

# Assignment One

Abanob Tawfik  
z5075490

March 2019

## 1 Problem 1

Given two sets A and B, we define the operation  $*$  as follows:

$$A * B := (A \cap B)^c$$

Answer the following questions using the Laws of Set Operations (and any related results proven in lectures or tutorials) to justify your answer:

(a) What is  $(A * B) * (A * B)$ ? (5 marks)

Solution:

from using the supplied definition of the  $*$  operator

$$\begin{aligned}(A * B) * (A * B) &= (A \cap B)^c * (A \cap B)^c && \text{(Supplied definition of *)} \\ &= ((A \cap B)^c \cap (A \cap B)^c)^c && \begin{aligned} &\text{(Substituting } A * B) \\ &\text{(using definition of *)} \end{aligned}\end{aligned}$$

Let's say set  $C = (A \cap B)^c$

$$\begin{aligned}&= (C \cap C)^c && \text{(using our substitution } C = (A \cap B)^c) \\ &= C^c && \text{(using law of idempotence)} \\ &= ((A \cap B)^c)^c && \text{(substituting } C) \\ &= A \cap B && \text{(using double complementation)}\end{aligned}$$

So finally,  $(A * B) * (A * B) = A \cap B$

(b) Express  $A^c$  using only  $A$ ,  $*$  and parentheses (if necessary). (5 marks)

Solution:

one way to express  $A^c$  is  $(A \cup A)^c$  OR  $(A \cap A)^c$  (law of idempotence)  
 using this fact, i will make the claim  $A^c = A * A$

$$\begin{aligned} A * A &= (A \cap A)^c && \text{(Supplied definition of *)} \\ &= (A)^c && \text{(law of idempotence)} \\ &= A^c && \text{(removing parenthesis)} \end{aligned}$$

So finally,  $A * A = A^c$ , using only  $A$ ,  $*$  and parenthesis.

(c) Express  $A \cup B$  using only  $A$ ,  $B$ ,  $*$  and parenthesis (if necessary). (10 marks)

Solution:

One way to express  $A \cup B$  is  $(A^c \cap B^c)^c$  (de Morgan's Law and double complementation)  
 using this fact, i will make the claim  $A \cup B = (A * A) * (B * B)$

$$\begin{aligned} (A * A) * (B * B) &= ((A \cap A)^c \cap (B \cap B)^c) && \text{(Supplied definition of *)} \\ &= ((A)^c \cap (B)^c)^c && \text{(law of idempotence)} \\ &= (A^c \cap B^c)^c && \text{(removing parenthesis)} \\ &= (A^c)^c \cup (B^c)^c && \text{(de Morgan's Law)} \\ &= A \cup B && \text{(using double complementation)} \end{aligned}$$

So finally,  $(A * A) * (B * B) = A \cup B$ , using only  $A$ ,  $B$ ,  $*$  and parenthesis.

## 2 Problem 2

A binary tree is a data structure where each node is linked to at most two successor nodes:

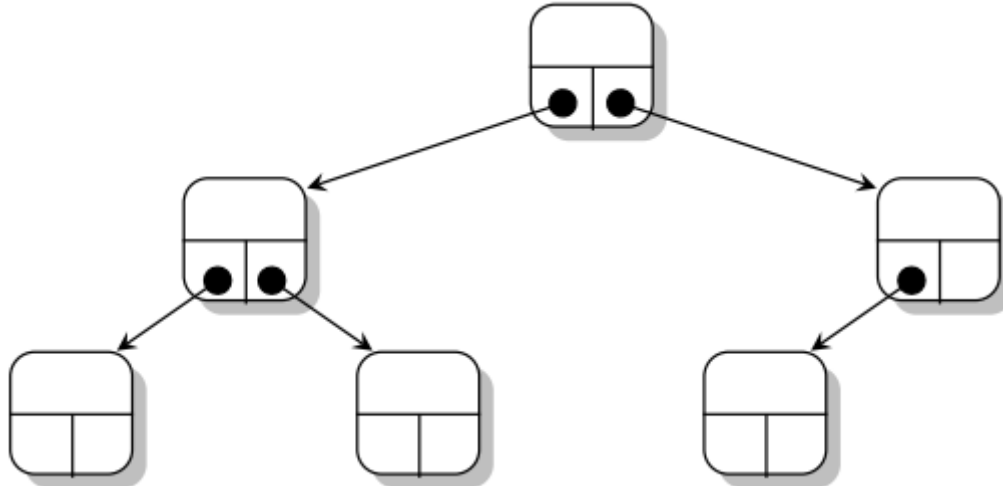


Figure 1: A visual representation of a binary tree

If we allow empty binary trees (trees with no nodes), then we can simplify the possibilities of zero, one, or two successors by saying a node has exactly two children which are binary trees

(a) Give a recursive definition of the binary tree data structure. Your definition may be concrete (i.e. code) or abstract, as long as it is clear what the base and recursive cases are. (4 marks)

Solution in java:

```
public class Tree<data>
{
    private data D;
    private Tree Root;
    private Tree Left;
    private Tree Right;
}
```

For this code, a base case would be the following: `Tree == null`. This means every aspect of the tree is null and there is no root, non existant tree.

The recursive case would be to pass the following attributes: Left and Right, which are also trees.

Leaves occur when Left and Right are both null and root is NOT null.

The Root node is there to distinguish between a tree that has no nodes, in comparison to an empty tree which has a parent node (leaves).

This allows for a data structure that can call upon itself to expand through the tree recursively.

(b) Based on your recursive definition above, define (in code or mathematically) the function `leaves(T)` that counts the number of leaves in a binary tree `T`. (4 marks)

Solution in java:

establishing the tree data type which will be used in our function `leaves(T)`

```
public class Tree<data>
{
    private data D;
    private Tree Root;
    private Tree Left;
    private Tree Right;
}
```

now we will create our `leaves(T)` function which will return the number of leaves in our tree recursively.

```
public int leaves(Tree T)
{
    // base case 1 for our recursion
    // if tree is empty we return 0
    if(T == NULL)
        return 0;

    // base case 2 for our recursion
    // if both children are empty trees return 1
    if(T.Left == NULL && T.Right == NULL)
        return 1;

    // Recursive Process
    // pass through both children which are also trees
    return leaves(T.Left) + leaves(T.Right);
}
```

(c) Based on your recursive definition above, define (in code or mathematically) the function `internal(T)` that counts the number of fully-internal nodes in a binary tree `T`. Hint: it is acceptable to define an empty tree as having -1 fully-internal nodes. (4 marks)

Solution in java:

establishing the tree data type which will be used in our function `internal(T)`

```
public class Tree<data>
{
    private data D;
    private Tree Root;
    private Tree Left;
    private Tree Right;
}
```

now we will create our `internal(T)` function which will return the number of fully-internal nodes in our tree recursively. Note, we have to perform this in reverse order we search from the leaves up rather than root down.

```
public int internal(Tree T)
{
    // base case 1 for our recursion
    // tree has no nodes at all
    // 0 parents/children nodes (no root)
    if(T.root == NULL)
        return -1;

    // base case 2 for our recursion
    // if tree is empty we return 0
    if(T == NULL)
        return 0;

    // base case 3 for our recursion
    // if both children are non-empty trees add 1 to the count
    int count = 0;
    if(T.Left != NULL && T.Right != NULL)
        count++;

    // Recursive Process
    // pass through both children which are also trees
    // unwind counter after whole tree processed
    return count += internal(T.Left) + internal(T.Right);
}
```

(d) If  $T$  is a binary tree, let  $P(T)$  be the proposition that  $\text{leaves}(T) = 1 + \text{internal}(T)$ . Prove that  $P(T)$  holds for all binary trees  $T$ . (8 marks)

Solution:

To perform this proof we can use structural induction on the Tree structure. Let  $S$  be the set of all Trees. A Tree can be defined with the following,

Base: Empty tree ( $\text{root} == \text{null}$ ).

else: a tree that contains 2 trees as it's children and potentially data.

First we need to prove it for the minimal object in the set of trees,  $S$ , the empty tree.

Let us use the symbol  $\tau$  to denote an empty tree leaves.

Base Case 1:  $P(\tau)$

show  $\text{leaves}(\tau) = 1 + \text{internal}(\tau)$

We know  $\text{leaves}(\tau) = 0$ , as it will return from the first base case from 2(b).

$0 = 1 + \text{internal}(\tau)$  (base case 2(b))

$\text{internal}(\tau) = -1$ , as the root specifically is null in an empty tree, this is the first base case from 2(c).

$0 = 1 + (-1)$  (base case 2(c))

Base Case 2: let  $T$  be a tree with a single node, where  $T \in S$

show  $\text{leaves}(T) = 1 + \text{internal}(T)$

We know  $\text{leaves}(T) = 1$  as a tree with one node is a leaf itself since it has no children

$1 = 1 + \text{internal}(T)$  (base case 2(b))

$\text{internal}(T) = 0$ , as a tree consisting of only one node cannot be an internal node since it has no children

$1 = 1 + 0$  (base case 2(c))

Hence our base cases holds for the minimal objects in our set  $S$ .

Inductive Hypothesis: Let us assume that for every tree  $T \in S$  with  $n$  nodes, where  $n > 1$ ,  $P(T)$  holds, so that the number of leaves  $l$ , is equal to the number of internal nodes  $i$  increased by 1, in other words  $l = i + 1$ .

Inductive Case: prove  $P(T')$  holds where  $T' \in S$  is an extension or reverse extension of a tree  $T \in S$ .

Assume  $T$  is a tree with  $l$  leaves, and  $i$  internal nodes.

Let us inspect the deepest possible fully-internal node ' $N$ ' of  $T$ . By definition of being the deepest possible fully internal node, the number of leaves from the sub-trees of ' $N$ ' is 2.

We can create a new tree  $T'$  from  $T$  by removing the children of ' $N$ '. ' $N$ ' no longer is a fully-internal node and is now a leaf.

By doing this our new tree  $T'$  has one less fully-internal node and two less leaves, however the fully-internal node that was removed has now become a leaf since we removed it's sub tree.

Our tree  $T'$  now has  $(l - 2 + 1)$  leaves (removed 2 sub-trees, internal node became leaf)

Our induction Hypothesis states that the number of leaves is always 1 more than the number of internal nodes, so

$T'$  also has  $(1 - 2 + 1) - 1$  internal nodes (Induction Hypothesis)  
 $\text{leaves}(T') = 1 - 2 + 1$  (number of leaves in  $T'$ )

Now we have to prove the proposition  $P(T')$ :  $\text{leaves}(T') = \text{internal}(T') + 1$ .

$1 - 2 + 1 = \text{internal}(T') + 1$  (Induction Hypothesis)

$\text{internal}(T') = (1 - 2) + 1 - 1$  (substitution from Induction Hypothesis)

By substituting  $\text{internal}(T')$  and  $\text{leaves}(T')$  we can prove our proposition.

$(1 - 2 + 1) = (1 - 2 + 1) - 1 + 1$  (substitution)

$1 - 1 = 1 - 1$  (arithmetic)

This expression equates to true, so  $P(T')$  holds.

We have proven both base cases for an empty tree, and a tree with a single node, which is also the base of our recursive structure.

We have also made the hypothesis that for some tree  $T \in S$ ,  $P(T)$  holds and have proven on the extension of our recursive structure that  $P(T')$  holds where the tree  $T' \in S$ ,  $P(T) \rightarrow P(T')$ .

Therefore by the principle of mathematical induction, since  $P$  holds for the base cases, and  $P(T) \rightarrow P(T')$ ,  $P$  is true for any tree in the set of all trees  $S$ .

### 3 Problem 3

Our WiFi networks, Alpha, Bravo, Charlie and Delta, all exist within close proximity to one another as shown below.



Figure 2: An example of a WiFi network

Networks connected with an edge in the diagram above can interfere with each other. To avoid interference networks can operate on one of two channels, hi and lo. Networks operating on different channels will not interfere; and neither will networks that are not connected with an edge.

Our goal is to determine (algorithmically) whether there is an assignment of channels to networks so that there is no interference. To do this we will transform the problem into a problem of determining if a propositional formula can be satisfied.

- (a) Carefully defining the propositional variables you are using, write propositional formulas for each of the following requirements: (12 marks)

Let  $A_L$  be the propositional variable that A uses channel lo  
 Let  $A_H$  be the propositional variable that A uses channel hi  
 Let  $B_L$  be the propositional variable that B uses channel lo  
 Let  $B_H$  be the propositional variable that B uses channel hi  
 Let  $C_L$  be the propositional variable that C uses channel lo  
 Let  $C_H$  be the propositional variable that C uses channel hi  
 Let  $D_L$  be the propositional variable that D uses channel lo  
 Let  $D_H$  be the propositional variable that D uses channel hi

- (i)  $\psi_1$ : Alpha uses channel hi or channel lo; and so does Bravo, Charlie and Delta. (4 marks)

Solution:

$$\psi_1 = (A_L \vee A_H) \wedge (B_L \vee B_H) \wedge (C_L \vee C_H) \wedge (D_L \vee D_H)$$

- (ii)  $\psi_2$ : Alpha does not use both channel hi and lo; and the same for Bravo, Charlie and Delta. (4 marks)

Solution:

$$\psi_2 = \neg(A_L \wedge A_H) \wedge \neg(B_L \wedge B_H) \wedge \neg(C_L \wedge C_H) \wedge \neg(D_L \wedge D_H)$$

- (iii)  $\psi_3$ : Alpha and Bravo do not use the same channel; and the same applies for all other pairs of networks connected with an edge. (4 marks)

Solution:

$$\psi_3 = (\neg(A_L \wedge B_L) \vee \neg(A_H \wedge B_H)) \wedge (\neg(B_L \wedge C_L) \vee \neg(B_H \wedge C_H)) \wedge (\neg(C_L \wedge D_L) \vee \neg(C_H \wedge D_H))$$



- (b) Note for this part of the question we will use the following truths for propositional logic:  
using the assignments T = true and F = false:

$$\begin{aligned}(T \vee F) &= T \\ (T \vee T) &= T \\ (F \vee F) &= F \\ (T \wedge F) &= F \\ (T \wedge T) &= T \\ (F \wedge F) &= F \\ \neg(T) &= F \\ \neg(F) &= T\end{aligned}$$

- (i) Show that  $\psi_1 \wedge \psi_2 \wedge \psi_3$  is satisfiable; so the requirements can all be met. Note that it is sufficient to give a satisfying truth assignment, you do not have to list all possible combinations. (4 marks)

Solution:

$\psi_1 \wedge \psi_2 \wedge \psi_3$  is satisfiable with the following truth assignment

$$\begin{aligned}A_L &= T \\ A_H &= F \\ B_L &= F \\ B_H &= T \\ C_L &= T \\ C_H &= F \\ D_L &= F \\ D_H &= T\end{aligned}$$

using our answers from (a)

$$\begin{aligned}\psi_1 &= (A_L \vee A_H) \wedge (B_L \vee B_H) \wedge (C_L \vee C_H) \wedge (D_L \vee D_H) \\ &= (T \vee F) \wedge (F \vee T) \wedge (T \vee F) \wedge (F \vee T) && \text{(substitution)} \\ &= T \wedge T \wedge T \wedge T && \text{(using rules of propositional logic above)} \\ &= (T \wedge T) \wedge (T \wedge T) && \text{(grouping left association)} \\ &= T \wedge T && \text{(using rules of propositional logic above)} \\ &= T && \text{(using rules of propositional logic above)}\end{aligned}$$

$$\begin{aligned}\psi_2 &= \neg(A_L \wedge A_H) \wedge \neg(B_L \wedge B_H) \wedge \neg(C_L \wedge C_H) \wedge \neg(D_L \wedge D_H) \\ &= \neg(T \wedge F) \wedge \neg(F \wedge T) \wedge \neg(T \wedge F) \wedge \neg(F \wedge T) && \text{(substitution)} \\ &= \neg(F) \wedge \neg(F) \wedge \neg(F) \wedge \neg(F) && \text{(using rules of propositional logic above)} \\ &= T \wedge T \wedge T \wedge T && \text{(using rules of propositional logic above)} \\ &= (T \wedge T) \wedge (T \wedge T) && \text{(grouping left association)} \\ &= T \wedge T && \text{(using rules of propositional logic above)} \\ &= T && \text{(using rules of propositional logic above)}\end{aligned}$$

$$\begin{aligned}\psi_3 &= (\neg(A_L \wedge B_L) \vee \neg(A_H \wedge B_H)) \wedge (\neg(B_L \wedge C_L) \vee \neg(B_H \wedge C_H)) \wedge (\neg(C_L \wedge D_L) \vee \neg(C_H \wedge D_H)) \\ &= (\neg(T \wedge F) \vee \neg(F \wedge T)) \wedge (\neg(F \wedge T) \vee \neg(T \wedge F)) \wedge (\neg(T \wedge F) \vee \neg(F \wedge T)) && \text{(substitution)} \\ &= (\neg(F) \vee \neg(F)) \wedge (\neg(F) \vee \neg(F)) \wedge (\neg(F) \vee \neg(F)) && \text{(using rules of propositional logic above)} \\ &= (T \vee T) \wedge (T \vee T) \wedge (T \vee T) && \text{(using rules of propositional logic above)} \\ &= T \wedge T \wedge T && \text{(using rules of propositional logic above)} \\ &= (T \wedge T) \wedge T && \text{(grouping left association)} \\ &= T \wedge T && \text{(using rules of propositional logic above)} \\ &= T && \text{(using rules of propositional logic above)}\end{aligned}$$

we have shown with these truth assignments that  $\psi_1 = \psi_2 = \psi_3 = T$

Hence to show  $\psi_1 \wedge \psi_2 \wedge \psi_3$  is satisfiable we need to show  $\psi_1 \wedge \psi_2 \wedge \psi_3 = T$

$$\begin{aligned}\psi_1 \wedge \psi_2 \wedge \psi_3 &= T \wedge T \wedge T \\ &= (T \wedge T) \wedge T && \text{(grouping left association)} \\ &= T \wedge T && \text{(using rules of propositional logic above)} \\ &= T && \text{(using rules of propositional logic above)}\end{aligned}$$

Therefore  $\psi_1 \wedge \psi_2 \wedge \psi_3$  is satisfiable as it equates to true with the following propositional variables ( $A_L = T, A_H = F, B_L = F, B_H = T, C_L = T, C_H = F, D_L = F, D_H = T$ ).

- (ii) Based on your answer to the previous question, which channels should each network use in order to avoid interference? (4 marks)

Solution:

a configuration in which no interference occurs when all the following conditions are met:

- a network is connected to either a channel lo or hi:  $\psi_1$
- a network is connected to only one channel:  $\psi_2$
- a connected network don't run on the same channel:  $\psi_3$

This can be simplified to find a configuration where the following holds,  $\psi_1 \wedge \psi_2 \wedge \psi_3 = T$ .

As seen from (i), this occurs when:

- $A_L = T$
- $A_H = F$
- $B_L = F$
- $B_H = T$
- $C_L = T$
- $C_H = F$
- $D_L = F$
- $D_H = T$

Therefore a configuration in which no interference occurs would be the following:

- A uses channel low ( $A_L = T$  and  $A_H = F$ )
- B uses channel high ( $B_L = F$  and  $B_H = T$ )
- C uses channel low ( $C_L = T$  and  $C_H = F$ )
- D uses channel high ( $D_L = F$  and  $D_H = T$ )

## 4 Problem 4

Prove the following arguments are valid using Natural Deduction:

(20 marks)

(a)  $A \vee (B \wedge C), A \rightarrow C \vdash C$

(10 marks)

Solution:

1. $A \rightarrow C$	
2. $A \vee (B \wedge C)$	
—	
3. $B \wedge C$	
—	
4. $B$	$\wedge$ <b>Elim:</b> 3
5. $C$	$\wedge$ <b>Elim:</b> 3
6. $A$	
—	
7. $C$	$\rightarrow$ <b>Elim:</b> 1, 6
8. $C$	$\vee$ <b>Elim:</b> 2, 3–5, 6–7

Explanation of the proof:

Our goal is to prove the first two antecedents 1 and 2 have the conclusion  $C$ . To do this first we need to breakdown each antecedent.

2 must be broken down by a logical or elimination. To do this we make an assumption on both sides of the  $\vee$  antecedent, and prove that they both yield the same conclusion.

We first assumed  $B \wedge C$ . By using and elimination we reduce to  $C$ , since  $B \wedge C$  is true, that means  $C$  is true.

Next we assume  $A$ . By using statement 1,  $A \rightarrow C$ , this is a modus ponens or otherwise known as  $\rightarrow$  elimination on 1 and 6 (if our antecedent  $A$  is true that implies the consequence  $C$  is also true). This means that  $C$  is true.

Since both sides of the  $\vee$  for 2, have the same consequence,  $C$  we can then say that under 1 and 2,  $C$  is the consequence by using  $\vee$  elimination.

Therefore  $A \vee (B \wedge C), A \rightarrow C \vdash C$ .

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

(10 marks)

Solution:

1. $\neg(P \rightarrow Q)$	
2. $\neg P$	
3. $P$	
4. $\perp$	$\perp$ <b>Intro:</b> 2, 3
5. $Q$	$\perp$ <b>Elim:</b> 4
6. $P \rightarrow Q$	$\rightarrow$ <b>Intro:</b> 3-5
7. $\perp$	$\perp$ <b>Intro:</b> 1, 6
9. $P$	$\neg$ <b>Intro:</b> 2-7

Explanation of the proof:

This proof is different to part (a) as we have to use RAA (reduction to absurdity) by first making a contradiction from an assumption allowing us to assume anything, for example, "If January has 1000 days, then the moon doesn't exist". Since our antecedent will always be false the consequence is true. By doing this we can perform an indirect proof.

An indirect proof works by first making an assumption, and then proving  $\perp$  from that assumption. If we can prove  $\perp$  from our assumption that means the negation of the assumption is true (proof by contradiction). For example if we assume A and the sub proof equates to  $\perp$ , then that means  $\neg A$  is true.

First we assume the opposite of what we want to prove ( $\neg P$ ), because if that proof leads to  $\perp$  that means the negation is true ( $P$ ).

Under the assumption of  $\neg P$  we can also assume  $P$ .

By assuming  $P$  and  $\neg P$  the second assumption equates to  $\perp$ .

Since we have  $\perp$ , we can use RAA to introduce  $Q$ , because since we have  $\perp$  we can assume anything else falls to true.

Since our assumption on line 3 leads to  $Q$  we can use modus ponens again, as our assumption  $P$  has led to the conclusion  $Q$  meaning from our first assumption on line 2 we have  $P \rightarrow Q$

$P \rightarrow Q$  is a direct contradiction to line 1 which was our given so that means our initial assumption falls to  $\perp$

Since our assumption  $\neg P$  is a contradiction and leads to  $\perp$ , that means the negation of our assumption is true,  $\neg\neg P$

$\neg\neg P = P$  from the indirect proof from 2-7

Therefore  $\neg(P \rightarrow Q) \vdash P$ .

The following images below were taken from the following natural deduction verification tool, <https://proofs.openlogicproject.org/>, which was used to validate the following natural deductions that was performed in (a) and (b). Note that the justification semantics are slightly different in (b) as IP is the same as negation introduction from the LaTeX package that was used, similarly for  $\perp$  and  $\top$  introduction/elimination. Below is the image showing the proof for part (a) was correct according to the verification tool.

1	$\neg(P \rightarrow Q)$	
2	$\neg P$	
3	$P$	
4	$\perp$	$\neg E$ 2, 3
5	$Q$	X 4
6	$P \rightarrow Q$	$\rightarrow I$ 3-5
7	$\perp$	$\neg E$ 1, 6
8	$P$	IP 2-7

NEW LINE

NEW SUBPROOF

Add a new line at end.


😊 Congratulations! This proof is correct.

Figure 3: Verification tool returning that my proof is correct for part (a)

Below is the image showing the proof for part (b) was correct according to the verification tool.

1		$A \rightarrow C$			
2		$A \vee (B \wedge C)$			
<hr/>					
3			$B \wedge C$		
4			$B$	$\wedge E\ 3$	
5			$C$	$\wedge E\ 3$	
6				$A$	
7				$C$	$\rightarrow E\ 1, 6$
8		$C$		$\vee E\ 2, 3-5, 6-7$	

 NEW LINE

 NEW SUBPROOF

😊 Congratulations! This proof is correct.

Figure 4: Verification tool returning that my proof is correct for part (b)