Problem 1:

**Recall the relation composition operator; deﬁned as:**

**R1; R2 = {(a, c) : there is a b with (a, b) ∈ R1 and (b, c) ∈ R2}**

**For any set S, and any binary relations R1, R2, R3 ⊆ S × S, prove or give a counterexample to disprove the following:**

**(a) (R1; R2); R3 = R1; (R2; R3)**

Proof:

To prove that (R1; R2); R3 = R1; (R2; R3) holds, we can prove statements below instead

(R1; R2); R3 ⊆ R1; (R2; R3)

R1; (R2; R3) ⊆ (R1; R2); R3

(1)

Assuming that there is an arbitrary element (a, b) ∈ (R1 ; R2) ; R3

Then there must be a m0 such that

(a, m0) ∈(R1 ; R2) and

(m0, b) ∈ R3

And since (a, m0) ∈(R1 ; R2), there must be a m1 such that

(a, m1) ∈ R1 and

(m1, m0) ∈ R2

For R2; R3, since (m1, m0) ∈ R2  and (m0, b) ∈ R3

So (m1, b) ∈ (R2;R3)

For R1; (R2; R3), since (a, m1) ∈ R1 and (m1, b) ∈ (R2;R3):

(a, b) ∈ R1 ; (R2 ; R3)

Thus for any element in (a, b) ∈ (R1; R2); R3 ,it must satisfy (a, b) ∈ R1; (R2; R3)

So (R1; R2); R3 ⊆ R1; (R2; R3).

(2)

for the versa case, for any element (a, b) ∈ R1 ; (R2 ; R3)

Then there must be a m2 such that

(a, m2) ∈R1 and

(m2, b) ∈ (R2 ; R3)

And since (m2, b) ∈ (R2 ; R3), there must be a m1 such that

(m2, m3) ∈ R2 and

(m3, b) ∈ R3

For R1; R2, since and (a, m2) ∈R1 and (m2, m3) ∈ R2

So (a, m3) ∈ (R1;R2)

For (R1; R2); R3, since (a, m3) ∈ (R1;R2) and (m3, b) ∈ R3

(a, b) ∈ (R1;R2); R3

Thus for any element in (a, b) ∈ R1; (R2; R3),it must satisfy (a, b) ∈ (R1; R2); R3

So R1; (R2; R3) ⊆ (R1; R2); R3

Since (R1; R2); R3 ⊆ R1; (R2; R3)

Thus R1; (R2; R3) = (R1; R2); R3

**(b) I; R1 = R1; I = R1 where I = {(x, x): x ∈ S}**

Proof:

According to the specification:

R1 ⊆ S × S

Since I = {(x, x): x ∈ S}

Then for every (r1, r2) ∈ R1, we can always find a (r1, r1) ∈ I such that:

{(r1, r1)}; {(r1, r2)} = {(r1, r2)}

So I; R1 = R1

Similarly, for every (r1, r2) ∈ R1, we can always find a (r2, r2) ∈ I such that:

Thus R1; I= R1

And I; R1 = R1; I = R1 holds

**(c) (R1; R2)← = R1; R2**

Giving a counterexample:

For R1 = R1**←** = {(1, 1), (2, 2)}

R2 = {(1, 3), (2, 3)}

R2**← =** {(3, 1), (3, 2)}

Then R1**←**; R2**←** = ∅

(R1; R2)**←** = {(3, 1), (3, 2)}

So (R1; R2)← = R1; R2 doesn’t hold.

**(d) (R1 ∪ R2) ; R3 = (R1 ; R3) ∪ (R2; R3)**

Proof:

To prove that (R1 ∪ R2) ; R3 = (R1 ; R3) ∪ (R2; R3) hold, we can prove statements below instead

(R1 ∪ R2) ; R3 ⊆ (R1 ; R3) ∪ (R2; R3) and

(R1 ; R3) ∪ (R2; R3)⊆ (R1 ∪ R2) ; R3

(1)

Assuming that there is an arbitrary element (a, b) ∈ (R1 **∪** R2) ; R3

Then there must be a m0 such that (a, m0) ∈ (R1 ∪ R2) and (m0, b) ∈ R3

Thus (a, m0) ∈ R1 or(a, m0) ∈ R2 holds

if (a, m0) ∈ R1

since (m0, b) ∈ R3

so (a, b) ∈ R1 ; R3

Since (R1 ; R3) ⊆ (R1 ; R3) ∪ (R2; R3)

then for any element (a, b) ∈ (R1 ∪ R2) ; R3

(a, b) ∈ (R1 ; R3) ∪ (R2; R3) holds

And vice versa, when (a, m0) ∈ R2 holds

(a, b) ∈ (R2; R3) holds

Then (a, b) ∈ (R1 ; R3) ∪ (R2; R3) holds

Thus (R1 ∪ R2) ; R3 ⊆ (R1 ; R3) ∪ (R2; R3)

(2)

for the versa case:

Assuming that there is an arbitrary element (a, b) ∈ (R1 ; R3) ∪ (R2; R3)

Then there must be a m0 such that

(a, m0) ∈ R1 and (m0, b) ∈ R3

or (a, m0) ∈ R2 and (m0, b) ∈ R3

if (a, m0) ∈ R1 and (m0, b) ∈ R3 holds, then (a, m0) ∈ R1 ∪ R2 holds

Thus (a, b) ∈ (R1 ∪ R2) ; R3

And vice versa, when (a, m0) ∈ R2 and (m0, b) ∈ R3

(a, b) ∈ (R1 ∪ R2) ; R3 holds

Then for any element (a, b) ∈ (R1 ; R3) ∪ (R2; R3)

(a, b) ∈ (R1 ∪ R2) ; R3 holds

Thus (R1 ; R3) ∪ (R2; R3) ⊆ (R1 ∪ R2) ; R3

since (R1 ∪ R2) ; R3 ⊆ (R1 ; R3) ∪ (R2; R3)

So (R1 ∪ R2) ; R3 = (R1 ; R3) ∪ (R2; R3)

**(e) R1; (R2 ∩ R3) = (R1; R2) ∩ (R1; R3)**

Giving a counterexample:

For R1 = {(1, 2), (2, 3)}

R2 = {(2, 2)}

R3= {(3, 5)}

Then for the left side of the equation:

R1; (R2 ∩ R3) = {(1, 2), (2, 3)}; ({(2, 2)} ∩ {(3, 5)})

= {(1, 2), (2, 3)}; ∅

= ∅

For the right side of the equation:

(R1 ; R2 ) ∩ (R1; R3) = ({(1, 2), (2, 3)}; {(2, 2)}) ∩ ({(1, 2), (2, 3)}; {(3, 5)})

= {(1,2)} ∩ {(2, 5)}

= {(1, 5)}

Problem 2:

**Let R ⊆ S × S be any binary relation on a set S. Consider the sequence of relations R0 , R1 , R2 , . . ., deﬁned as follows:**

**R0 := I = {( x, x ) : x ∈ S}, and**

**Ri+1 := Ri ∪ ( R; Ri ) for i ≥ 0**

**(a) Prove that if there is an i such that Ri = Ri+1 , then Rj = Ri for all j ≥ i.**

According to the specification:

Inductive proof: for any j ≥ i, Rj = Ri holds

Assume j = i + a, a ＞0

Base case (a = 1):

Ri = Ri+1 holds

And Ri+1 = Ri = Ri ∪ (R; Ri) holds

Inductive case: assuming that Ri+a = Ri for any a＞0 holds, then:

Ri+a+1 = Ri+a ∪ (R; Ri+a)

= Ri ∪ (R; Ri) = Ri+1 = Ri

Thus Rj = Ri for all j ≥ i holds

**(b) Prove that if there is an i such that Ri = Ri+1 , then Rk** ⊆ **Ri for all k ≥ 0.**

According to the specification:

Inductive proof: for any k ≥ 0, R0 ⊆ Rk holds:

Base case (k = 0):

R1 := R0 ∪ (R; R0)

According to the set theory, R0 ⊆ R1 holds

Inductive case(k = a for any 0 ≤a ≤i):

Assuming that R0 ⊆ Ra-1

Then Ra =Ra-1 ∪ (R; Ra-1)

According to the set theory, Ra-1 ⊆ Ra holds

Since R0 ⊆ Ra-1 holds, according to transitivity of set, R0 ⊆ Ra holds.

Thus R0 ⊆ Ri holds

for k > i, according to question (a), Rk = Ri for all k ≥ i,

And R0  ⊆ Ri

So Rk ⊆ Ri for all k ≥ 0 holds

**(c) Let P(n) be the proposition that for all m ∈ N: Rn ; Rm = Rn+m . Prove that P(n) holds for all n∈ N.**

Inductive proof: for any n ∈ N, Rm = Rn+m holds

Base case(n = 0):

Since R0 = I = {(x, x): x ∈ S}

For each relation (a, b) in Rm, from question1, we have

I; Rm = R0; Rm = Rm

Inductive case(n > 0) :

Assuming that for any n-1 > 0 Rn-1; Rm = Rn+m+1 holds (1)

Then Rn ; Rm  = [Rn-1 ∪ (R; Rn-1)]; Rm

from Problem1 (a)(d), we have proved that

(R1 ∪ R2); R3 = (R1; R3) ∪ (R2; R3) (2)

(R1; R2); R3 = R1; (R2; R3) (3)

Then Rn ; Rm  = [Rn-1 ∪ (R; Rn-1)]; Rm

= (Rn-1; Rm­) ∪[(R ; Rn-1); Rm­] (2)

= Rn+m-1 ∪ [(R ; (Rn-1; Rm)­] (3)

= Rn+m-1 ∪ [R ; Rn+m-1­] (1)

= Rn+m

Thus for all n ∈ N, P(N) holds.

**(d) If |S| = k, explain why Rk = Rk+1.(Hint: Show that if (a, b) ∈ Rk+1 then (a, b) ∈ Ri for some i< k + 1)**

From problem 2(a)(b), we have proved that if there is an i such that Ri = Ri+1

Rj = Ri for all j ≥ i

And Rk⊆ Rk+1for all k ≥ 0.

To prove Rk = Rk+1, we can prove Rk+1⊆ Rkinstead:

assuming that there are some relations (a, b) ∈ Rk+1 where a, b ∈ S

Since Rk+1 = Rk ∪ (R; Rk)

Then there are two possibilities:

1. (a, b) ∈ Rk or

2. (a, b) ∈ R; Rk

(1) if (a, b) ∈ Rk holds

Then for any relation (a, b) ∈ Rk+1 , (a, b)∈Rk holds.

Thus Rk+1⊆ Rkholds.

(2) if (a, b) ∈ R; Rk holds

Then there must be a mk such that

(a, mk) ∈ R and

(mk, b) ∈ Rk

Inductive proof: for any p where k ≥ p ≥ 0, there must be (a, mk-p) ∈ R and (mk-p, b) ∈ Rk-p

Base case(p = 0) have been proven.

Inductive case: assuming that for any k ≥ p ≥ 0

(a, mk-p) ∈ R and

(mk-p, b) ∈ Rk-p holds

Since Rk-p = Rk-p-1 ∪ (R; Rk-p-1)

Then there must be a mk-p-1 such that

(a, mk-p-1) ∈ R and

(mk-p-1, b) ∈ Rk-p-1

Thus we have (m0, b) ∈ R0and

(m1, b) ∈ R1and

(m2, b) ∈ R2and

…..

(mk, b) ∈ Rk

Since from problem(b) we have proved that

Ri⊆ Ri+1for all i ≥ 0.

Thus {(mp, b): k ≥ p ≥ 0} ⊆ Rk

Since mp ∈ S, the maximum possible number of relations represented by (m, b) is |S| = k

Then (a, b) ∈ {(mp, b): k ≥ p ≥ 0} ⊆ Rk holds.

Thus (a, b) ∈Rk holds and for any relation (a, b) ∈ Rk+1 , (a, b)∈Rk holds.

So Rk+1⊆ Rkholds.

In conclusion, both Rk ⊆ Rk+1 and Rk+1 ⊆ Rk hold, then

Rk= Rk+1

**(e) If |S| = k, show that Rk is transitive.**

From (d), we have proved that if |S| = k, Then Rk = Rk+1.

Assuming that (a, b) ∈ Rk , (b, c) ∈ Rk, then according the specification:

(a, c) ∈ Rk; Rk

From Problem2 (c), we have proved that Rn ; Rm = Rn+m holds for all n, m ∈N

Thus Rk; Rk = R2k holds

From Problem2 (b), we have proved that if there is an k such that Rk = Rk+1, then Ri ⊆ Rk for all k ≥ 0

Thus R2k  ⊆ Rk holds

Then (a, c) ∈ Rk; Rk = R2k ⊆Rk holds

And (a, c) ∈Rk

Thus Rk is transitive

**(f) If |S| = k, show that (R ∪ R←)k is an equivalence relation.**

If (R ∪ R←)k is an equivalence relation, it must satisfy Reflexivity(R), Symmetricity(S), Transitivity(T)

Proof:

(1) Reflexivity(R)

Suppose that W = R ∪ R←

From Problem2 (b), we have proved that k ≥ 0, then W0 ⊆ Wk

Since W0 = I, So for all x ∈ S, there must be

{(x, x): x ∈S}⊆ I = W0 ⊆ Wk

Thus Wk is reflexive holds.

(2) Transitivity(T)

From Problem2(e), we have proved that Rk is transitive where k = |S|

Regard W = R ∪ R← as a whole

Then Wk is transitive holds

(3) Symmetricity

According to the specification:

W0 = I

W1 = W0 ∪(W0; W)

According to problem 1(b), it can be simplified as (base case):

W1 = W0 ∪ W

Then W2 = W1 ∪ (W1; W)

= W1∪ ((W0 ∪ W) ; W)

= W1∪ ((W0; W) ∪ (W; W))

= W1∪ (W; W)

Inductive case:

Assuming that

Wn = Wn-1 ∪ (((W; W);W)…;W) holds

(\*note that W occurs n times in the latter operator, and replacing it by Wg)

Then Wn = Wn-1 ∪ Wg holds

And Wn = Wn-1 ∪(Wn-1; W)

Wn-1; W ⊆ Wn

Then, to prove that Wn+1 = Wn ∪ (Wg ; W)

We have Wn+1 = Wn ∪ (Wn; W)

= Wn ∪(Wn-1 ; W)

= Wn ∪ ((Wn-1 ∪ Wg ) ;W)

= Wn∪(Wn-1 ; W)∪(Wg ;W)

= Wn ∪ (Wg ; W)

Thus Wn+1 = Wn ∪ (((W; W);W)…;W) holds (W occurs (n+1) times)

Inductive proof: Wn is symmetric for all n>=0

Base case(n = 0):

Since W0 = I, for all (a, b) ∈ W0, a = b holds

Then (b, a) ∈ W0

Assuming that Wn is symmetric for all n > 1

Then Wn+1 = Wn ∪ (((W; W);W)…;W)

Since W = R ∪ R←

Then for all x, y ∈ S such that (x, y) ∈ R, there must be (y, x) ∈ R← and vice versa

Thus W is symmetric

Then for all (x, y) ∈W, there must be (y, x) ∈W such that

(x, x) ∈ W; W

Thus W; W ⊆ I

Then

Both of them is symmetric

Thus for all n > 0, if Wn is symmetric holds

Wn+1 = Wn ∪ (((W; W);W)…;W)

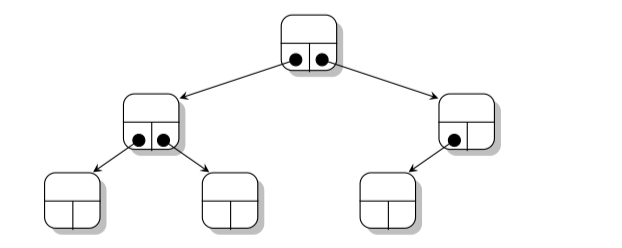
Wn+1 is symmetric holds and Wn is symmetric for all n ≥ 0

Then Wk is symmetric

In conclusion, (R ∪ R←)k is an equivalence relation

Problem 3

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

****

**If we allow empty binary trees (trees with no nodes), then we can simplify the description by saying a node has exactly two children which are binary trees.**

(**a) Give a recursive deﬁnition of the binary tree data structure. Hint: review the recursive deﬁnition of a Linked List**

A binary tree is:

① (B) an empty tree

② (R) a node point to two binary tree children(\*Left\_tree, \*Right\_tree)

(\* means a reference or a pointer(like pointer in C)

**A leaf in a binary tree is a node that has no successors (i.e. it has two empty trees as children). A fully-internal node in a binary tree is a node that has two successors. The example above has 3 leaves and 2 fully-internal nodes.**

**(b) Based on your recursive deﬁnition above, deﬁne the function count(T) that counts the number of nodes in a binary tree T.**

According the definition, empty binary trees are trees with no nodes

Then if T is empty, then T has not successor and there is no node

count(T) = 0, T is empty (B)

Else if T is not empty, then T itself is a node

Also, there are two reference connecting to its successor

Thus, the number of nodes of its successors are:

count(Left\_tree) + count(Right\_tree)

Then the sum of nodes is:

count(T) = count(Left\_tree) + count(Right\_tree) + 1 (R)

**(c) Based on your recursive deﬁnition above, deﬁne the function leaves(T) that counts the number of leaves in a binary tree T.**

According to the definition, a leaf in a binary tree is a node that has no successors

If a Tree T is empty, there is no leaf

leaves(T) = 0, T is empty (B)

Else if a Tree T has no successor, this tree itself is a leaf

leaves(T) = 1, T.Left\_tree and T.Right\_tree are empty (B)

Else if it has successor, the number of leaves in this tree is equal to the sum of the number of leaves of its two children

leaves(T) = leaves(Left\_tree) + leaves(Right\_tree) (R)

**(d) Based on your recursive deﬁnition above, deﬁne 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.**

According to the definition, a fully-internal node in a binary tree is a node that has two successors.

If a tree T is empty or T has no successor, there is no internal node

internal(T) = 0 (B)

Else if a tree T has one successor T1, this tree itself is not an fully-internal node and the number of internal nodes is equals to the number of internal nodes of its only successor, then:

internal(T) = internal(T1) (R)

Else, this tree T has 2 successors, this tree itself is an internal node and the number of internal nodes is equals to the sum of the number of internal nodes of its two successor TL and TR plus 1

internal(T) = internal(TL) + internal(TR) + 1 (R)

**(e) If T is a binary tree, let P(T) be the proposition that leaves(T) = 1 + internal(T). Prove that P(T) holds for all binary trees T.**

Inductive proof: for any number of nodes n, leaves(Tn) = 1 + internal(Tn)

Base case(n = 1), the mother-tree Tn has no successor, then:

then the T itself is a leaf and it has no internal node

leaves(T0) = 1 and

internal(T0) = 0

Thus when n = 1, leaves(T) = internal(T) + 1 holds

Inductive case: assuming that for any n>1, P(Tn) holds, then

leaves(Tn) = internal(Tn) + 1

for the case n +1, there are two possibilities.

(1) if the new node connected to a leaf, then the number of leaves remain the same as the new-added node would become a new leaf connected to the previous one while this previous node turn to be a binary tree with one successor but not an internal node.

leaves(Tn+1) = leaves(Tn) + 1 - 1

Also, no new internal node would be generated. Then:

internal(Tn+1) = internal(Tn)

Thus leaves(Tn+1) = internal(Tn+1) + 1 holds

(2) else if the new node is added to a tree with one successor previously, then it turn it into an internal node:

internal(Tn+1) = internal(Tn) + 1

In the meanwhile, the new-added node is empty and become a leaf

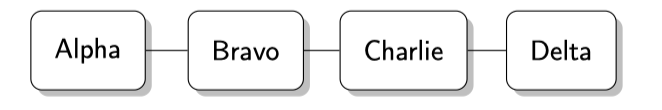
leaves(Tn+1) = leaves(Tn) + 1

Thus leaves(Tn+1) = internal(Tn+1) + 1 holds

In conclusion, the proposition P(Tn) such that leaves(Tn) = 1 + internal(Tn) holds for all n > 0

Problem 4

**Four wiﬁ networks, Alpha, Bravo, Charlie and Delta, all exist within close proximity to one another as shown below.**



**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:**

**(i) ϕ1 : Alpha uses channel hi or channel lo; and so does Bravo, Charlie and Delta.**

**(ii) ϕ2: Alpha does not use both channel hi and lo; and the same for Bravo, Charlie and Delta.**

**(iii) ϕ3: Alpha and Bravo do not use the same channel; and the same applies for all other pairs of networks connected with an edge.**

Defining the propositional variables as below:

AH: Alpha uses channel hi;

AL: Alpha uses channel lo;

BH: Bravo uses channel hi;

BL: Bravo uses channel lo;

CH: Charlie uses channel hi;

CL: Charlie uses channel lo;

DH: Delta uses channel hi;

DL: Delta uses channel lo;

ϕ1 : (AH ∨ AL) ∧ (BH ∨ BL) ∧ (CH ∨ CL) ∧ (DH ∨ DL)

ϕ2 : ((AH ∧ ￢AL) ∨ (￢AH ∧ AL)) ∧

((BH ∧ ￢BL) ∨ (￢BH ∧ BL)) ∧

((CH ∧ ￢CL) ∨ (￢CH ∧ CL)) ∧

((DH ∧ ￢DL) ∨ (￢DH ∧ DL)) ∧

ϕ3 : ((AH ∧ BL) ∨ (AL ∧ BH)) ∧

((BH ∧ CL) ∨ (BL ∨ CH)) ∧

((CH ∧ DL) ∨ (CL ∨ DH)) ∧

**(b) (i) Show that ϕ1** ∧ **ϕ2** ∧ **ϕ3 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.**

Giving a truth assignment that all these three requirements can be met:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| AH | AL | BH | BL | CH | CL | DH | DL | ϕ1 | ϕ2 | ϕ3 |
| T | F | F | T | T | F | F | T | T | T | T |
| F | T | T | F | F | T | T | F | T | T | T |

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

Based on the previous proof, to avoid interference, there are two solution:

**1.** Alpha uses channel hi; Bravo uses channel lo; Charlie uses channel hi; Delta uses channel lo;

**2.** Alpha uses channel lo; Bravo uses channel hi; Charlie uses channel lo; Delta uses channel hi;

Note that to avoid interference, each network would use either hi or lo channels but not both.