# CS3234 - Tutorial 5, Solutions

## 1.

**(a)** $p(a, x, f(g(y)))$ and $p(y, f(z), f(z))$

1 First derive the equations:

(1) $a = y$

(2) $x = f(z)$

(3) $f(g(y)) = f(z)$

2 Apply step 4 and replace (1) by (4) $y = a$:

(4) $y = a$

(2) $x = f(z)$

(3) $f(g(y)) = f(z)$

3 Apply step 6 and replace (3) by (5) $f(g(a)) = f(z)$, using (4) :

(4) $y = a$

(2) $x = f(z)$

(5) $f(g(a)) = f(z)$

4 Apply step 5 and replace (5) by (6) $g(a) = z$:

(4) $y = a$

(2) $x = f(z)$

(6) $g(a) = z$

5 Apply step 4 and replace (6) by (7) $z = g(a)$:

(4) $y = a$

(2) $x = f(z)$

(7) $z = g(a)$

6 Apply step 6 and replace (2) by (8) $x = f(g(a))$, using (7):

(4) $y = a$

(8) $x = f(g(a))$

(7) $z = g(a)$

We cannot further apply any other step of the unification algorithm and the most general unifier (mgu) is : $\{y/a, y/f(g(a)), z/g(a)\}$

**(b)** $p(x, g(f(a)), f(x))$ and $p(f(a), y, y)$

1 First we derive the equations:

(1) $x = f(a)$
(2) $g(f(a)) = y$
(3) $f(x) = y$

2 Apply step 6 and replace (3) by (4) $f(f(a)) = y$, using (1):

(1) $x = f(a)$
(2) $g(f(a)) = y$
(4) $f(f(a)) = y$

3 Apply step 4 and replace (2) by (5) $y = g(f(a))$

(1) $x = f(a)$
(5) $y = g(f(a))$
(4) $f(f(a)) = y$

4 Apply step 6 and replace (4) by (6)$f(f(a)) = g(f(a))$, using (5):

(1) $x = f(a)$
(5) $y = g(f(a))$
(6) $f(f(a)) = g(f(a))$

5 Apply step 5 to unify $f(f(a)) = g(f(a))$ and fail, thus atoms are not unfiable.

**(c)** $p(x, g(f(a)), f(x))$ and $p(f(y), z, y)$

1 First we derive the equations:

(1) $x = f(y)$
(2) $g(f(a)) = z$
(3) $f(x) = y$

2 Apply step 4 and replace (2) by (4) $z = g(f(a))$:

(1) $x = f(y)$
(4) $z = g(f(a))$
(3) $f(x) = y$

3 Apply step 6 and replace (3) by (5) $f(f(y)) = y$, using (1):

(1) $x = f(y)$
(4) $z = g(f(a))$
(5) $f(f(y)) = y$

4 Apply step 4 and replace (5) by (6) $y = f(f(y))$:

(1) $x = f(y)$
(4) $z = g(f(a))$
(6) $y = f(f(y))$

5 Apply step 6 to (6) in order to unify $y = f(f(y))$ and fail because y appears in the right-hand-side of the equality, thus atoms are not unifiable.

**(d)** $p(a, x, f(g(y)))$ and $p(z, h(z, u), f(u))$

1 First, we derive the equations:

(1) $a = z$
(2) $x = h(z, u)$
(3) $f(g(y)) = f(u)$

2 Apply step 4 and replace (1) by (4) $z = a$:

(4) $z = a$
(2) $x = h(z, u)$
(3) $f(g(y)) = f(u)$

3 Apply step 6 and replace (2) by (5) $x = f(a, u)$, using (4):

(4) $z = a$
(5) $x = h(a, u)$
(3) $f(g(y)) = f(u)$

4 Apply step 5 and replace (3) by (6) $g(y) = u$:

(4) $z = a$
(5) $x = h(a, u)$
(6) $g(y) = u$

5 Apply step 4 and replace (6) by (7) $u = g(y)$:

(4) $z = a$
(5) $x = h(a, u)$
(7) $u = g(y)$

5 Apply step 6 and replace (5) by (8) $x = f(a, g(y))$, using (7):

(4) $z = a$
(8) $x = f(a, g(y))$
(7) $u = g(y)$

We cannot further apply the unification algorithm and the most general unifier (mgu) is : $\{a/z, f(a, g(y))/x, g(y)/u$.

## 2.

**(a)** Let's assume that the first clause, `p(a, b)` is missing.

```
1. p(c, b)
2. p(x, y) <- p(x, z), p(z, y)
3. p(x, y) <- p(y, x)
4. Goal : <- p(a, c)
5. <- p(a, z), p(z, c)              [a/x, c/y]   4,2
6. <- p(a, z), p(z, z1), p(z1, c)   [a/x, z1/y]  5,2
7. ... infinite
```

So, there is no refutation.

**(b)** Let's assume that the second clause, `p(c, b)` is missing.

```
1. p(a, b)
2. p(x, y) <- p(x, z), p(z, y)
3. p(x, y) <- p(y, x)
4. Goal : <- p(a, c)
5. <- p(a, z), p(z, c)             [a/x, c/y]   4,2
6. <- p(a, b), p(b, c)             [b/z]        5,1
7. <- p(b, c)                                   6,1
8. <- p(b, z), p(z, c)             [b/x, c/y]   7,2
9. <- p(b, z), p(z, z1), p(z1, c)  [b/x, z1/y]  8,2
10. ... infinite
```

So, there is no refutation

**(c)** Let's assume that the third clause, `p(x, y) <- p(x, z), p(z, y)` is missing.

```
1. p(a, b)
2. p(c, b)
3. p(x, y) <- p(y, x)
4. Goal : <- p(a, c)
5. <- p(c, a)                      [a/x, c/y]  4,3
6. <- p(a, c)                      [c/x, a/y]  5,3
7. ... infinite
```

So, there is no refutation

(d) Let's assume that the forth clause, `p(x, y) <- p(y, x)` is missing.

```
1. p(a, b)
2. p(c, b)
3. p(x, y) <- p(x, z), p(z, y)
4. Goal : <- p(a, c)
5. <- p(a, z), p(z, c)             [a/x, c/y]   4,3
6. <- p(a, b), p(b, c)             [b/z]        5,1
7. <- p(b, c)                                   6,2
8. <- p(b, z), p(z, c)             [b/x, c/y]   6,3
9. <- p(b, z), p(z, z1), p(z1, c)  [b/x, z1/y]  8,3
10. ... infinite
```

So, there is no refutation.

Proof:
Clauses 3 and 4 have completely general heads,`p(x, y)`, they will match any subgoal. Thus, if clause 3 is before clause 4 in the program, the system will never consider clause 4 and vice versa. Using depth-first search algorithm (like we did), it will never be found a refutation (the leftmost path in the search tree is infinite).

**3.**

**(a)** the `split` predicate takes a list of integers and splits it into two lists containing the odd-ranked, and the even-ranked elements of the original list, respectively.

```
split ([], [], []).
split ([H|T], [H|Odds], Evens) :-
       split (T, Evens, Odds).
```

Another solution is the following:

```
split ([], [], [])
split ([A], [A], []).
split ([A, B|T], [A|T1], [B|T2]) :-
       split (T, T1, T2).
```

**(b)** the `merge` predicate takes two sorted lists of integers and merges them into a sorted list containing all the elements of the two lists

```
merge ([], T, T).
merge (T, [], T).
merge ([H1|T1], [H2|T2], [H1|T3]) :-
       H1 <= H2, merge (T1, [H2|T2], T3).
merge ([H1|T1], [H2|T2], [H2|T3]) :-
       H1 > H2, merge ([H1|T1], T2, T3).
```

**(c)** the `mergesort` predicate uses `split` and `merge`, defined previously, to sort a list of integers using the mergesort algorithm.

```
mergesort ([], []).
mergesort ([H], [H]).
mergesort ([H1, H2|T], B) :-
       split ([H1, H2|T], Aodds, Aevens),
       mergesort (Aodds, Bodds),
       mergesort (Aevens, Bevens),
       merge (Bodds, Bevens, B).
```