AI2 Module 2
Tutorial 1: ANSWERS

Bonnie Webber
School of Informatics

1 Most General Unifiers

Make sure that students know what a substitution looks like: for example, {X/a, Y/f(W)} – substitute a for all instances of the variable X, and f(W) for all instances of the variable Y.

1. $\{X/Y/b, Z/b\}$
   Result is p(a,b,b)

2. *fail*
   Walk left to right across the two clauses. Start with substitution {Y/g(X,X)}. Then add X/a to get {Y/g(X,X), X/a}. At this point, one clause is q(g(a,a),g(a,b)), and the other is q(g(a,a),g(a,a)). g(a,b) and g(a,a) can't be unified, so the match fails.

3. $\{X/john, Y/john\}$
   Result is older(father(john),john)

4. *fail*
   Walk left to right across the two clauses. Start with substitution {X/father(Y)}. At this point, one clause is knows(father(Y),Y) and the other is knows(father(Y),father(Y)). The match fails here because a variable can't match with a term that contains it (i.e., Y and father(Y) in the second argument). This is the *occurs check*.

Of course Prolog unification omits the occurs check, so fails to return *fail* on the (d) example, while the R&N algorithm returns the value "fail".

2 Representation

1. **FOL** ∀X horse(X) ⊃ mammal(X)
   Horn Clause without Qs horse(X) ⊃ mammal(X)

   FOL ∀ X cow(X) ⊃ mammal(X)
   Horn Clause without Qs cow(X) ⊃ mammal(X)

2. **FOL** ∀ X,Y horse(X) ∧ offspringOf(Y,X) ⊃ horse(Y)
   HC without Qs horse(X) ∧ offspringOf(Y,X) ⊃ horse(Y)
   Read offspringOf(Y,X) as "Y is the offspring of X".

3. **FOL** and **HC without Qs** horse(bluebeard)

4. **FOL** and **HC without Qs** parent(bluebeard,charlie)
   Read parent(Y,X) as "Y is the parent of X".

5. **FOL** ∀ X,Y offspringOf(X,Y) ⊃ parentOf(Y,X)
   HC without Qs offspringOf(X,Y) ⊃ parentOf(Y,X)

FOL $\forall$ parent(Y,X) $\supset$ offspring(X,Y)
HC without Qs parent(Y,X) $\supset$ offspring(X,Y)

3 Generalised Modus Ponens

The problem here is that with left-to-right depth-first search, the fact (3) is ordered **after** the self-recursive rule (2), meaning that (2) will be tried before (3), leading to infinite expansion. So with L-R DFS, nothing follows.

If (2) is ordered **before** (3), then L-R DFS will first find horse(bluebeard) and then horse(charlie). This exhausts the factual answers.