INF3230 – Mandatory assignment 2

Vegard Fjogstad

Exercise 62:

2. **{f(x) = g(x), g(b) = f(a)}**

First equation: by lpo-1 we have that x=x. To prove that f(x) >lpo g(x), we need to first prove that f(x) >lpo x. Using lpo-3 we can see that f(x) >lpo x.
Because f(x) >lpo x, we can use lpo-2 to prove that f(x) >lpo g(x), given precedence f > g.

Second equation: g(b) = f(a). b and f are different variables, so we cannot use lpo-3. We will instead use lpo-1 from this step to get b > f(a). Using lpo-2 we have that b >lpo a. For the second equation to hold, b > f must be true.

Combining the precedence from equation 1 with equation 2, we get the following precedence which holds: b > f > g > a

3. **{f(g(x)) = g(f(x))}**

We start with lpo-1, giving us x=x. We show that f(x) >lpo x by using lpo-3. This means that using lpo-2, we show that f(x) >lpo g(x), given precedence f > g.
Lastly, given x=x, we know that by using lpo-3 we show that g(x) >lpo x also holds.

4. **{g(f(x)) = f(g(x))}**

This problem is basically the same as the one above. We can use the same logic to get our answer. Lpo-1 for x=x. Then we show that f(x) >lpo x and g(x) >lpo x by using lpo-3. We can then show that g(x) >lpo f(x) given precedence g > f.

Eksamensoppgave 1 i INF2022 fra 2002

1b.

Eks is well founded when there is no infinite sequences like: *t >eks t' >eks t''>eks ..*
The definition of a strict partial ordering says that > (don't have the correct symbol in libreoffice) is well founded over a set S if there is no infinite sequence s1 > s2 > s2 > ... of S-elements s1, s2, s3, ...
Eks is an extension of lex. We can express all properties of the >eks relation over terms as properties of >lex over tuples of natural number, i.e α(u) >lex α(b). This is helpful because by proving that an infinite sequence would reach a contradiction with lex, we have proven the same for eks.
Given an infinite sequence α(t) >lex α(t') >lex α(t'') >lex etc ..

1c.
Eks is a simplification ordering. We can prove this by showing that the term is: *transitive, anti-reflexive, monotonic and subterm-property.* As shown above, eks can be represented using >lex. We will use this to prove the first two points.
Eks is transitive. Having three symbols: u, v, w.
u->v, v->w, u->w.
Using the definition of >lex and it's transitive properties we can substitute u, v, w with α(a), α(b), α(c). α(a)->α(b), α(b)->α(c), α(a)->α(c). This shows that eks is transitive.

Eks is anti-reflexive, because by using the definition of >lex we see that both are not reflexive. α(a) >lex α(a) does not hold for, given the properties of lexicographic order. Because of this, we know that >eks is not reflexive aswell.

Eks is monotonic. We show this with the definition of >lex. A monotone function is either entirely increasing or entirely decreasing. The very definition of lexographic ordering is that the order is of decreasing value going from left to right. In a dictionary, all words that begin with a comes before the words beginning with b. This can be written as a >lex b, showing that a has higher "value" then b, and it's therefore increasing or decreasing, depending on the way you read it. Because lex is monotonic, eks is also monotonic.

Eks satisfies the subterm-property. We can prove this because we've already proven that it is monotonic, i.e. a >lex b implies f(a) >lex f(b). Because f(a) >lex f(b), we know that f(a) >lex a also holds, as this is implied when a relation is monotonic. Because eks can be written as lex for tuples over natural numbers α(a), we know that eks also satisfies the subterm property.

1d.

1. We would have to compare the variables that could occur on both the left and right-hand side of the >eks. We do this by comparing the variables symbolic value. If their values are the same, we can simply discard them. (Example: Both sides contains one x variable. We discard it, and only look at the remaining symbols.) If one side contains more of the same variable then the other, then we store this number in case both sides has equal value after checking the symbols. (Example: Left hand side contains 2 x's, right hand side only contain 1. We then store the number 2 on the left hand side.) If both sides contain different variables, the specification is non terminating because a variable can be instantiated to any term. This coupled with checking the "normal" symbols occurence on both sides, we can check the precedence and if left hand side has a higher "value" than the right hand side, it is terminating. If a variable is only on one side, the specification is non-terminating, because they can be instantiated to any term, which means that we cannot determine which side has the higher value.
2. Using the precedence *b f g a* the specifications are terminating because *f(x) >eks g(x)* holds and *g(b) >eks f(a)* holds. We discard the x variable on both sides using the rules above.

1e.
We need to find an example where >eks can be used to prove termination, but not with >lpo. A way to do this would be to use mpo.

Exercise 74

1. Find terms r1 and r2 such that *{f(g(x)) = r1, g(h(x)) = r2}* **is confluent (and terminating).**

The way we do this is by finding an r1 and r2 that we can prove local confluence and termination. If both these demands are met, the specification is confluent. We prove local confluence by checking if all critical pairs can be joined. We prove termination by either lpo or weight functions. Proving confluence by showing local confluence and terminations is also known as the Newman's lemma.

The overlap term for the left hand sides are f(g(h(x))). We use this to find r1 = f(x) and r2 = h(x), giving us the specification *{f(g(x)) = f(x), g(h(x)) = h(x)}*. Let's prove this.

Using this, we achieve the following critical pairs:
f(g(h(x))) → Equation 1 → f(h(x))
f(g(h(x))) → Equation 2 → f(h(x))
We see that they share a common term, and the specification is therefore locally confluent.

We prove termination using lpo.
Left hand equation:
Using lpo-1 we get x=x. From this, using lpo-3 we derive that f(x) >lpo x, and finally using lpo-2 we have that f(x) > g(x), with precedence f > g.

Right hand equation:
Lpo-1 for x=x. We use lpo-2 to show that g(x) > h(x), with precedence g > h.

We combine the right hand side with the left, and get the following precedence: f > g > h.
We have proved that the specification is terminating, and using Newman's lemma, it's also confluent.

Exercise 1 Confluence

1.

{f(a) = b, f(f(x)) = x}

a and *b* are constants, whereas *x* is a variable. We can not rename these to match the second specification, meaning the specification is not confluent. We prove this by showing that it's either non terminating or locally confluent. First we find the critical pairs.

Equations do not share a variable, so we do not have to rename x to x'.

Equation 1: f(a) = b

Equation 2: f(f(x) = x

f(f(x)) $|_{\epsilon}$ f(a):

“Equation” 3: mgu {f(x) |-> a }

f(f(x)) $|_{1}$ f(a):

“Equation” 4: mgu {x |-> a}

Now we use these equations to show that we cannot reduce the critical pairs to the same term.

f(f(x)) → equation 3 → f(a) → equation 1 → b

f(f(x)) → equation 4 → f(f(a)) → equation 1 → f(b)

The critical pairs share no common term, which means that the specification is not locally confluent. It's therefore not confluent.

2.

By adding the equation b = x, the system would be confluent. We can show this by changing the substitutions for the above critical pairs.

f(f(x)) → equation 3 → f(a) → equation 1 → b → new equation → x → equation 4 → a

f(f(x)) → equation 4 → f(f(a)) → equation 1 → f(b) → new equation → f(x) → equation 3 → a

The critical pairs now reduce to the same term, and the specification is therefore locally confluent. There's no loops either, so the specification is terminating. This proves that the specification is confluent by Newman's lemma. There's also no conflicts with the previous equations.

New specification: *{f(a) = b, f(f(x) = x, b = x}*