**[5pts]** 5. Define a regular language and give an example of a regular language. Explain why it is (the example you provided) considered a regular language.

**Answer**: Regular language is a type of language recognizable by regular expressions and finite

automata (Moore, et al., n.d.). This is a specific class of formal languages that is characterized by simplicity. Examples of regular languages are repeating substrings and phone numbers. Phone numbers are easily described by regular expressions while repeating substrings are basic to

represent via DFA and NFA. To illustrate: (1) (0+1+2+3+4+5+6+7+8+9)^n, where n is the number of digits in a number; (2) M = {{q0,q1,q2}, {a,b}, δ, q0, q3 }, δ:

|  |  |  |
| --- | --- | --- |
| State | a | b |
| -> q0 | q1 | ∅ |
| q1 | ∅ | q2 |
| \* q2 | q1 | ∅ |

**[5pts]** 6. Create a regular expression that matches all valid email addresses. Explain your choice of regular expression components.

## Answer:

(a+b+c+d+e+f+g+h+i+j+k+l+m+n+o+p+q+r+s+t+u+v+w+x+y+z+A+B+C+D+E+F+G+H+I+J+K+L+M+N+O

+P+Q+R+S+T+U+V+W+X+Y+Z + 0+1+2+3+4+5+6+7+8+9 +! + # + $+ %+ &+ '+ \*+ '+' + - + / + =+ ?+ ^+ \_+ `+

{+ |+ }+

~+)^+@(a+b+c+d+e+f+g+h+i+j+k+l+m+n+o+p+q+r+s+t+u+v+w+x+y+z+)^+.(a+b+c+d+e+f+g+h+i+j+k+l

+m+n+o+p+q+r+s+t+u+v+w+x+y+z+)^+

(a+b+c+d+e+f+g+h+i+j+k+l+m+n+o+p+q+r+s+t+u+v+w+x+y+z+A+B+C+D+E+F+G+H+I+J+K+L+M+N+O

+P+Q+R+S+T+U+V+W+X+Y+Z + 0+1+2+3+4+5+6+7+8+9 +! + # + $+ %+ &+ '+ \*+ '+' + - + / + =+ ?+ ^+ \_+ `+

{+ |+ }+ ~+)^+ -> describes all the possible usernames - small and uppercase letters, numbers, and special characters included; Kleene plus is used to denote no empty username fields allowed.

(a+b+c+d+e+f+g+h+i+j+k+l+m+n+o+p+q+r+s+t+u+v+w+x+y+z+)^+ -> describes all possible mail servers, assuming the mail servers use small cases only and no numbers and special characters; Kleene Plus is used to denote no empty username fields allowed.

(a+b+c+d+e+f+g+h+i+j+k+l+m+n+o+p+q+r+s+t+u+v+w+x+y+z+)^+ -> describes all possible domain, assuming the domain use small cases only and no numbers and special characters; Kleene plus is used to denote no empty username fields allowed.

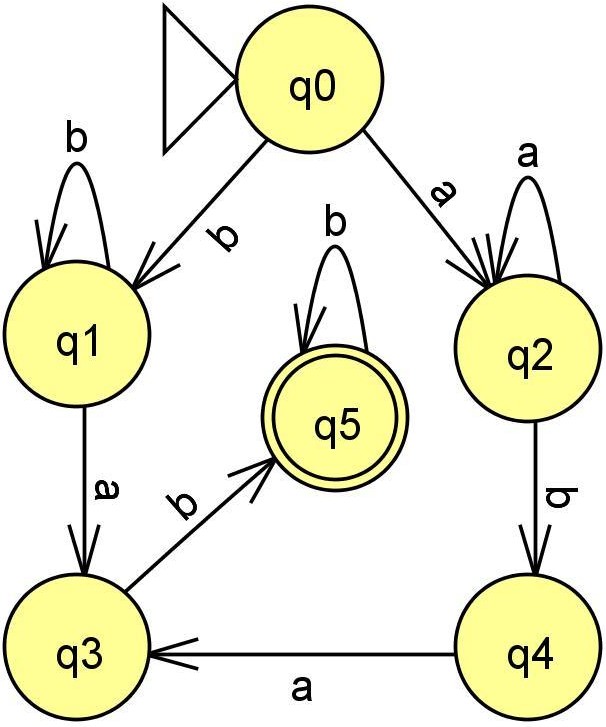
**[5pts]** 7. Given a regular expression **(a\*+ b\*) bab+**, convert it into an equivalent finite automaton. Give the complete definition of your resulting automaton.

## Answer:

M = {{q0, q1, q2, q3, q4, q5}, {a, b}, δ, q0, q5}

δ:

|  |  |  |
| --- | --- | --- |
| State | a | b |
| -> q0 | q2 | q1 |
| q1 | q3 | q1 |
| q2 | q2 | q4 |
| q3 | ∅ | q5 |
| q4 | q3 | ∅ |
| \* q5 | ∅ | q5 |



**[5pts]** 8. Create a regular expression for a simple programming language (say for C language) that recognizes valid variable names consisting of letters, digits and underscore, starting with a

letter.

Answer:

(a+b+c+d+e+f+g+h+i+j+k+l+m+n+o+p+q+r+s+t+u+v+w+x+y+z+A+B+C+D+E+F+G+H+I+J+K+L+M+N+O

+P+Q+R+S+T+U+V+W+X+Y+Z)((a+b+c+d+e+f+g+h+i+j+k+l+m+n+o+p+q+r+s+t+u+v+w+x+y+z+A+B+C+

D+E+F+G+H+I+J+K+L+M+N+O+P+Q+R+S+T+U+V+W+X+Y+Z)+(0+1+2+3+4+5+6+7+8+9)+’\_’)\*

**[5pts]** 9. Given a DFA, demonstrate how it can be minimized to a smaller DFA while recognizing the same language. Show the steps and final minimized DFA.

**Answer**: First we find which states are equivalent. States are equivalent if they behave similarly enough which means they both transit to equivalent states and end at an equivalent final state or in other words, their string successors are the same. Once all equivalences are found, we combine the equivalent states and we get a minimized dfa. The table below demonstrates the process. Here the machine M accepts strings with an odd number of 0’s. It is then minimized to M’ when q0 and q2 were found to be equivalent since they both behave similarly and so can be combined to a single state.

|  |  |  |  |
| --- | --- | --- | --- |
| **M=(Q,Σ,q0,F,δ)** | Q = {q0,q1,q2} | F = {q1} |  |
| δ: | Q/Σ | 0 | 1 |
|  | q0 | q1 | q0 |
|  | q1 | q2 | q1 |
|  | q2 | q1 | q2 |
| **State Equivalance Table** | | | |
| q0 | q0 |  |  |
| q1 | Distinguishable | q1 |  |
| q2 | Equivalent | Distinguishable | q2 |
| q0 and q2 collapses to state [q0q2] in the minimized Machine M' | | | |
| M' = {Q',Σ,[q0q2], q1,δ') | | Q' = {[q0q2],q1} | Σ = {0,1} |
| δ': | Q'/Σ | 0 | 1 |
|  | [q0q2] | q1 | [q0q2] |
|  | q1 | [q0q2] | q1 |

**[5pts]** 10. Explain the difference between regular expressions and context-free grammars. Provide examples of languages that can be described by each.

**Answer**: Regular expressions are basically a sequence of characters that can be expressed using alphabets, unions, intersections, differences, and loops. Context-free grammars act as a given a set of rules for producing possible strings in a language. To further differentiate the two, see the example below provided by Geeksforgeeks in their 2019 article:

Basic Example of RegEx: (0+1)\* = 0, 1, 0000, 1111, … n

Basic Example of CFG: S -> 0A | 1A | e, A -> 0A | 1A | e = 0, 1, 0000, 1111, … n

Both RegEx and CFG show the same thing 0 or 1 in a loop that starts at 0. While RegEx shows an algebraic approach, CFG opts for a set-ruled approach. RegEx shows the characters, their relationships, and what you can do with them in linear mathematical fashion. On the other

hand, CFG lays down rules to follow in order to accept the output of the machine.

CFG’s are also equivalent to pushdown automata, (Shawn took STALGCM last term too). Push down automata is more powerful since it has persistent memory called a stack. It can

remember previous input–output and allows for more complex languages. This means that we can balance parenthesis now since it knows how many open parenthesis, ‘(‘, need to be

closed by ,’)’. The complexity of the possible languages brings forth the rules-based nature of CFG’s.

**[5pts]** 11. Describe the Pumping Lemma for regular languages. Discuss how it can be used to prove that a language is not regular?

**Answer**: Pumping Lemma is a property of regular languages that is used to prove

non-regularity. To perform it, there are three conditions. For any regular language L, there exists an integer P, such that for all w in L |w|>=P. ‘w’ can be broken into three strings, w=xyz such that: (1) lxyl < P, (2) lyl > 1; (3) for all k>= 0: the string x(y^k)z is also in L (Priya, 2021).

This means regular languages must allow their strings to repeat a substring called lemma zero or more times. If a string in language L is proven to have a lemma that cannot be repeated zero or more times and still be a string in L, L is not regular.

**[5pts]** 12. Given the NFA M = ({a, b, c, d}, Σ, δ, a, {b,c}) where Σ = {0,1} and δ is defined as follows:

|  |  |  |
| --- | --- | --- |
| δ | 0 | 1 |
| a | {b,c} | ∅ |
| b | ∅ | {a,c} |
| c | {d} | {d} |
| d | {b} | {d} |

DFA M’ = (Q’,Σ, δ’, a, F’)

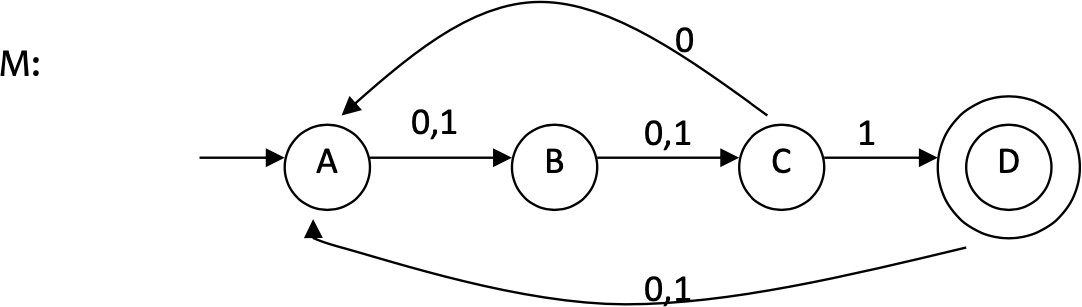
Q’ = {a,b,d,[bc],[acd],[bcd],[bd]}

Construct its equivalent DFA, and answer the following questions:

1. Determine the final state(s) in the equivalent DFA. F’ = {b,[bc],[acd],[bcd],[bd]}
2. List down (or in table form) the δ in the equivalent DFA.

|  |  |  |
| --- | --- | --- |
| δ’ | 0 | 1 |
| a | [bc] | n/a |
| b | n/a | [ac] |
| d | b | d |
| [bc] | d | [acd  ] |
| [acd] | [bcd] | [acd  ] |
| [bcd] | [bd] | [acd  ] |
| [bd] | [b] | [d] |

1. You are required to enter only the values for (a), (b) and (c) only (no need to simplify the expressions) and supply the regular expression accepted by the given finite automaton. You are not required to simplify the L(M). Assume state A is state 1, state B is state 2, state C is state 3 and state D is state 4.



1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | K=0 | K=1 | K=2 | K=3 |
| R11 | ε | ε |  |  |
| R12 | 0+1 | 0+1 |  |  |
| R13 | ∅ | ∅ | **(b) [2pts]** (0+1)(0+1) | (0+1)(0+1)(0(0+1)(0+1))\* |
| R14 | ∅ | ∅ | ∅ | (0+1)(0+1)((0)(0+1)(0+1))\*1 |
| R21 | ∅ |  |  |  |
| R22 | ε | ε |  |  |
| R23 | 0+1 | 0+1 | (0+1) |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| R24 | ∅ | ∅ | ∅ | (0+1)(0(0+1)(0+1))\*1 |
| R31 | 0 |  |  |  |
| R32 | ∅ | 0(0+1) |  |  |

2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| R33 | ε | ε | 0(0+1)(0+1) + ε |  |
| R34 | 1 | 1 | 1 | (0(0+1)(0+1))\*1 |
| R41 | 0+1 |  |  |  |
| R42 | ∅ | **(a) [2pts]** (0+1)(0+1) |  |  |
| R43 | ∅ | ∅ | (0+1)(0+1)(0+1) |  |
| R44 | ε | ε | ε | **(c) [4pts]** (0+1)(0+1)(0+1)(0(0+1)(0+1))\*1 |

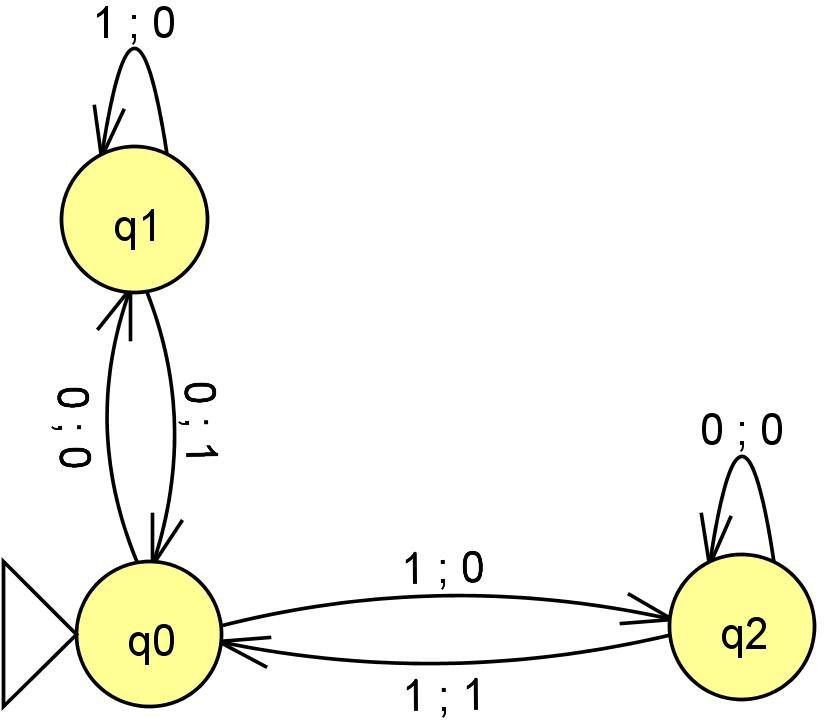
**[2pts]** Regular Expression for M: (0+1)(0+1)((0)(0+1)(0+1))\*1

## [5pts]

1. Study the given behaviour (given an input, a corresponding symbol is being produced)

below. Create a Mealy machine that has 3 states and captures the following input-output behaviour.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| Output | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |



1. Given a Moore machine, M= (Q, Σ, Δ, δ, λ, A) where Q = {A, B, C}, Σ = {0,1}, Δ = {0,1}, δ and λ are defined below:

|  |
| --- |
| λ |
| A | 0 |
| B | 1 |
| C | 0 |

|  |  |  |
| --- | --- | --- |
| δ | 0 | 1 |
| A | B | C |
| B | C | B |
| C | A | C |

You are required to find the equivalent Mealy machine, M’ and then answer the following questions:

* 1. **[3pts]** Give the λ table for M’.

|  |  |  |
| --- | --- | --- |
| λ’: | 0 | 1 |
| A' | 0 | 1 |
| B' | 0 | 1 |
| C' | 0 | 1 |

* 1. **[1pt]** What are the elements of Σ? Answer: Σ = {0,1}

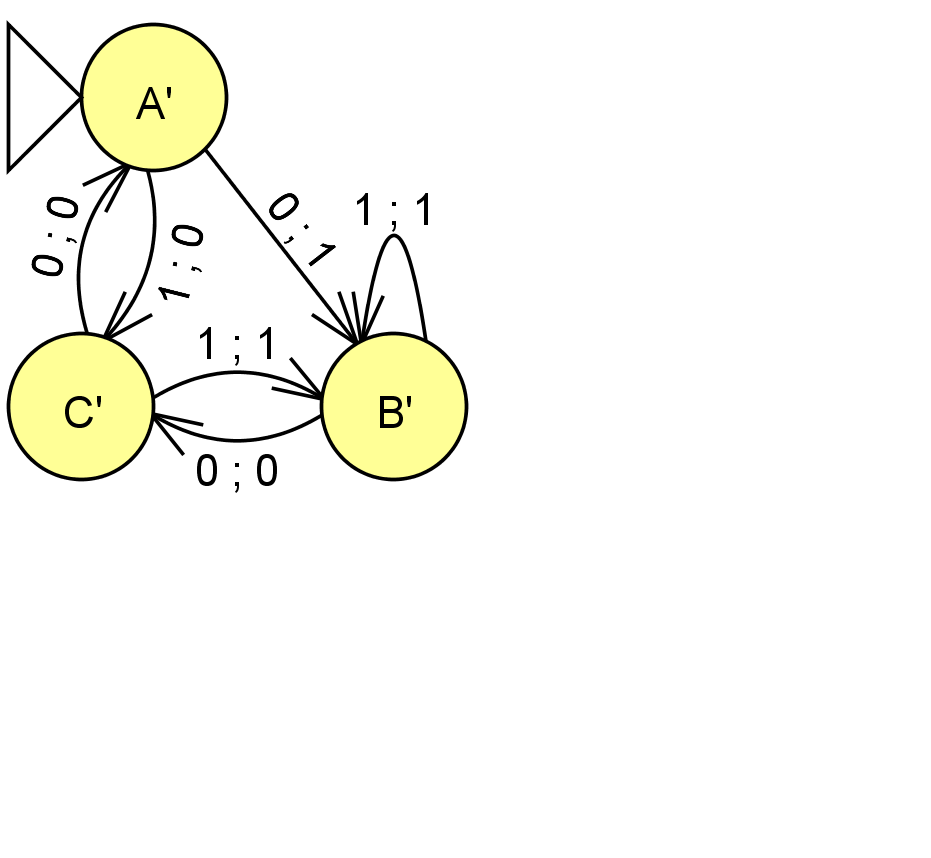
1. **[1pt]** What is/are the final state/s (if there are any)? Answer: M’ is a mealy machine. Why is there a need to find the final states?

|  |  |  |
| --- | --- | --- |
| λ’: | 0 | 1 |
| A' | 0 | 1 |
| B' | 0 | 1 |
| C' | 0 | 1 |

|  |  |  |
| --- | --- | --- |
| δ’: | 0 | 1 |
| A’ | B’ | C’ |
| B’ | C’ | B’ |
| C’ | A’ | C’ |

M’ = { Q’, Σ, Δ, δ’, λ’, A’}, Q’ = {A’,B’,C’}, Σ = {0,1}, Δ= {0,1}

M’:



\*\* End of MCO1-1 \*\*

## References:

Gribkoff, E. (2013, May 29). *Finite State Machine Applications*. University of California - Davis.

Retrieved on October 19, 2023 from

<https://www.cs.ucdavis.edu/~rogaway/classes/120/spring13/eric-applications>

Moore, K., Chumbley, A., Ho, N., & Khim, J. (n.d.). *Regular Languages*. Brilliant. Retrieved on October 20, 2023 from

[https://brilliant.org/wiki/regular-languages/#:~:text=A%20regular%20language%20is%20a,alphabet](https://brilliant.org/wiki/regular-languages/#%3A~%3Atext%3DA%20regular%20language%20is%20a%2Calphabet%2C%20or%20set%20of%20symbols)

[%2C%20or%20set%20of%20symbols.](https://brilliant.org/wiki/regular-languages/#%3A~%3Atext%3DA%20regular%20language%20is%20a%2Calphabet%2C%20or%20set%20of%20symbols)

Geeksforgeeks. (2019, May 1). *Regular Expressions vs Context Free Grammar*. Retrieved on October 19, 2023 from

[https://www.geeksforgeeks.org/regular-expression-vs-context-free-grammar/#:~:text=Regular%20E](https://www.geeksforgeeks.org/regular-expression-vs-context-free-grammar/#%3A~%3Atext%3DRegular%20Expressions%20are%20most%20useful%2Cbe%20define%20by%20Regular%20Expression) [xpressions%20are%20most%20useful,be%20define%20by%20Regular%20Expression](https://www.geeksforgeeks.org/regular-expression-vs-context-free-grammar/#%3A~%3Atext%3DRegular%20Expressions%20are%20most%20useful%2Cbe%20define%20by%20Regular%20Expression).

Priya, B. (2021, June 11). *What is the pumping lemma for regular language?*. TutorialsPoint.

Retrieved on October 20, 2023 from

<https://www.tutorialspoint.com/what-is-the-pumping-lemma-for-regular-language>