ICS Answer Sheet #3

Sakar Gopal Gurubacharya
s.gurubacharya@jacobs-university.de

Problem 3.1: cartesian products

Prove or disprove,

a. $(A \cap B) \times (C \cap D) = (A \times C) \cap (B \times D)$

From RHS to LHS,

Let, $(x, y) \in (A \times C) \cap (B \times D)$.

This means $(x, y) \in (A \times C)$ and so $x \in A$ & $y \in C$,
This also means $(x, y) \in (B \times D)$ and so $x \in B$ & $y \in D$

Now,

$x \in A$ AND $x \in B$ means $x \in (A \cap B)$, and similarly,
$y \in C$ AND $y \in D$ means $y \in (C \cap D)$.

Therefore, it also follows that. $(x, y) \in (A \cap B) \times (C \cap D)$.

From LHS to RHS,

Let, $(x, y) \in (A \cap B) \times (C \cap D)$.

This means, $x \in (A \cap B)$ and so $x \in A$ AND $x \in B$.
This also means, $y \in (C \cap D)$ and so $y \in C$ AND $y \in D$.

Now,

$x \in A$ and $y \in C$ means $(x, y) \in A \times C$, and similarly,
$x \in B$ and $y \in D$ means $(x, y) \in B \times D$.

Therefore, follows that $(x, y) \in (A \times C) \cap (B \times D)$.

Since it shows mutual inclusion, i.e. proving from both sides match,
$(A \cap B) \times (C \cap D) = (A \times C) \cap (B \times D)$

b. $(A \cup B) \times (C \cup D) = (A \times C) \cup (B \times D)$

From RHS to LHS,

Let, $(x, y) \in (A \times C) \cup (B \times D)$.

This means $(x, y) \in (A \times C)$ and so $x \in A$ & $y \in C$,
This also means $(x, y) \in (B \times D)$ and so $x \in B$ & $y \in D$

Now,

$x \in A$ OR $x \in B$ means $x \in (A \cup B)$, and similarly,
$y \in C$ OR $y \in D$ means $y \in (C \cup D)$.

Therefore, it also follows that. $(x, y) \in (A \cup B) \times (C \cup D)$.

From LHS to RHS,

Let, $(x, y) \in (A \cup B) \times (C \cup D)$.

This means, $x \in (A \cup B)$ and so $x \in A$ OR $x \in B$.
This also means, $y \in (C \cup D)$ and so $y \in C$ OR $y \in D$.

Now,

$x \in A$ and $[y \in C$ OR $y \in D]$ means $(x, y) \in A \times (C \cup D)$, and similarly,
$x \in B$ and $[y \in C$ OR $y \in D]$ means $(x, y) \in B \times (C \cup D)$

Therefore, follows that $(x, y) \in [A \times (C \cup D)] \cup [B \times (C \cup D)]$

Since it doesn't show mutual inclusion, i.e. proving from both sides don't match,
$(A \cup B) \times (C \cup D) \neq (A \times C) \cup (B \times D)$

Problem 3.2: reflexive, symmetric, transitive

a. $R = \{(a, b)|a, b \in Z \wedge |a - b| \leq 3\} = R_1$
(The absolute difference of the numbers a and b is less than or equal to 3.)

This means if b is a certain integer value, $b - 3 \leq a \leq b + 3$ …(i)
Or also if a is a certain integer value, $a - 3 \leq b \leq a + 3$ …(i)

I. For this to be **reflexive**, it should satisfy that $(a, a) \in R$ for $\forall a \in R$

The above R says that $|a - a| \leq 3$. This is always true for all $a \in R$. For a certain value for 'a' the other 'a' would also be the same. Since the relation is subtracting one 'a' from the other, the result would always be **0**. 0 is less than or equal to 3 and **therefore, the given relation is reflexive.**

II. For this to be **symmetric**, it should satisfy that $(a, b) \in R \Rightarrow (b, a) \in R$ for $\forall a$ and $b \in R$ or in other words, if $a = b$ then $b = a$.

The above properties (i) shows that it is symmetric. There are sets of values that satisfies the above expression and also switching the domain and codomain around wouldn't exclude it as the modulus fixes any possible problems.

Suppose $a = 3$ and $b = 2 \Rightarrow$ this pair falls in the set.

$|3 - 2| \leq 3 = 1 \leq 3$ (True)

If we switch the values of a and b,

$|2 - 3| \leq 3 = |-1| = 1 \leq 3$ (True)

III. For this to be **transitive,** it should satisfy that if $(a, b) \in R$ and $(b, c) \in R$ then $(a, c) \in R$ for $\forall a$, b and $c \in R$

For this, it means, if $|a - b| \leq 3$ and $|b - c| \leq 3$ then $|a - c| \leq 3$.

For the first two parts to be satisfied, we look at (i).

One number can only be from -3 to +3 than the other.

If we take the extreme end, if it holds, then we can say that it is transitive.

Let's take a = 42, b = 39 and c = 36. Each are in a series with common difference of 3 (the extreme example).

With (42, 39) (a, b), it satisfies the expression as $|a - b| \leq 3 = |42 - 39| \leq 3$ (True)

With (39, 36) (b, c), it also satisfies the expression as $|b - c| \leq 3 = |39 - 36| \leq 3$ (True)

Since the first two match, transitive property says that it should also match with (42, 36) (a, c)

However, with (42, 36) (a, c), it doesn't match the expression as $|a - c| \leq 3 = |42 - 36| \leq 3$ (False)

Since, we found a mismatch the relation cannot be transitive as it doesn't hold true for all a, b and $c \in R$

∴ $\mathbf{R_1}$ is **reflexive**, **symmetric** and **non-transitive.**

b. $R = \{(a, b) | a, b \in Z \wedge (a \bmod 10) = (b \bmod 10)\} = R_2$
(The last digit of the decimal representation of the numbers a and b is the same.)

We know that doing a **mod 10** operation to any natural number returns a range from 0 - 9
The last digit of decimal representation of a and b falls between 0 and 9… (ii)

I. For this to be **reflexive**, it should satisfy that $(a, a) \in R$ for $\forall a \in R$

If 'a' is a particular number, then the other 'a' would also be that same number. It is one of the ordered pairs in the set. Elaborating on (ii), if both the numbers are the same, their modulo by 10 would also be the same because their digit in position ones is the same.

Therefore, this relation is **reflexive.**

II. For this to be **symmetric**, it should satisfy that $(a, b) \in R \Rightarrow (b, a) \in R$ for $\forall a$ and $b \in R$ or in other words, if $a = b$ then $b = a$.

The above property (ii) shows that it is symmetric. There are sets of values that satisfies the above expression and also switching the domain and codomain around wouldn't exclude it as long as both of these have the same digit in the ones position.

Suppose $a = 64$ and $b = 2174 \Rightarrow$ this pair falls in the set.

$(64 \bmod 10) = (2174 \bmod 10) = 4$ (True)

If we switch the values of a and b,

$(2174 \bmod 10) = (64 \bmod 10) = 4$ (True)

III. For this to be **transitive,** it should satisfy that if $(a, b) \in R$ and $(b, c) \in R$ then $(a, c) \in R$ for $\forall a, b$ and $c \in R$

Similar to II, if all three of them have the same digit in the ones position, then any combination of pairs would work and **hence it would be transitive.**

The counter argument would be, let's say 'a' has a certain digit, 'b' has another and 'c' has completely different digit in position ones.

The likely thing to say here is that it is not transitive. However, as had been said for R_1, the **first two cases should be true for the property to progress**. If a certain digit for 'a' don't match that of 'b', then there is no progression here to the next steps.

Had the first two cases matched (a, b and c have all the same digit at position ones) then surely (a, c) ∈ R would also match as all three should have the same digit Therefore, it **has a transitive property.**

∴ **R_2** is **reflexive**, **symmetric** and **transitive** and hence is an **equivalence relation.**

Problem 3.3: proof by induction

Consider the two Haskell functions cnt and con defined below.

```
cnt :: Eq a => a -> [a] -> Int
cnt x [] = 0
cnt x (y:ys)
    | x == y = 1 + (cnt x ys)
    | otherwise = cnt x ys


con :: [a] -> [a] -> [a]
con [] ys = ys
con (x:xs) ys = x : (con xs ys)
```

Proof by induction over s that cnt x (con s t) == (cnt x s) + (cnt x t) holds.

Solution:

Let P(n) be the proposition that the above statement holds. Let '**n**' be a part of the set **t** and **s** as well as a subset

Step 1: Prove that the proposition holds for the base case, i.e. an empty set.

From the two sets, **s** and **t**, we let **s** be the empty set here.

Proving from both sides,

From LHS,

```
cnt n (con [] t)
```

or, **cnt n t**

From RHS,

```
(cnt n []) + (cnt n t)
```

or, **cnt n t**
Since, LHS = RHS, the proposition is true for the base case.

Step 2: Now let's assume that the given proposition is indeed true.

cnt n (con s t) == (cnt n s) + (cnt n t)

Step 3: Now, we need to prove the proposition is true for all set **s** and for one more element

Since, we are only concerned about set **s**, if this set has an extra element **'e'**

New proposition:

cnt n (con (se) t) == (cnt n (ce)) + (cnt n t)

LHS,

cnt n (con (se) t) = cnt n (e : (con s t))

The cnt guards says,

```
| n == y = 1 + (cnt n ys)
| otherwise = cnt n ys
```

0 if n, e

Then cnt n (e : (cnt s t)) = 1 + cnt n (cnt s t)

cnt n (e : (cnt s t))= 1 + (cnt n s) + (cnt n t)

1 + (cnt n s) + (cnt n t) = (cnt n(se)) + (cnt s t)

0 if n ≠ y

Then cnt n (e : (cnt s t)) = 0 + cnt n (cnt s t)

0 + cnt n (e : (cnt s t)) = 0 + (cnt n s) + (cnt n t)

Pattern matching of cnt,

0 + (cnt n s) + (cnt n t) = cnt (n (se)) + cnt (n t)

Therefore,

cnt n (con (se) t) = cnt n (se) + (cnt n t)

Step 4:

Since the proposition holds true for an extra element, we can say that the proposition holds true for all elements of the set.

So,

cnt x (con s t) == (cnt x s) + (cnt x t)

Problem 3.4: rotate a list and produce all possible rotations of a list (haskell)

a.

The following rotate function takes a string inside quotation marks and left shifts the characters in the string based on the value of n. The cycle is used so that the value of n can be greater than the string length (length xs). That makes 'rotate 1 "abcdef"' same as 'rotate 7 "abcdef"'

```
rotate :: Int -> [a] -> [a]
rotate n xs = take (length xs) (drop n (cycle xs))
```

b.

The following circle function takes a string inside quotation marks and prints all the possible rotations in a list form. It doesn't print permutations but rather all rotations. It utilizes the rotate function from above and essentially calls itself over and over. At first, it goes from 0 to length of the string and after each call, the length is reduced and makes it so that it visits every possible left shift rotation.

```
circle :: [a] -> [[a]]
circle xs = [(rotate n xs) | n <- [0..(length(xs)-1)]]
```