Prolog & Peano Lecture 22

1. Prolog execution
2. Peano Arithmetic  
     
   **Prolog**

Question. How does Prolog evaluate things?

foo(…) :- bar(…), baz(…). //rule 1 about foo

//To prove foo we prove bar and baz.

foo(…) :- zonk(…), zink(…). //rule 2 about foo

//we can prove foo using rule 1 or 2

Prolog will explore these options

* Earlier options are more preferable than late ones.
* It will try rule 1 until it has exhausted all possibilities before moving on to rule 2.

head(xxx) :- tail1(yyy), tail2(zzz), tail3(www).

\_\_\_↑\_\_\_ \_\_\_\_\_\_\_\_↑\_\_\_\_\_\_\_\_\_

The head the tail

* The terms in the tail have an AND relationship  
  (i.e. we must prove both bar **and** baz to prove foo)

**Ordering**

Ordering is important as sometimes we must prove bar before baz and vice versa.

If foo doesn’t match the shape of bar and baz then the rule doesn’t apply.

Ordering can work regardless but may change the computational time (see notes on family tree in lecture 21).

So the speed/complexity if Prolog is related to the order.

foo(+, +,-) //means that when foo is called with the first two argument filled in then the third will be figured out.

(See notes on foo(-, -, +))

If signature is (-,-,-,) then the action goes on forever.

Example (using family tree)  
father (X, Nili)  
  
unification to see if father(X, Nili) matches father(X’,Y’)

Unify(

father(X, Nili)

father(X1, Y1)) //this works if X unifies with X1 and Nili unifies with Y1

unifier {Y1 = Nili}

Now we can prove;

parent(X, Nili) male(X)

* We prove parent and find a solution
* We then bring this solution to male
* If male X is not true then we go back to parent and find a new solution
* We bring the next solution to male

parent(Fishel, male) male(X)

{X = Fishel} {X = Fishel}

Yes no

parent(Frances, male) male(X)

{X = Frances} {X = Frances}

Yes no

In the data from lecture 21 these are the results we get.

Prolog goes through the data looking for parents of Nili and then checks if that parent is male.

**Note:** It is easy to ask questions in exams about Prolog.

**Peano**

Peano came from wanting to axiomatise numbers/arithmetic/natural numbers.

This turned out to be fruitless

We were able to partially axiomatise.

z for zero

s(z) successor of zero

Prolog will not handle commutativity very well.

Prolog can find all solutions to A + B = 3

sum(A, B, s(s(s(z))))

subtract(s(s(s(s(s(z))))), s(s(z)), X)

X = s(s(s(z))) //5-2=3

This only deals with natural numbers

mult(z, \_, z) //0\*(anything) = 0

mult(s(A), B, C) :- mult(A, B, AB), sum(AB, B, C).

ge(\_, z). //everything is greater than 0

ge(s(X), s(Y)) :- g(X,Y).

Rules of mult are not written perfectly here   
ex. X\*Y = 4

Prolog breaks when looking for a Y when X = 3