Fundamentals of Computer Science

Midterm Coursework Assignment

Cindy Chen
GOLDSMITHS, UNIVERSITY OF LONDON

Table of Contents

Question 1

Without using the truth table show the following statements are true. Explain your reasoning.

a) $(\boldsymbol{p} \to \boldsymbol{q}) \vee (\boldsymbol{p} \to \boldsymbol{r}) \equiv \neg \boldsymbol{r} \to (\neg \boldsymbol{p} \vee \boldsymbol{q})$ **[3 marks]**

b) $(\boldsymbol{p} \to \neg \boldsymbol{q}) \wedge \neg \boldsymbol{p} \equiv \boldsymbol{p} \to (\neg \boldsymbol{q} \wedge \neg \boldsymbol{p})$ **[3 marks]**

c) $\neg \boldsymbol{p} \to (\boldsymbol{q} \to \boldsymbol{r}) \equiv \boldsymbol{q} \to (\boldsymbol{p} \vee \boldsymbol{r})$ **[3 marks]**

a) $(\boldsymbol{p} \to \boldsymbol{q}) \vee (\boldsymbol{p} \to \boldsymbol{r}) \equiv \neg \boldsymbol{r} \to (\neg \boldsymbol{p} \vee \boldsymbol{q})$ **[3 marks]**

According to page 15 of the essential reading being Koshy, Thomas. *Discrete Mathematics with Applications*. (Academic Press, 2004), the order of precedence for logical operators is:

1) $\neg$;
2) $\wedge$;
3) $\vee$;
4) $\to$; and
5) $\leftrightarrow$.

Page 15 also notes that notwithstanding the above order of precedence, parenthesized subexpressions are always evaluated first.

According to the Implication Conversion Law as outlined on page 22 of the essential reading being Koshy, Thomas. *Discrete Mathematics with Applications*. (Academic Press, 2004) which states that for any propositions p, q and r:

$$p \to q \equiv \neg p \vee q$$

For $(p \to q) \vee (p \to r)$ as given in a):

$$(p \to q) \vee (p \to r) \equiv (\neg p \vee q) \vee (\neg p \vee r)$$

According to the Associative Laws as outlined on page 22 of the essential reading being Koshy, Thomas. *Discrete Mathematics with Applications*. (Academic Press, 2004) which states that for any propositions p, q and r:

$$p \vee (q \vee r) \equiv (p \vee q) \vee r$$

$$p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$$

$(p \to q) \vee (p \to r) \equiv (\neg p \vee q) \vee (\neg p \vee r)$ as worked out above can be further converted to:

$$(p \to q) \vee (p \to r) \equiv (\neg p \vee q) \vee (\neg p \vee r) \equiv \neg p \vee q \vee \neg p \vee r$$

According to the Communicative Laws as outlined on page 22 of the essential reading being Koshy, Thomas. *Discrete Mathematics with Applications*. (Academic Press, 2004) which states that for any propositions p, q and r:

$$p \vee q \equiv q \vee p$$

$$p \wedge q \equiv q \wedge p$$

For $(p \to q) \vee (p \to r) \equiv \cdots \equiv \neg p \vee q \vee \neg p \vee r$ as worked out above there is:

$$(p \to q) \vee (p \to r) \equiv \cdots \equiv \neg p \vee q \vee \neg p \vee r \equiv r \vee \neg p \vee \neg p \vee q$$

According to the Idempotent Laws as outlined on page 21 of the essential reading being Koshy, Thomas. *Discrete Mathematics with Applications*. (Academic Press, 2004) which states that for any propositions p, q and r:

$$p \vee p \equiv p$$

$$p \wedge p \equiv p$$

What is worked out above being $(p \to q) \vee (p \to r) \equiv \cdots \equiv r \vee \neg p \vee \neg p \vee q$ can be further converted into:

$$(p \to q) \vee (p \to r) \equiv \cdots \equiv r \vee \neg p \vee \neg p \vee q \equiv r \vee \neg p \vee q$$

Using again the Implication Conversion Law being $p \to q \equiv \neg p \vee q$ and one (1) of the two (2) Associative Laws being $p \vee (q \vee r) \equiv (p \vee q) \vee r$ as referenced above gives:

$$(p \to q) \vee (p \to r) \equiv \cdots \equiv r \vee \neg p \vee q \equiv r \vee (\neg p \vee q) \equiv \neg r \to (\neg p \vee q)$$

Therefore, it has now been shown that $(\boldsymbol{p \to q}) \vee (\boldsymbol{p \to r}) \equiv \neg \boldsymbol{r} \to (\neg \boldsymbol{p} \vee \boldsymbol{q})$ is true.

b) $(\boldsymbol{p} \to \neg \boldsymbol{q}) \wedge \neg \boldsymbol{p} \equiv \boldsymbol{p} \to (\neg \boldsymbol{q} \wedge \neg \boldsymbol{p})$ **[3 marks]**

According to page 15 of the essential reading being Koshy, Thomas. *Discrete Mathematics with Applications*. (Academic Press, 2004), the order of precedence for logical operators is:

1) $\neg$;

2) $\wedge$;

3) $\vee$;

4) $\rightarrow$; and

5) $\leftrightarrow$.

Page 15 also notes that notwithstanding the above order of precedence, parenthesized subexpressions are always evaluated first.

According to the Implication Conversion Law as outlined on page 22 of the essential reading being Koshy, Thomas. *Discrete Mathematics with Applications*. (Academic Press, 2004) which states that for any propositions p, q and r:

$$p \rightarrow q \equiv \neg p \vee q$$

For $(p \rightarrow \neg q) \wedge \neg p$ as given in b) there is:

$$(p \rightarrow \neg q) \wedge \neg p \equiv (\neg p \vee \neg q) \wedge \neg p$$

According to the Distributive Law as outlined on page 22 of the essential reading being Koshy, Thomas. *Discrete Mathematics with Applications*. (Academic Press, 2004) which states that for any propositions p, q and r:

$$p \wedge (q \vee r) \equiv (p \wedge q) \vee (q \wedge r)$$

$$p \vee (q \wedge r) \equiv (p \vee q) \wedge (q \vee r)$$

For $(p \rightarrow \neg q) \wedge \neg p \equiv (\neg p \vee \neg q) \wedge \neg p$ as worked out above there is:

$$(p \rightarrow \neg q) \wedge \neg p \equiv (\neg p \vee \neg q) \wedge \neg p \equiv (\neg p \wedge \neg p) \vee (\neg q \wedge \neg p)$$

According to the Idempotent Laws as outlined on page 21 of the essential reading being Koshy, Thomas. *Discrete Mathematics with Applications*. (Academic Press, 2004) which states that for any propositions p, q and r:

$$p \vee p \equiv p$$

$$p \wedge p \equiv p$$

For $(p \rightarrow \neg q) \wedge \neg p \equiv \cdots \equiv (\neg p \wedge \neg p) \vee (\neg q \wedge \neg p)$ as worked out above there is:

$$(p \rightarrow \neg q) \wedge \neg p \equiv \cdots \equiv (\neg p \wedge \neg p) \vee (\neg q \wedge \neg p) \equiv \neg p \vee (\neg q \wedge \neg p)$$

Using again the Implication Conversion Law being $p \rightarrow q \equiv \neg p \vee q$ on the above gives:

$$(p \rightarrow \neg q) \wedge \neg p \equiv \cdots \equiv \neg p \vee (\neg q \wedge \neg p) \equiv p \rightarrow (\neg q \wedge \neg p)$$

Therefore, it has now been shown that $(p \rightarrow \neg q) \wedge \neg p \equiv p \rightarrow (\neg q \wedge \neg p)$ is true.

c) $\neg \boldsymbol{p} \rightarrow (\boldsymbol{q} \rightarrow \boldsymbol{r}) \equiv \boldsymbol{q} \rightarrow (\boldsymbol{p} \vee \boldsymbol{r})$ **[3 marks]**

According to page 15 of the essential reading being Koshy, Thomas. *Discrete Mathematics with Applications*. (Academic Press, 2004), the order of precedence for logical operators is:

1) $\neg$;
2) $\wedge$;
3) $\vee$;
4) $\rightarrow$; and
5) $\leftrightarrow$.

Page 15 also notes that notwithstanding the above order of precedence, parenthesized subexpressions are always evaluated first.

According to the Implication Conversion Law as outlined on page 22 of the essential reading being Koshy, Thomas. *Discrete Mathematics with Applications*. (Academic Press, 2004) which states that for any propositions p, q and r:

$$p \rightarrow q \equiv \neg p \vee q$$

For $\neg p \rightarrow (q \rightarrow r)$ as given in c) there is:

$$\neg p \rightarrow (q \rightarrow r) \equiv \neg p \rightarrow (\neg q \vee r) \equiv p \vee (\neg q \vee r)$$

According to the Associative Laws as outlined on page 22 of the essential reading being Koshy, Thomas. *Discrete Mathematics with Applications*. (Academic Press, 2004) which states that for any propositions p, q and r:

$$p \vee (q \vee r) \equiv (p \vee q) \vee r$$

$$p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$$

For $\neg p \rightarrow (q \rightarrow r) \equiv \cdots \equiv \neg p \vee (\neg q \vee r)$ as worked out above there is:

$$\neg p \rightarrow (q \rightarrow r) \equiv \cdots \equiv p \vee (\neg q \vee r) \equiv p \vee \neg q \vee r$$

According to the Communicative Laws as outlined on page 22 of the essential reading being Koshy, Thomas. *Discrete Mathematics with Applications*. (Academic Press, 2004) which states that for any propositions p, q and r:

$$p \vee q \equiv q \vee p$$

$$p \wedge q \equiv q \wedge p$$

For $\neg p \rightarrow (q \rightarrow r) \equiv \cdots \equiv p \vee \neg q \vee r$ as worked out above there is:

$$\neg p \rightarrow (q \rightarrow r) \equiv \cdots \equiv p \vee \neg q \vee r \equiv \neg q \vee p \vee r$$

Using again one (1) of the two (2) Associative Laws being $p \vee (q \vee r) \equiv (p \vee q) \vee r$ as referenced above gives:

$$\neg p \rightarrow (q \rightarrow r) \equiv \cdots \equiv \neg q \vee p \vee r \equiv \neg q \vee (p \vee r)$$

Using again the Implication Conversion Law being $p \rightarrow q \equiv \neg p \vee q$ on the above gives:

$$\neg p \rightarrow (q \rightarrow r) \equiv \cdots \equiv \neg q \vee (p \vee r) \equiv q \rightarrow (p \vee r)$$

Therefore, it has now been shown that $\neg \boldsymbol{p} \rightarrow (\boldsymbol{q} \rightarrow \boldsymbol{r}) \equiv \boldsymbol{q} \rightarrow (\boldsymbol{p} \vee \boldsymbol{r})$ is true.

Question 2

Prove the following statement by induction. For all nonnegative integers n, 3 divides $n^3 + 2n + 3$. State the mathematical induction and show your work clearly. [9 marks]

According to Lesson 2.3 Inductive proof; Video 2.301 Proof by Induction, the principle of mathematical induction is:

$$IF$$

$$P(0)\ is\ true$$

$$AND$$

$$\forall k \in \mathbb{N},\ P(k) \rightarrow P(k+1)$$

$$THEN$$

$$\forall n \in \mathbb{N},\ P(n)\ is\ true$$

Following from this principle, proving a statement true using mathematic induction is comprised of three steps:

I. 'The Basis': Prove $P(0)$ is true;
II. 'The Inductive Step': Prove if $P(k)$ then $P(k+1)$, where the assumption that $P(k)$ is true is called inductive hypothesis; and
III. 'Conclusion': Conclude, by induction, that $P(n)$ is true for all n.

To apply the above inductive reasoning steps to the given statement by reference to the class example covered in Lesson 2.3 Inductive proof; Video 2.303 Proof by Induction, let $P(n)$ be $P(n)$: 3 divides $n^3 + 2n + 3$ and the task is to prove that $P(n)$ is true for all non-negative integers n.

I. 'The Basis': Notice that $P(0)$: 3 divides $0^3 + 2 \times 0 + 3 = 3$ is true as every integer is divisible by itself;

II. 'The Inductive Step': Assume that if $P(k)$ is true (i.e. the inductive hypothesis), that is, 3 divides $k^3 + 2k + 3$

Notice that

$$(k+1)^3 + 2 \times (k+1) + 3$$

$$= (k+1)(k^2 + 2k + 1) + 2 \times (k+1) + 3$$

$$= (k^3 + k^2 + 2k^2 + 2k + k + 1) + (2k + 2) + 3$$

$$= k^3 + 3k^2 + 5k + 6$$

$$= (k^3 + 2k + 3) + 3k^2 + 3k + 3$$

$$= (k^3 + 2k + 3) + 3 \times (k^2 + k + 1)$$

Given the fact that it is the inductive hypothesis that 3 divides $k^3 + 2k + 3$, and the accompanying fact that 3 naturally divides $3 \times (k^2 + k + 1)$ because the latter is the multiplication of 3 to the integer being $k^2 + k + 1$ (which has to be an integer because k is an integer), 3 should also divide $(k+1)^3 + 2 \times (k+1) + 3 = (k^3 + 2k + 3) + 3 \times (k^2 + k + 1)$.

Therefore, it has now been proved that 3 divides $(k+1)^3 + 2 \times (k+1) + 3$ when it is assumed that 3 divides $k^3 + 2k + 3$, i.e. $P(k+1)$ is true when it is assumed that $P(k)$ is true.

III. 'Conclusion': Therefore, $P(n)$: 3 divides $n^3 + 2n + 3$ is true for all non-negative integers n. **The given statement being "For all nonnegative integers n, 3 divides $n^3 + 2n + 3$." has thus been proved by induction.**

Question 3

Students are required to create 6-character long passwords to access the library. The letters must be from lowercase letters or digits. Each password must contain at least two lowercase-letters, at least one digit and contains no repeated digits. How many valid passwords are there? You are required to show your work step-by-step. [9 marks]

Note: 1hfgt1 is invalid because 1 appears more than once. 134ggg is valid because there are 3 lower-case letters and digits are not repeated.

Notice that possible letter combinations containing at least two (2) lowercase letters, at least one (1) digit and no repeated digits can be divided into the below four (4) categories.

I. Possible letter combinations containing two (2) lowercase letters and four (4) non-repeating digits;
II. Possible letter combinations containing three (3) lowercase letters and three (3) non-repeating digits;
III. Possible letter combinations containing four (4) lowercase letters and two (2) non-repeating digits; and
IV. Possible letter combinations containing five (5) lowercase letters and one (1) non-repeating digit.

See below for detailed analyses of each of the above four (4) categories:

I. Possible letter combinations containing two (2) lowercase letters and four (4) non-repeating digits;

This means that two (2) slots out of six (6) have twenty-six (26) possible candidates being the twenty-six (26) lowercase letters whilst the remaining four (4) slots containing digits have ten (10), nine (9), eight (8) and seven (7) possible candidates being four (4) non-repeating digits selected from all ten (10) digits, which when expressed using permutations is $P(10,4) = \frac{10!}{(10-4)!} = 10 \times 9 \times 8 \times 7 = 5{,}040$ (Lesson 3.3 Order matters; Video: 3.304: Combinations). As the four (4) digits can be anywhere among the six (6) slots, the number of ways to select four

(4) out of six (6) calculated using combinations (instead of permutations as order does not matter, that is, it does not matter which slot is selected first) is $C(6,4) = \frac{6!}{4!(6-4)!} = 15$ (Lesson 3.3 Order matters; Video: 3.304: Combinations).

To summarise, the above discussion leads to:

$$26^2 \times P(10,4) \times C(6,4) = 26^2 \times 5{,}040 \times 15 = 51{,}105{,}600$$

II. Possible letter combinations containing three (3) lowercase letters and three (3) non-repeating digits;

This means that three (3) slots out of six (6) have twenty-six (26) possible candidates being the twenty-six (26) lowercase letters whilst the remaining three (3) slots containing digits have ten (10), nine (9) and eight (8) possible candidates being three (3) non-repeating digits selected from all ten (10) digits, which when expressed using permutations is $P(10,3) = \frac{10!}{(10-3)!} = 10 \times 9 \times 8 = 720$ (Lesson 3.3 Order matters; Video: 3.304: Combinations). Notice here that as the three (3) digits can be anywhere, the number of ways to select three (3) out of six (6) slots can be expressed using combinations (instead of permutations as order does not matter, that is, it does not matter which slot is selected first) per Lesson 3.3 Order matters; Video: 3.304: Combinations as $C(6,3) = \frac{6!}{3!(6-3)!} = 20$.

To summarise, the above gives rise to:

$$26^3 \times P(10,3) \times C(6,3) = 26^3 \times 720 \times 20 = 253{,}094{,}400$$

III. Possible letter combinations containing four (4) lowercase letters and two (2) non-repeating digits

This means that four (4) slots out of six (6) have twenty-six (26) possible candidates being the twenty-six (26) lowercase letters whilst the remaining two (2) slots containing digits have ten (10) and nine (9) possible candidates being two (2) non-repeating digits selected from all ten (10) digits, which when expressed using permutations is $P(10,2) = \frac{10!}{(10-2)!} = 10 \times 9 = 90$

(Lesson 3.3 Order matters; Video: 3.304: Combinations). As the two (2) digits can be anywhere among the six (6) slots, the number of ways to select two (2) out of six (6) calculated using combinations (instead of permutations as order does not matter, that is, it does not matter which slot is selected first) is $C(6,2) = \frac{6!}{2!(6-2)!} = 15$ (Lesson 3.3 Order matters; Video: 3.304: Combinations).

To summarise, the above gives rise to:

$$26^4 \times P(10,2) \times C(6,2) = 26^4 \times 90 \times 15 = 616{,}917{,}600$$

IV. Possible letter combinations containing five (5) lowercase letters and one (1) non-repeating digit

This means that five (5) slots out of six (6) have twenty-six (26) possible candidates being the twenty-six (26) lowercase letters whilst the last one (1) slot has (10) possible candidates being the ten (10) digits, which when expressed using permutations is $P(10,1) = \frac{10!}{(10-1)!} = 10$ (Lesson 3.3 Order matters; Video: 3.304: Combinations). As the digit slot can be anywhere among the six (6) slots, the number of ways to select one (1) out of six (6) calculated using combinations (instead of permutations as order does not matter when there is only slot being selected) is $C(6,1) = \frac{6!}{1!(6-1)!} = 6$ (Lesson 3.3 Order matters; Video: 3.304: Combinations).

To summarise, the above gives rise to:

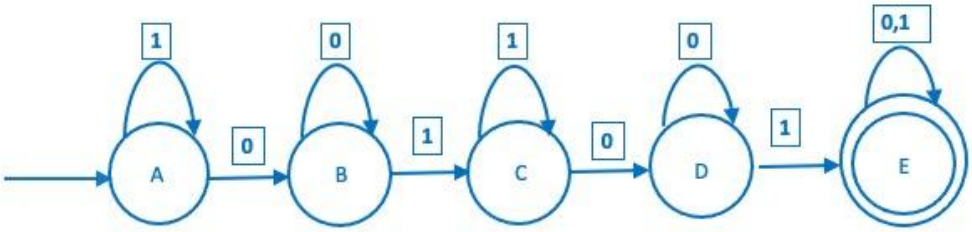
$$26^5 \times P(10,1) \times C(6,1) = 26^5 \times 10 \times 6 = 712{,}882{,}560$$

Having worked through all four (4) categories above, the total number of possible letter combinations containing at least two (2) lowercase letters, at least one (1) digit and no repeated digits are: $\mathbf{51,105,600 + 253,094,400 + 616,917,600 + 712,882,560 = 1,634,000,160.}$

As such, the total number of valid passwords out there is $\mathbf{1,634,000,160.}$

Question 4

Consider the following automaton.



a) **Give an example of a string containing 11 that is accepted by the following automaton. [2 marks]**

b) **Give an example of a string of length 8 that is rejected by the following automaton. [2 marks]**

c) **Describe the language of this automaton in plain English. [4 marks]**

d) **Describe the language of this automaton using Regular expression. [3 marks]**

a) **Give an example of a string containing 11 that is accepted by the following automaton. [2 marks]**

Example: 01101

Explanation: When the input string 01101 is fed into the above automaton, the processing proceeds as follows:

1. Start in state A.
2. Read 0, follow transition from A to B.
3. Read 1, follow transition from B to C.
4. Read 1, stay in C.
5. Read 0, follow transition from C to D.
6. Read 1, follow transition from D to E.
7. *Accept* because the automaton is in an accept state E at the end of the input.

b) Give an example of a string of length 8 that is rejected by the following automaton. [2 marks]

Example: 10000001

Explanation: When the input string 10000001 is fed into the above automaton, the processing proceeds as follows:

1. Start in state A.
2. Read 1, stay in A.
3. Read 0, follow transition from A to B.

4. Read 0, stay in B.
5. Read 0, stay in B.
6. Read 0, stay in B.
7. Read 0, stay in B.
8. Read 0, stay in B.
9. Read 0, stay in B.
10. Read 1, follow transition from B to C.
11. *Reject* because the automaton is in state C at the end of the input which is not an accept state.

c) Describe the language of this automaton in plain English. [4 marks]

Description: All binary strings containing two (2) 01s can be accepted by the above automaton, i.e. the language of this automation when described in plain English is all binary strings containing two (2) 01s, or:

$$L(M) = \{x | x\ contains\ two\ (2)\ 01s\}$$

Explanation: Following the method of backward computation as described in Lesson 4.2 Deterministic Automata; Video: 4.202 Language of the automata gives:

1. Observe incoming transitions to the accept state E. Note that 1 leads to state E but both 1 and 0 stay in state E. As such, it appears that input that will be accepted by state E has at least a 1.
2. Next observe incoming transitions to the penultimate state being state D. It appears that only 0 leads to and stays in state D. As such, it appears that input that will be accepted by state E has at least a 01.

3. Next observe incoming transitions to the antepenultimate (third-to-last) state being state C. It appears that only 1 leads to and stays in state C. As such, it appears that input that will be accepted by state E has at least a 1 and a 01.

4. Next observe incoming transitions to the preantepenultimate (fourth-to-last) state being state B. It appears that only 0 leads to and stays in state B. As such, it appears that input that will be accepted by state E has at least a 01 and another 01.

5. Finally check to ensure that no binary strings with two (2) 01s ends at a non-accept state. There is indeed none.

   As such, the above automaton accepts all binary strings containing two (2) 01s. That is, the language of this automation when described in plain English is all binary strings containing two (2) 01s. To use the mathematic notations per Lesson 4.2 Deterministic Automata; Video: 4.202 Language of the automata and Lesson 5.1 Formal Languages; Video: 5.101 Regular expressions, the language of the above automaton being $L(M)$ is:

$$L(M) = \{x | x\ contains\ two\ (2)\ 01s\}$$

d) Describe the language of this automaton using Regular expression. [3 marks]

Description:

$$L(M) = \{x | x \in \Sigma^* 01 \Sigma^* 01 \Sigma^*\}$$

Explanation:

It can be observed from the automation graph that its alphabet is $\Sigma = \{0,1\}$

By reference to the lecture example covered in Lesson 5.1 Formal Languages; Video: 5.103 Design regular expressions regarding “all binary words containing bb”, as per c), it has been found that the language of this automation is “all binary strings containing two (2) 01s”, meaning that there are at least two (2) occurrences of 01, and that there can be any

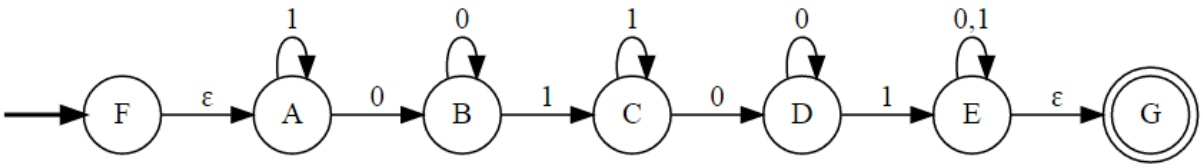
string composed from letters in the alphabet being $\Sigma = \{0,1\}$ including an empty string ε before, after and between these two (2) occurrences.

As the set of all strings composed from the alphabet $\Sigma = \{0,1\}$ is, including an empty string ε, is denoted by Σ^* (Lesson 5.1 Formal Languages; Video: 5.101 Regular expressions), the language of this automation can be described using regular expression as $\Sigma^*01\Sigma^*01\Sigma^*$. To use the mathematic notations per Lesson 4.2 Deterministic Automata; Video: 4.202 Language of the automata and Lesson 5.1 Formal Languages; Video: 5.101 Regular expressions, the language of the above automaton being $L(M)$ is:

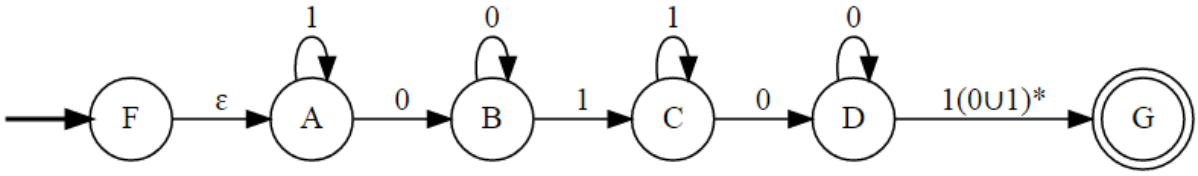
$$L(M) = \{x | x \in \Sigma^*01\Sigma^*01\Sigma^*\}$$

Note that the regular expression corresponding to the language of this automation can also be found by converting from the automaton direct. The process is as follows:

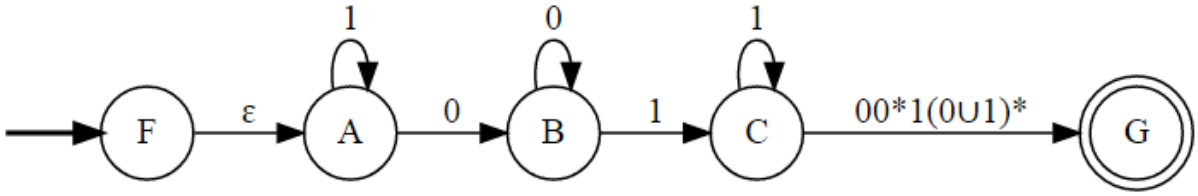
Firstly, create a new initial state F and a new accept state G



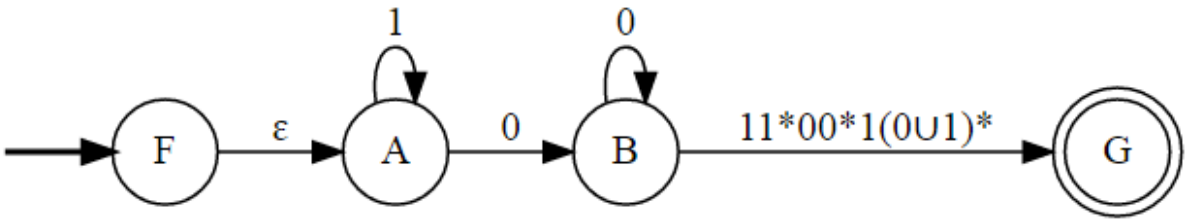
Secondly, remove E



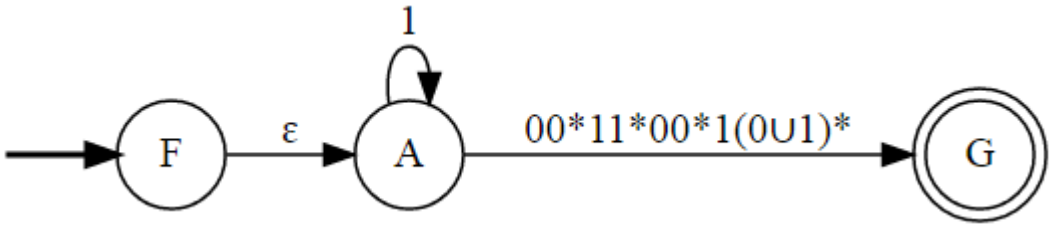
Thirdly, remove D



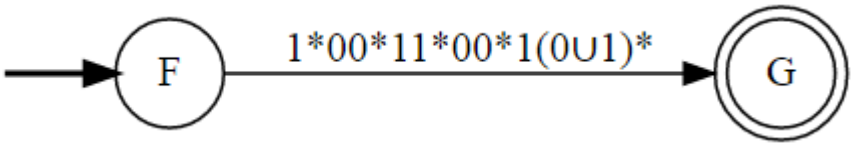
Next, remove C



Next, remove B



Removing A leads to



That is, the language of the automaton can be described in regular expression form as $L(M) = \{x | x \in 1^*00^*11^*00^*1(0 \cup 1)^*\} = \{x | x \in 1^*0^+1^+0^+1\Sigma^*\}$ which is practically equivalent to $L(M) = \{x | x \in \Sigma^*01\Sigma^*01\Sigma^*\}$ from above.

Question 5

Given $\boldsymbol{R = (0^*10^+)^+1^*}$ and $\boldsymbol{S = (1^*01^+)^*}$

a) **Give an example of a string that is neither in the language of R nor in S. [2 marks]**
b) **Give an example of a string that is in the language of S but not R. [2 marks]**
c) **Give an example of a string that is in the language of R but not S. [2 marks]**
d) **Give an example of a string that is in the language of *R* and *S*. [2 marks]**
e) **Design a regular expression that accepts the language of all binary strings with no occurrences of bab [4 marks]**

a) **Give an example of a string that is neither in the language of R nor in S. [2 marks]**

Example: 111111111

Explanation: The language represented by $R = (0^*10^+)^+1^*$ can be written out as $\{101, 1011, 0100101, 01001011, 00101, 01101, 101001, 10, 1000, 1000001, \dots\}$. Doing so amplifies the fact that any strings included in R has to have at least one (1) 0, which can also be read from the $(0^*10^+)^+$ part which denotes all **non-empty** strings formed with 0^*10^+ strings that are structured as any number of 0s or an empty string ε followed by 1 followed by any number of 0 BUT NOT an empty string ε, meaning that any 0^*10^+ string used to put together with each other and form $(0^*10^+)^+$ needs to end with 0, leading to at least one (1) 0 contained in any string in the language of R.

The language represented by $S = (1^*01^+)^*$ can be written out as $\{0101, 0111, 01110111, 010101, 01101, \dots\}$. Doing so amplifies the fact that any strings included in R has to have at least one (1) 0, which can also be read from $(1^*01^+)^*$ which denotes **all strings (can be an empty string ε which is achieve with nil of 1^*01^+)** formed with 1^*01^+ strings that are structured as any number of 1s or an empty string ε followed by 0 followed by any number

of 1s BUT NOT an empty string ε, meaning that any 1^*01^+ string used to put together with each other and form $(1^*01^+)^*$ except for the empty string ε needs to either start with 0 or have a 0 in the middle, leading to at least one (1) 0 contained in any string in the language of S.

As such, an example of a string that is in the language of neither R nor S can be one comprised entirely of 1s such as 111111111.

b) Give an example of a string that is in the language of S but not R. [2 marks]

Example: 011011

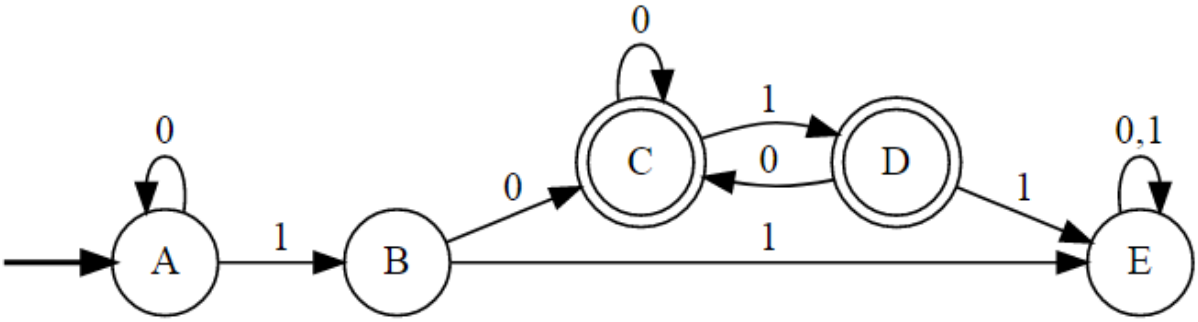
Explanation: As detailed in a) regarding $R = (0^*10^+)^+1^*$, any 0^*10^+ string used to put together with each other and form $(0^*10^+)^+$has to be structured as any number of 0s or an empty string ε followed by 1 followed by any number of 0 BUT NOT an empty string ε. This means that such strings need to either start with 1 or have a 1 in the middle. Notice however that such strings cannot start with 11 or have a 11 in the middle. Of course, as detailed above in a), such strings also have to always end with 0. However, the practical significance of this latter observation is diminished when one considers that there is a 1^* part that follows $(0^*10^+)^+$ in $R = (0^*10^+)^+1^*$, which can be either a string of 1s or ε - this part when put together with $(0^*10^+)^+$ means that a string in the language of R can end with anything – that is, either 0 or 1.

As detailed above also, any 1^*01^+ string used to put together with each other and form $S = (1^*01^+)^*$ except for the empty string ε has to be structured as any number of 1s or an empty string ε followed by 0 followed by any number of 1s BUT NOT an empty string ε. This implies that such strings need to either start with 0 or have a 0 in the middle. Notice however that such strings cannot start with 00 or have a 00 in the middle. Note the 1^+ part in $(1^*01^+)^*$, which can only be a string of 1s and CANNOT be an empty string ε - this part suggests that any string used to put together with each other and form $S = (1^*01^+)^*$ also has to end with 1.

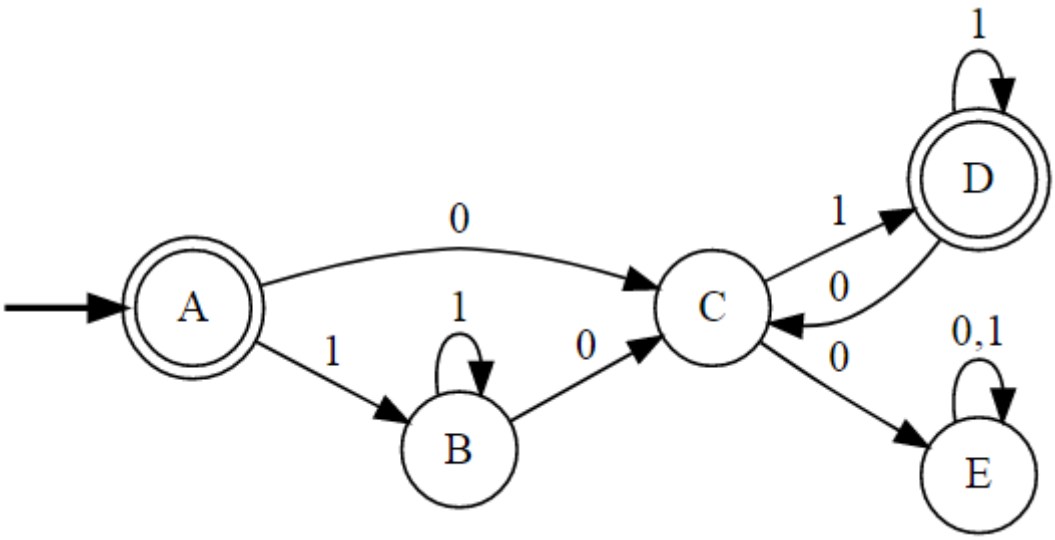
As such, an example of a string in the language of S but not R may take the form of a string that starts with 0 but not 00, ends with 1, has a 0 but not a 00 in the middle and either does not have a 1 in the middle or has a 11 in the middle, such as 011011.

Summarising a) and b), the DFAs for $R = (0^*10^+)^+1^*$ and $S = (1^*01^+)^*$ have been plotted below for reader reference:

R:



S:



c) Give an example of a string that is in the language of R but not S. [2 marks]

Example: 100100

Explanation: By reference to a) and b), an example of a string in the language of R but not S may take the form of a string that starts with 1 but not 11 and has a 1 but not a 11 in the middle which either does not have a 0 in the middle or has a 00 in the middle and does not end with 1 such as 100100.

d) Give an example of a string that is in the language of *R* and *S*. [2 marks]

Example: 101

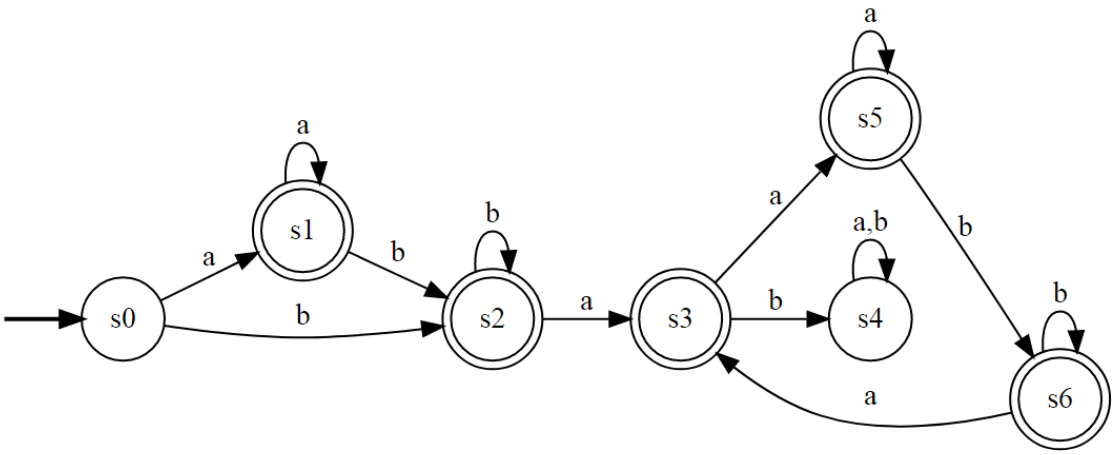
Example: By reference to a) and b), an example of a string that is in the languages of both R and S may take the form of a string that starts with 1, has a 0 in the middle, does not have a 11 or 00 anywhere in the string and ends with 1 such as 101.

e) Design a regular expression that accepts the language of all binary strings with no occurrences of bab [4 marks]

Answer: $a^*b^*(aa^+b^*)^*a^*$

Explanation: A binary string with no bab occurrence can start with any number of as, but the moment there starts to be bs, when bs end there has to be more than one (1) a before the string reverts back to b again, hence $a^*b^*(aa^+b^*)^*a^*$where a^*b^* at the start ensures that strings can start with either a or b, the $(aa^+b^*)^*$ part ensures that there are always at least two (2) as after any series of bs, and the a^* part in the end ensures that a string in the form of $a^*b^*a^*$ will not be rejected (i.e. a string will not be rejected if it turns from b to a but does not turn back to b again).

Converting $a^*b^*(aa^+b^*)^*a^*$ into a DFA gives:



This upon closer examination can be simplified into: