

# CPSC-406 Report

Marc Nathaniel Domingo  
Chapman University

April 27, 2023

## Abstract

The purpose of this course is to understand how the Theory of Algorithms coincide with the application of functions and algorithms in Software Engineering. In particular, this course places an emphasis on the concepts of Automata, Logic, and Concurrency in regard to algorithms in our eventual application of these topics to a final project.

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Week 1 . . . . .	1
1.2	Week 2 . . . . .	2
1.3	Week 3 . . . . .	2
1.4	Week 4 . . . . .	2
1.5	Week 5 . . . . .	2
1.6	Week 6 . . . . .	2
1.7	Week 7 . . . . .	2
1.8	Week 8 . . . . .	2
1.9	Week 9 . . . . .	2
1.10	Week 10 . . . . .	2
1.11	Week 11 . . . . .	2
1.12	Week 12 . . . . .	2
<b>2</b>	<b>Homework</b>	<b>3</b>
2.1	Week 2 . . . . .	3
2.2	Week 3 . . . . .	3
2.3	Week 6 . . . . .	6
2.4	Week 9 . . . . .	7
2.5	Week 11 . . . . .	9
<b>3</b>	<b>Paper</b>	<b>13</b>
<b>4</b>	<b>Conclusions</b>	<b>13</b>

## 1 Introduction

### 1.1 Week 1

During this week, the overall outline and "roadmap" of topics and project deadlines were covered during class, as well as an introduction to the concept of Definitive Finite Automata (DFA).

## **1.2 Week 2**

Lectures covered Definitive Finite Automata (DFA) and Non-Definitive Finite Automata (NFA) in more detail, as well as how to convert NFA to DFA.

## **1.3 Week 3**

During this week, the lectures covered the concepts of querying a database, as well as algorithms for Unification and Resolution.

## **1.4 Week 4**

During this week, lectures covered the concept of Distributed Hash Tables (DHT) in the application of Decentralized Systems.

## **1.5 Week 5**

During this week, the lectures covered the concepts of Turing Machines and their relation to Polynomial and Non-Polynomial Time Problems.

## **1.6 Week 6**

During this week, the lectures reviewed Satisfiability as covered during our lectures on Non-Polynomial Time Problems in their application to a Sudoku Solver.

## **1.7 Week 7**

During this week, the lectures covered the concept of Distributed Systems in terms of how to create a Logical Clock.

## **1.8 Week 8**

During this week, the lectures covered the concept of Big O-complexity in terms of certain sorting algorithms such as bubble sort, in addition to an introduction to Complexity Theory.

## **1.9 Week 9**

During this week, the lectures covered the concept of model checking in addition to how the Needham-Schroeder Key Protocol can be modeled through Spin to witness how messages can be intercepted.

## **1.10 Week 10**

During this week, the lectures covered the concept of Temporal Logic and how it can be applied to solve Satisfiability problems.

## **1.11 Week 11**

During this week, the lectures covered the concepts of Composing Automata in addition to writing Regular Expressions based off of Non-Definitive Automata (NFA).

## **1.12 Week 12**

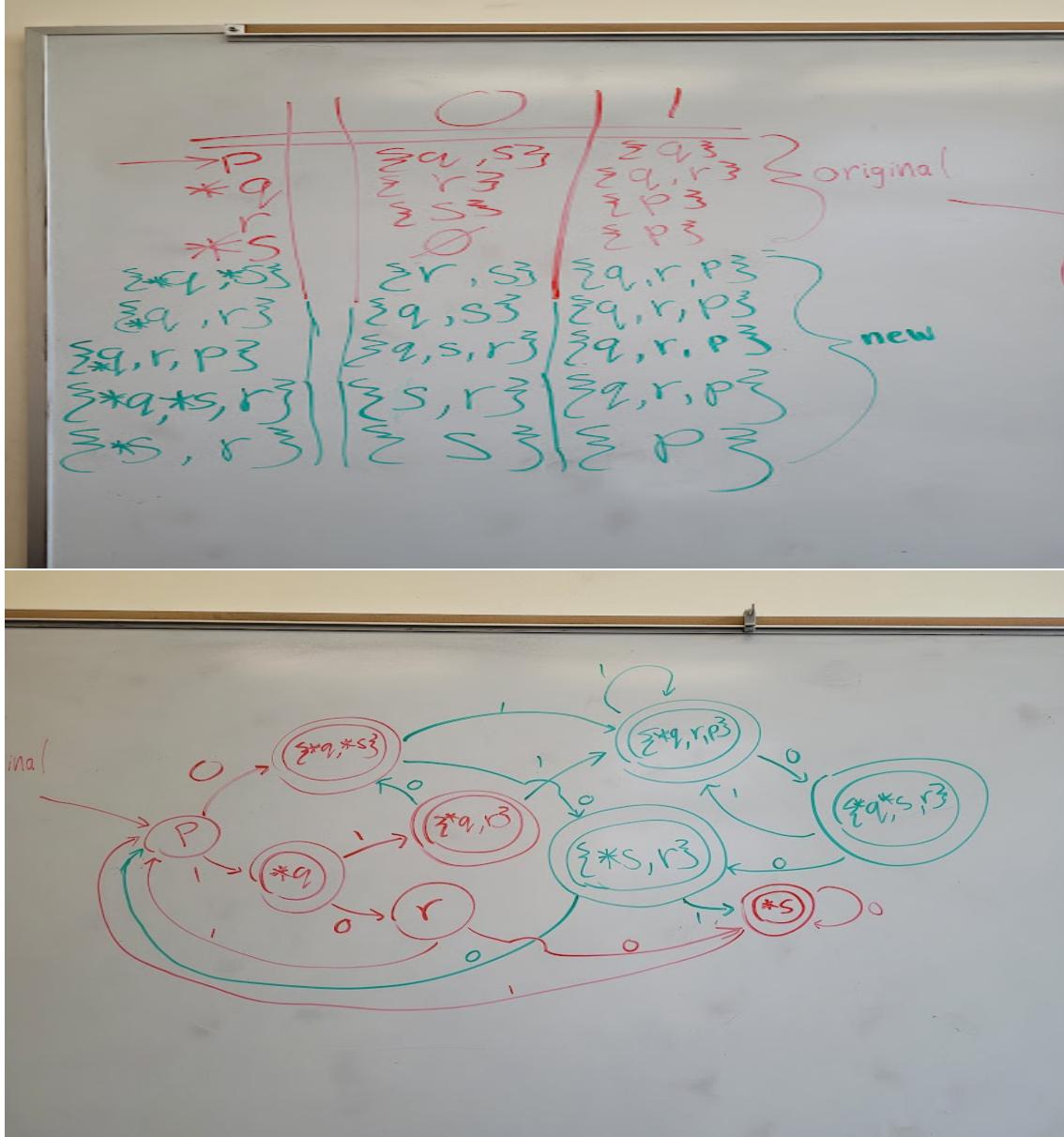
During this week, the lectures covered the concepts of

## 2 Homework

This section contains solutions to homework.

### 2.1 Week 2

The objective of this week's homework is to convert a given NFA to a DFA by hand.



### 2.2 Week 3

The objective of this week's homework is to solve Unification Algorithm problems in addition to practicing drawing an SLD-tree solution to a program given a particular input.

$$1. f(x, f(x, y)) \stackrel{?}{=} f(f(y, a), f(v, b))$$

$$1. X \stackrel{?}{=} f(y, a)$$

$$\sigma_1 = [f(y, a) / x]$$

$$2. f(x, y) \stackrel{?}{=} f(v, b)$$

$$3. X \stackrel{?}{=} v$$

$$X_{\sigma_1} \stackrel{?}{=} v \rightarrow f(y, a) \stackrel{?}{=} v$$

$$4. Y \stackrel{?}{=} b$$

$$\sigma_4 = [b / y]$$

$$\sigma_2 = \sigma_3 \circ \sigma_4 = [f(y, a) / v, b / y]$$

$$\sigma_1 = \sigma_1 \circ \sigma_2 = [f(y, a) / x, f(y, a) / v, b / y]$$

$$MGU \sigma = [f(b, a) / x, f(b, a) / v, b / y]$$

$$2. f(g(v), f(x, y)) \stackrel{?}{=} f(x, f(y, v))$$

$$1. g(v) \stackrel{?}{=} x$$

$$\sigma_1 = [g(v) / x]$$

$$2. f(x, y) \stackrel{?}{=} f(y, v)$$

$$3. X \stackrel{?}{=} y$$

$$X_{\sigma_1} \stackrel{?}{=} y \rightarrow g(v) \stackrel{?}{=} y$$

$$\sigma_3 = [g(v) / y]$$

$$4. Y \stackrel{?}{=} v$$

$$Y_{\sigma_3} \stackrel{?}{=} v \rightarrow g(v) \stackrel{?}{=} v$$

Fail check occurs

$$3. h(u, f(g(v), w), g(w)) \stackrel{?}{=} h(f(x, b), v, z)$$

$$1. v \stackrel{?}{=} f(x, b)$$

$$\sigma_1 = [f(x, b) / v]$$

$$2. f(g(v), w) \stackrel{?}{=} u$$

$$f(g(v), w) \stackrel{?}{=} u_{\sigma_1} \rightarrow f(g(v), w) \stackrel{?}{=} f(x, b)$$

$$3. g(v) \stackrel{?}{=} x \rightarrow \sigma_2 = [g(v) / x]$$

$$4. w \stackrel{?}{=} b \rightarrow \sigma_4 = [b / w]$$

$$\sigma_2 = \sigma_3 \circ \sigma_4 = [g(v) / x, b / w]$$

$$5. g(w) \stackrel{?}{=} z \rightarrow \sigma_5 = [g(w) / z]$$

$$MGU \sigma = \sigma_1 \circ \sigma_2 \circ \sigma_4 = [f(x, b) / v, g(v) / x, b / w, g(w) / z]$$

$$MGU \sigma = [f(g(v), b) / v, g(v) / x, b / w, g(w) / z]$$

▷	$\% \text{addr}(x,y) :- X \text{ holds the address of } Y$						
▷	$\% \text{serv}(X) :- X \text{ is an address server}$						
▷	$\% \text{conn}(X,Y) :- X \text{ can initiate a connection to } Y$						
▷	$\% \text{knowing}(X,Y) :- \text{either end can initiate a connection}$						
▷	$\text{addr}(a,d)$						
▷	$\text{addr}(a,b).$						
▷	$\text{addr}(b,c).$						
▷	$\text{addr}(c,a).$						
▷	$\text{serv}(b).$						
▷							
▷	$? - \text{addr}(W,a), \text{conn}(a,W).$						
▷							
▷	$? - \text{addr}(Z_1,a)$						
▷							
▷	$  Z_1 = a   Z_2 = a   Z_3 = b   Z_4 = c$	$Z_2 = a, Z_1 = d$	$Z_2 = a, Z_1 = b$	$Z_2 = b, Z_1 = c$	$Z_2 = c, Z_1 = d$		
▷	$? - \text{addr}(a,a) ? - \text{addr}(a,a) ? - \text{addr}(a,b) ? - \text{addr}(c,a)$	$? - \text{addr}(a,b) ? - \text{serv}(b) ? - \text{addr}(b,a)$	$? - \text{addr}(c,a) ? - \text{serv}(a) ? - \text{addr}(a,a)$	$? - \text{addr}(c,a) ? - \text{serv}(a) ? - \text{addr}(a,a)$			
▷	fail	fail	fail	fail	fail	fail	fail
▷							
▷							
▷	$? - \text{addr}(a,d) ? - \text{serv}(d) ? - \text{addr}(d,a)$	$? - \text{addr}(b,c) ? - \text{serv}(c) ? - \text{addr}(c,a)$					
▷							
▷							
▷							

## 2.3 Week 6

The objective of this week's homework was to create Indirect Truth Tables to understand Satisfiability in terms of Propositional Logic. For the following tables, we used the Indirect Truth Table method to prove why the formulas above the tables are valid.

		$p \vee \neg p$		$R \vee Q \leftarrow (R \vee P) \wedge (Q \vee P)$									
		$\square P \quad \neg P$		$P \vee \neg P$		0		0		0		0	
		0	1	0		0		0		0		0	
		$\square (P \rightarrow Q) \rightarrow (\neg Q \rightarrow \neg P)$		0		0		0		0		0	
		$\square P \quad q$		$P \rightarrow q$		$\neg q$		$\neg P$		$(\neg q \rightarrow \neg P)$		$(P \rightarrow q) \wedge (\neg q \rightarrow \neg P)$	
		0	0	1		1		1		1		1	
		0	1	1		0		1		1		1	
		1	0	0		1		0		0		1	
		1	1	1		0		1		1		1	

		$P \rightarrow (Q \rightarrow P)$		$(P \rightarrow Q) \vee (Q \rightarrow P)$									
		$\square P \quad q$		$P \rightarrow (q \rightarrow P)$		$P \rightarrow q$		$q \rightarrow P$		$(P \rightarrow q) \vee (q \rightarrow P)$			
		0	0	1		1		1		1		1	
		0	1	1		1		0		0		1	
		t	0	1		1		0		1		1	
		1	1	1		1		1		1		1	

		$((P \rightarrow Q) \rightarrow P) \rightarrow P$		$(P \rightarrow Q) \rightarrow P \leftarrow (P \rightarrow Q) \wedge (Q \rightarrow P)$									
		$\square P \quad q$		$P \rightarrow q$		$(P \rightarrow q) \rightarrow P$		$((P \rightarrow q) \rightarrow P) \rightarrow P$		$(P \rightarrow Q) \rightarrow P$		$(P \rightarrow Q) \wedge (Q \rightarrow P)$	
		0	0	1		0		1		0		0	
		0	1	1		0		1		1		0	
		1	0	0		1		1		1		1	
		1	1	1		1		1		1		1	

$$(P \vee Q) \wedge (\neg P \vee R) \rightarrow Q \vee R$$

$P$	$Q$	$R$	$P \vee Q$	$\neg P$	$\neg P \vee R$	$(P \vee Q) \wedge (\neg P \vee R)$	$Q \vee R$	$(P \vee Q) \wedge (Q \vee R)$	$\neg (P \vee Q) \rightarrow Q \vee R$
0	0	0	0	1	1	0	0	0	1
0	0	1	0	1	1	0	1	1	0
0	1	0	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1
1	0	0	1	0	0	0	1	0	1
1	0	1	1	0	1	0	1	1	1
1	1	0	1	0	0	0	1	1	1
1	1	1	1	0	1	1	1	1	1

For the following tables, the Indirect Truth Table method was used to prove why the formulas above the table are not valid.

$$(P \vee Q) \rightarrow (P \wedge Q)$$

$P$	$Q$	$P \vee Q$	$P \wedge Q$	$(P \vee Q) \rightarrow (P \wedge Q)$
0	0	0	0	1
0	1	1	0	0
1	0	1	0	0
1	1	1	1	0

Invalid

$$(P \rightarrow Q) \rightarrow (\neg P \rightarrow \neg Q)$$

$P$	$Q$	$P \rightarrow Q$	$\neg P \rightarrow Q$	$(P \rightarrow Q) \rightarrow (\neg P \rightarrow \neg Q)$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	1	1	1

Invalid

## 2.4 Week 9

The objective of this week's homework is to explore exercises relating to the Needham-Schroeder Public-Key Protocol using Spin models.

One such example is to go through the program **ns.pml** and define what the meaning behind the propositions *success*, *AliceBob*, and *BobAlice*.

- **success** is a propositional variable that is true if and only if the first protocol (Alice) has successfully sent a reply to the second protocol (Bob), and if the second protocol (Bob) had sent a reply to the first protocol's (Alice) initial message.

- **aliceBob** is a propositional variable that is true if and only if the partner of the first protocol, Alice, is Bob.
- **bobAlice** is a propositional variable that is true if and only if the partner of the second protocol, Bob, is Alice.

The next exercise required determining which of the two formulas at the bottom of the code would violate the protocol when executed, as well as to explain the nature of the protocol based on that violation.

- The violation was caused by

$$ltl\{[](\text{success} \& \& \text{aliceBob} \rightarrow \text{aliceBob})\}$$

- As the only difference between the two lines is the placement of **aliceBob** and **bobAlice** are swapped, this implies that the protocol is hard-coded such that the protocol does not get violated if Alice replies to Bob's reply, but in the event that the roles are reversed, the protocol is susceptible to intruders.

The next exercise of the homework involved looking through the execution sequence output from the protocol violation.

- The execution sequence ended up being 83 lines long, which matched up with the protocol violation message, with an additional line representing the attempt at the next check.

The last exercise involved creating a Message Sequence Chart (MSC) that represents a successful attack on the protocol.

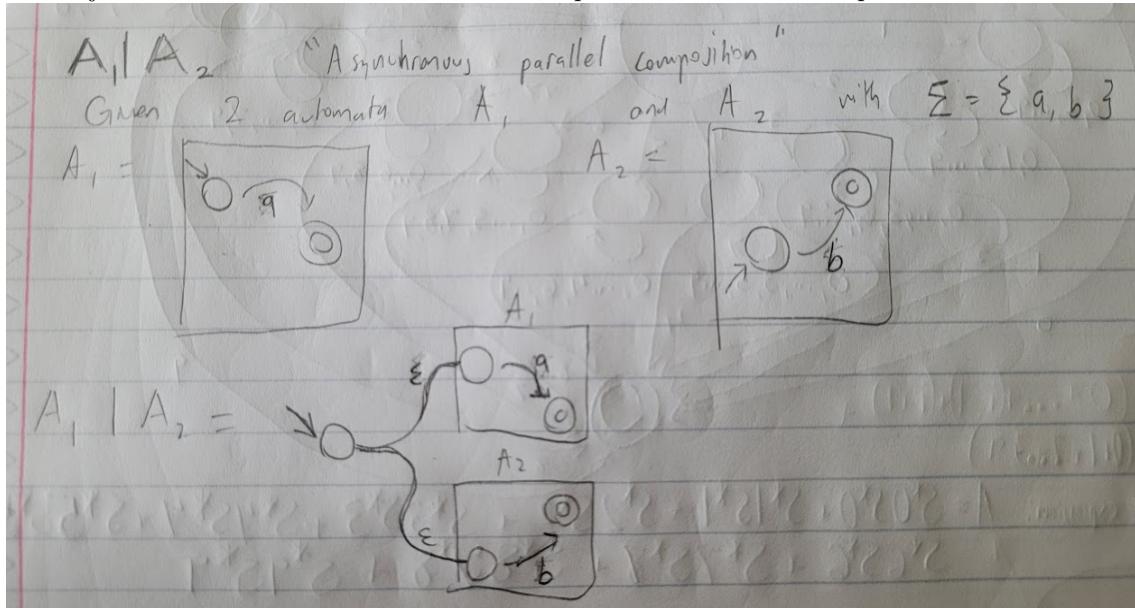
```

6: proc 0 (Alice:1) ns.pml:77 (state 12) [network!msg1,partnerA,data.key,data.d1,data.d2]
7: proc 2 (Intruder:1) ns.pml:144 (state 1) [network?msg_,_,data.key,data.d1,data.d2]
23: proc 2 (Intruder:1) ns.pml:201 (state 64)
[network!msg,rcpt,data.key,data.d1,data.d2]
24: proc 1 (Bob:1) ns.pml:104 (state 1) [network?msg1,bob,data.key,data.d1,data.d2]
34: proc 1 (Bob:1) ns.pml:123 (state 17)
[network!msg2,partnerB,data.key,data.d1,data.d2]
35: proc 2 (Intruder:1) ns.pml:144 (state 1) [network?msg_,_,data.key,data.d1,data.d2]
45: proc 2 (Intruder:1) ns.pml:180 (state 37)
[network!msg,rcpt,intercepted.key,intercepted.d1,intercepted.d2]
46: proc 0 (Alice:1) ns.pml:80 (state 13) [network?msg2,alice,data.key,data.d1,data.d2]
52: proc 0 (Alice:1) ns.pml:93 (state 20)
[network!msg3,partnerA,data.key,data.d1,data.d2]
53: proc 2 (Intruder:1) ns.pml:144 (state 1) [network?msg_,_,data.key,data.d1,data.d2]
69: proc 2 (Intruder:1) ns.pml:201 (state 64)
[network!msg,rcpt,data.key,data.d1,data.d2]
70: proc 1 (Bob:1) ns.pml:126 (state 18) [network?msg3,bob,data.key,data.d1,data.d2]
```

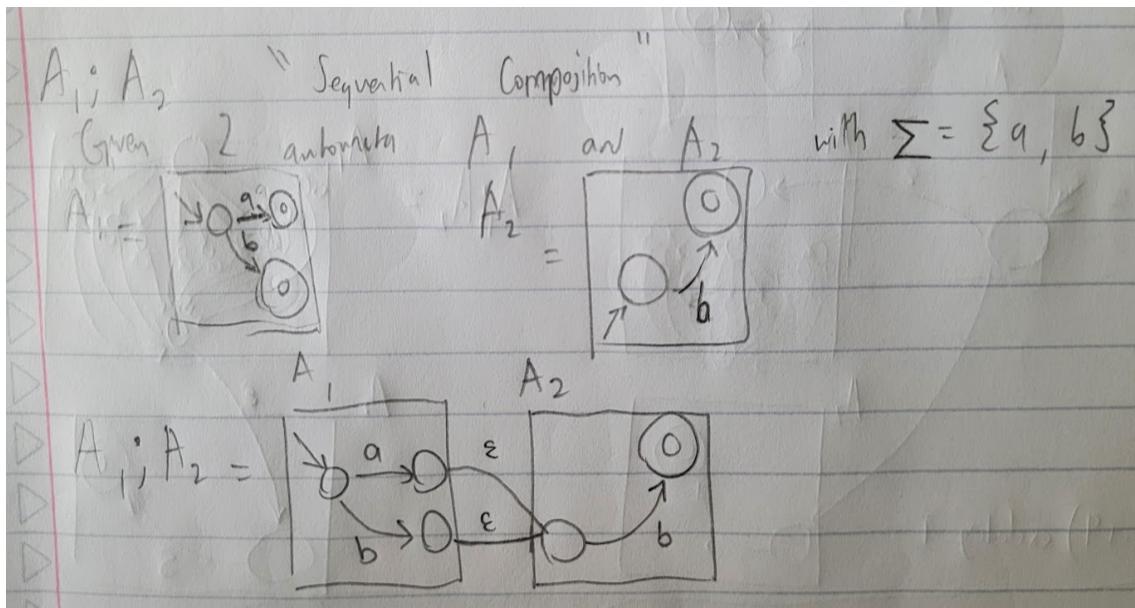
- The MSC represents a successful attack on the protocol as the intruder has managed to message both Alice and Bob before the other is able to receive and reply to the message from a proper protocol.

## 2.5 Week 11

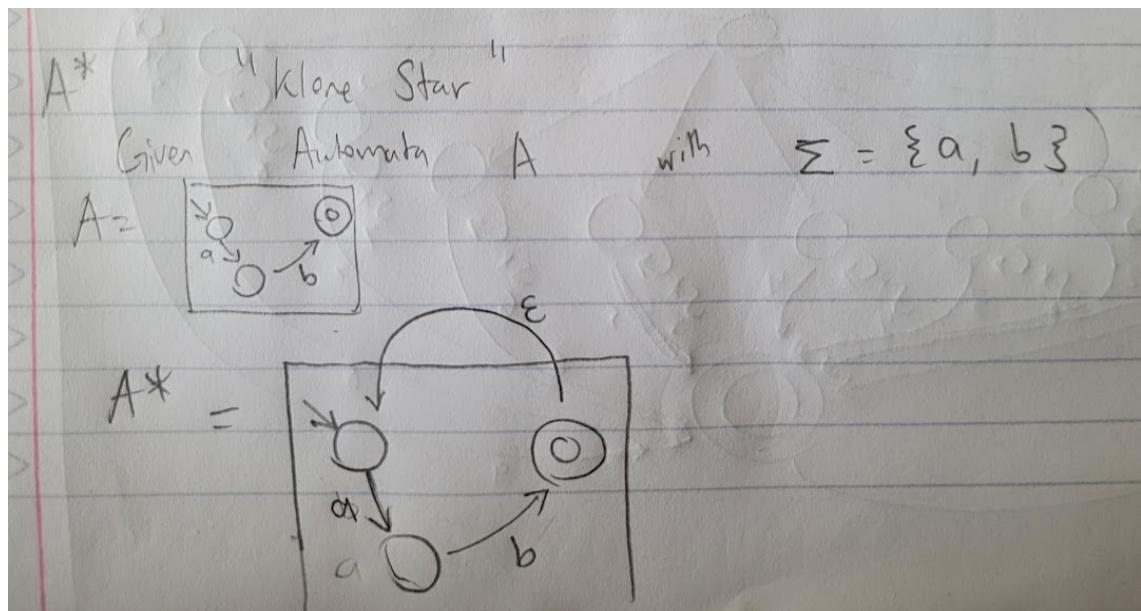
The objective of this week's homework was to explore  $\epsilon$ -transitions and operations on various automata.



- Asynchronous Parallel Composition accepts languages that are capable of operating in parallel to each other, utilizing a set of strings that consists of the combined alphabet that each language accepts.
- The above example would have a regular expression of  $A_1 | A_2 = \epsilon a + \epsilon b$



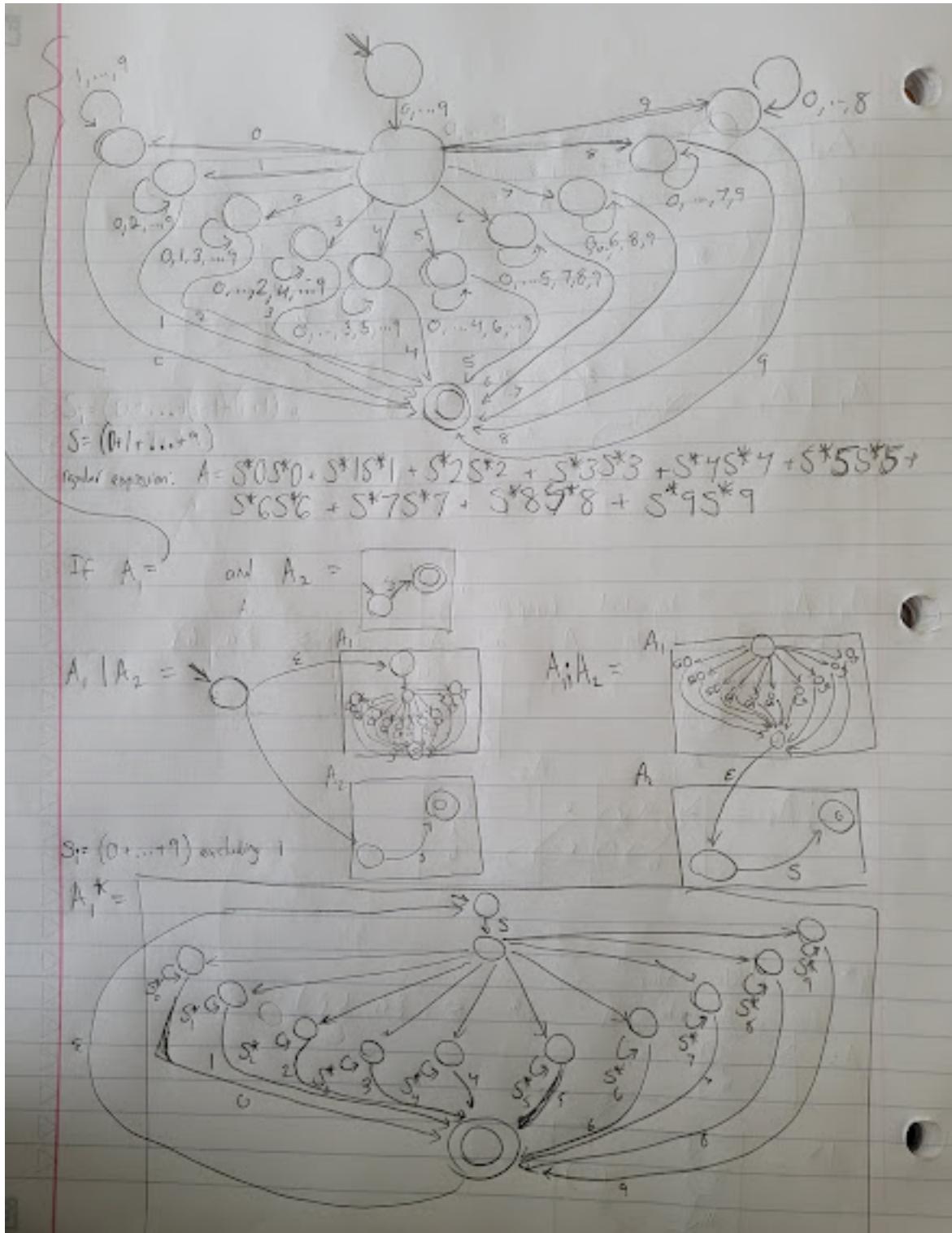
- Sequential Composition accepts languages that are capable of running in a sequential sequence, that is the output of one language is used as the starting point of another language, utilizing a set of strings that both languages share.
- The above example would have a regular expression of  $A_1 ; A_2 = (a + b)\epsilon b$



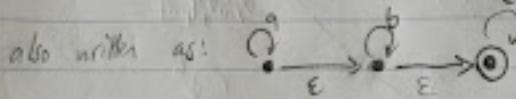
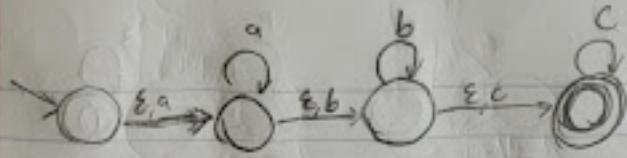
- Kleene Star accepts languages that are capable of infinitely repeating, starting back at the machine's initial state.
- The above example would have a regular expression of  $A^* = abc^*$

The next section of the homework consisted of finding the DFA and regular expressions of examples from the textbook, as well as finding the  $\epsilon$ -NFA if the above operations were applied to the examples.

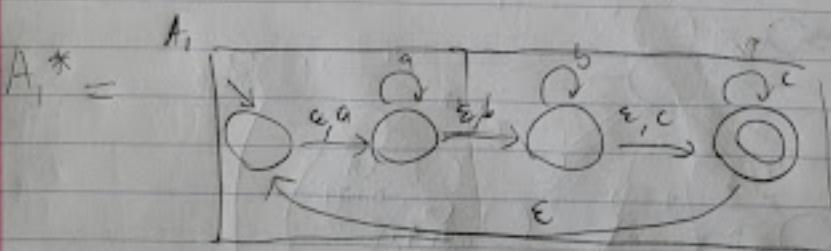
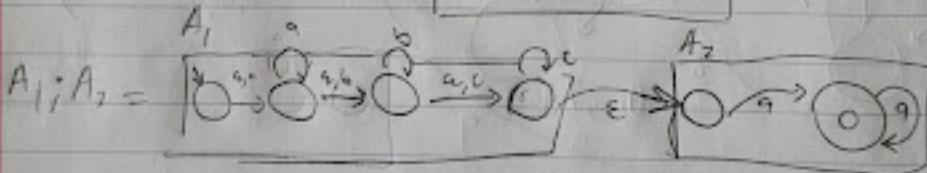
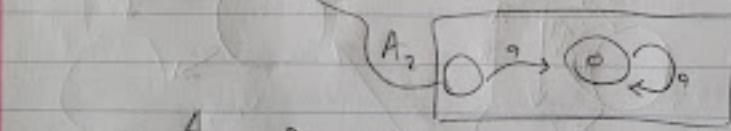
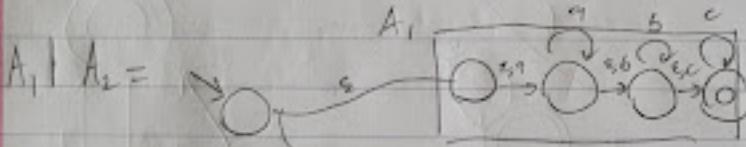
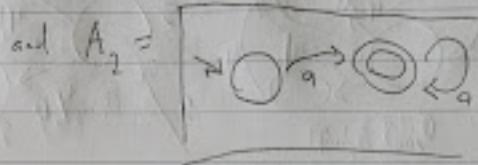
- The set of strings over the alphabet  $\{0, 1, \dots, 9\}$  such that the digit has appeared before.



- The set of strings consisting of zero or more a's followed by zero or more b's, followed by zero or more c's.



regular expression:  $a^* b^* c^*$



## **3 Paper**

...

## **4 Conclusions**

(approx 400 words) A critical reflection on the content of the course. Step back from the technical details. How does the course fit into the wider world of software engineering? What did you find most interesting or useful? What improvements would you suggest?

## **References**

[ALG] [Algorithm Analysis](#), Chapman University, 2023.