Exercise Artificial Intelligence Assignment 3

1 Predicate Logic: Unification

(A) *{Student(x), Student(y)}*

{x/y}

(B) *{Student(passed(w), y), Student(z, grade(passed(u), v))}*

{z/passed(w), y/grade(passed(u), v)}

(C) *{Student(enthusiastic(x), passed(x), owns(y)), Student(x, y, z, owns(z))}*

Arguments count mismatch, the predicates have the same name, but the Student predicate in the first sentence takes 3 Parameters, where as the Student predicate in the second sentence takes 4 parameters, therefore the two predicates can not be the same.

(D) *{Student(y, enthusiastic(x), MATH), Student(AICOURSE, enthusiastic(x), y)}*

Rename the variables in the sentences to avoid name clashes ("standardising apart"):

{Student(y1, enthusiastic(x1), MATH), Student(AICOURSE, enthusiastic(x2), y2)}

{y1/AICOURSE, x1/x2, y2/MATH}

(E) *{Student(grade(x, AICOURSE), x), Student(grade(z, u), genius(150, v)), Student(y, genius(v, MATH))}*

"standardising apart":

{Student(grade(x, AICOURSE), x), Student(grade(z, u), genius(150, v2)), Student(y, genius(v3, MATH))}

{v3/150, v2/MATH, u/AICOURSE, x/genius(150, MATH), y/grade(genius(150, MATH), AICOURSE), z/genius(150, MATH)}

(F) *{Student(grade(x, y), genius(x)), Student(grade(enthusiastic(z), y), genius(enthusiastic(z)))}*

"standardising apart":

{Student(grade(x, y1), genius(x)), Student(grade(enthusiastic(z), y2), genius(enthusiastic(z)))}

{x/enthusiastic(z), y1/y2}

(G) *{Student(y, passed(x), MATH, z), Professor(x, AICOURSE, enthusiastic(x), y)}*

Predicate mismatch, can not unify two different predicates

(H) *{Student(passed(x), grade(passed(u), v)), Student(passed(grade(w, v)), grade(z, w))}*

"standardising apart":

{Student(passed(x), grade(passed(u), v1)), Student(passed(grade(w, v2)), grade(z, w))}

{w/v1, z/passed(u), x/grade(v1, v2)}

3 Knowledge Representation, Forward/Backward Chaining

(A) Write down logical representations for the following sentences.

Make sure they are usable for forward and backward chaining → Sentences must be **Definite Horn Clauses.**

Unicorns, jackalopes and hippocamps are mammals.

(1) Unicorn(x) ∨ Jackalope(x) ∨ Hippocamp(x) → Mammal(x)

This rule can be split up into 3 sub-rules

(1.1) Unicorn(x) → Mammal(x)

(1.2) Jackalope(x) → Mammal(x)

(1.3) Hippocamp(x) → Mammal(x)

An offspring of a unicorn is a unicorn.

(2) (Unicorn(x) ∧ Offspring(y, x)) → Unicorn(y)

HappyRainbowDancer is a unicorn.

(3) Unicorn(HappyRainbowDancer)

HappyRainbowDancer is Greenyboony's parent.

(4) Parent(HappyRainbowDancer, Greenyboony)

Offspring and parent are inverse relations.

(5) Parent(x, y) ↔ Offspring(y, x)

This rule can be split up into 2 sub-rules (by biconditional elimination and "and elimination"

(5.1) Parent(x, y) → Offspring(y, x)

(5.2) Offspring(x, y) → Parent(y, x)

Explaining some predicates:

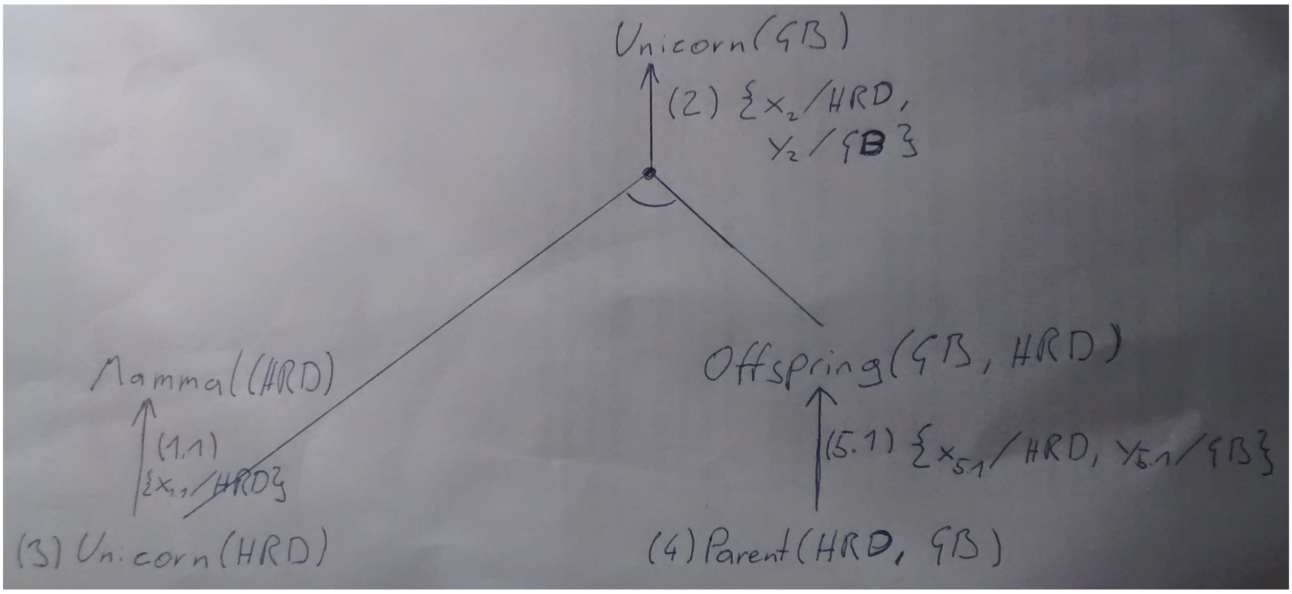
Offspring(x, y) … is true if x is an offspring of y

Parent(x, y) … is true if x is a parent of y

(B) Forward Chaining

Can you prove that Greenyboony is a unicorn? Goal is Unicorn(Greenyboony).

In the following and-or graph HappyRainbowDancer is replaced with HRD and Greenyboony with GB.



The different variable names result from “standardising apart”, I simple used the rule number to distinguish the variables.

<u>Description:</u> (Based on algorithm from the lecture slides 06_Predicate_Logic p. 29)

Iterate through every sentence in KB starting with 1.1. Check if there exists a unifier, that makes all its preconditions matchable to corresponding facts in knowledge base.

Precondition from Rule 1.1 can be matched with unifier {$x_{1.1}$/HRD} to the fact from Rule 3, the derived conclusion Mammal(HRD) is not our searched query, but will later be added to our KB.

The Rules 1.2, 1.3 and 2 can not be matched with facts in the KB. Rules 3 and 4 are already facts no need to match something here.

The Precondition from Rule 5.1 can be matched with the unifier {$x_{5.1}$/HRD, $y_{5.1}$/GB} to the fact from Rule 4, the derived conclusion Offspring(GB, HRD) is not our searched query, but will later be added to our KB.

The Rule 5.2 can not be matched with any facts in the KB.

We finished the iteration of our KB, and add our new Facts (6) Mammal(HRD) and (7) Offspring(GB, HRD) to the KB.

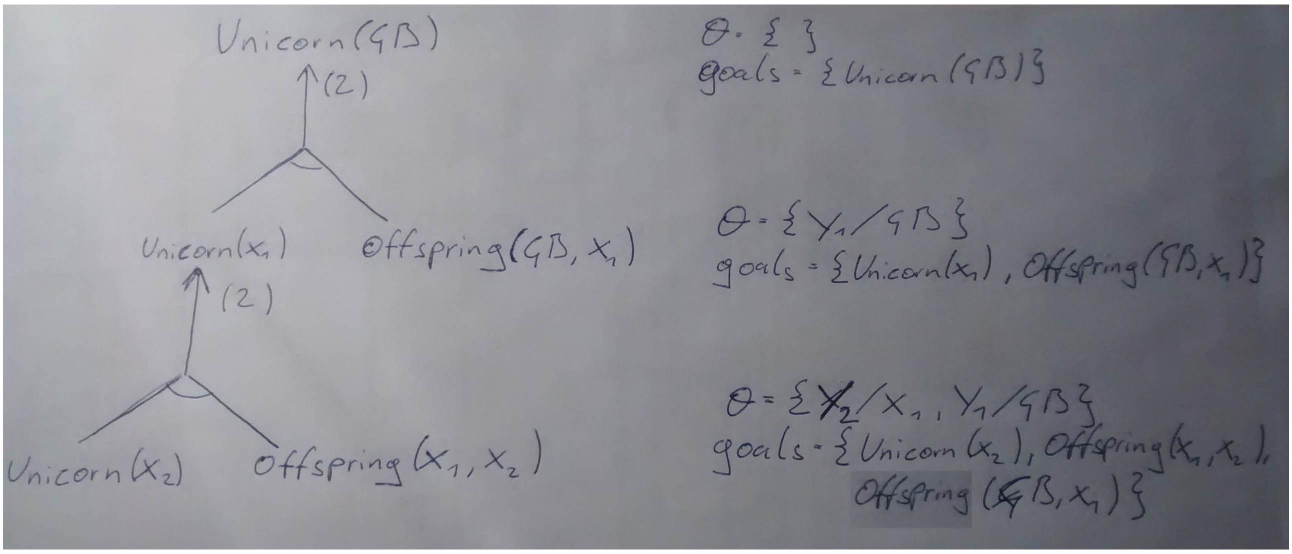
As we have new sentences in our KB we iterate it again. Rule 1.1 was already matched with Rule 3, so no new Facts here. Rule 1.2 and 1.3 still can not be matches with facts in the KB.

The Preconditions from Rule 2 can be matches with the unifier {x_2/HRD, y_2/GB} to the fact from Rule 3 and to the new Fact (7) Offspring(GB, HRD). The derived conclusion Unicorn(GB) is our query, we return the unifier {x_2/HRD, y_2/GB}.

(C) Backward Chaining

Can you prove that Greenyboony is a unicorn? Goal is Unicorn(Greenyboony).

In the following and-or graph HappyRainbowDancer is replaced with HRD and Greenyboony with GB.



The different variable names result from "standardising apart"

<u>Description:</u> (Based on algorithm from the lecture slides 06_Predicate_Logic p. 38)

The basic idea is to, start with the goal, then find an applicable rule = a rule whose conclusion (right-hand side) can be unified with the goal, then add the preconditions of this rule as new subgoals, and prove them recursively.

I can not prove that Unicorn(Greenyboony), because the algorithm, as described in the lecture slides is a left-to-right, depth-first search, and in the way our KB is ordered I get an infinite loop.

First call: goal = {Unicorn(GB)}, Θ={}, the Conclusion of Rule 2 can be unified with {y_1/GB} to match with our goal, therefore add the preconditions of Rule 2 to our goals and call algorithm again.

Second call: goal = {Unicorn(x_1), Offspring(GB, x_1)}, Θ={y_1/GB}, the first goal is used, again the Conclusion of Rule 2 can be unified to match with our goal.

Third call: goal = {Unicorn(x_2), Offspring(x_1, x_2), Offspring(GB, x_1)}, Θ={y_1/GB, y_2/x_1}

It can be observed that with the goal Unicorn(x), always Rule 2 will be matched, witch adds the precondition Unicorn(y) to the goals → infinite loop.

(D) If one of the proofs was not possible…

To avoid the infinite loop, switch Rule 2 and Rule 3, i.e. the algorithm considers Rule 3 before Rule 2. Lets go through the algorithm again:

First call: goal = {Unicorn(GB)}, Θ={}
The Conclusion of Rule 2 can be unified with {y_1/GB} to match with our goal, therefore add the preconditions of Rule 2 to our goals and call algorithm again. Rule 3 is evaluated before Rule 2 but Unicorn(HRD) can not be unified to Unicorn(GB) therefore the algorithm continues to iterate through the KB and eventually comes to Rule 2.

Second call: goal = {Unicorn(x_1), Offspring(GB, x_1)}, Θ={y_1/GB}
The first goal Unicorn(x_1) is used, this time, the goal can be unified to Rule 3 with {x_1/HRD},

which we add to Θ, as Rule 3 is a fact no precondition are added to your goals.

Third call: goal = {Offspring(GB, HDR)}, Θ={y_1/GB, x_1/HDR}
The remaining goal is updated, as we have a substitution for variable x_1. The Conclusion of Rule 5.1 can be unified with {$x_{5.1}$/HRD, $y_{5.1}$/GB} to match with our goal. The precondition parent(HRD, GD) is added to your goals.

Fourth call: goal = {parent(HRD, GD)}, Θ={y_1/GB, x_1/HDR, $x_{5.1}$/HRD, $y_{5.1}$/GB}
Your goal is Rule 4, no preconditions are added.

Final call: goal = {}, Θ={y_1/GB, x_1/HDR, $x_{5.1}$/HRD, $y_{5.1}$/GB}
Goal is empty, Θ will be returned.