Symbolic Programming - Chapter 2 - Unification and Proof Search

Varjak Wolfe

November 17, 2021

These notes follow the online coursebook Learn Prolog Now.

Unification

Recall the different types of terms:

- Constants. These can either be atoms (such as vincent) or numbers (such as 24).
- Variables. (Such as X , Z3 , and List .)
- Complex terms. These have the form:
- functor(term1,...,termn) .

Considering the Knowledge Base from the previous chapter, we saw that Prolog unifies woman(X) with woman(mia), thereby instantiating the variable X to mia.

Two terms unify if they are the same term if they contain variables that can be uniformly instantiated with terms in such a way that the resulting terms are equal.

42 and 42 unify

mia and mia unify

woman(mia) and woman(vincent) do not unify

Now what about mia and X? They are not the same but the variable X can be instantiated to mia which makes them equal. Similarly, woman(X) and woman(mia) unify because X can be instantiated to mia.

How about loves(vincent, X) and loves (X, mia). They do not unify because there is no instantiation of X that would make them equal. Instantiating X to vincent would give the term loves(vincent, vincent) and loves (vincent, mia) which are not equal. Same if X is mia.

When Prolog unifies two terms it performs all the necessary instantiations, so that the terms really are equal afterwards. This functionality, together with the fact that we are allowed to build complex terms (that is, recursively structured terms) makes unification a powerful programming mechanism.

- If term1 and term2 are constants, then term1 and term2 unify if and only if they are the same atom, or the same number.

- If term1 is a variable and term2 is any type of term, then term1 and term2 unify, and term1 is instantiated to term2 . Similarly, if term2 is a variable and term1 is any type of term, then term1 and term2 unify, and term2 is instantiated to term1 . (So if they are both variables, they're both instantiated to each other, and we say that they share values.)

- If term1 and term2 are complex terms, then they unify if and only if: They have the same functor and arity, and all their corresponding arguments unify, and the variable instantiations are compatible. (For example, it is not possible to instantiate variable X to mia when unifying one pair of arguments, and to instantiate X to vincent when unifying another pair of arguments .)

- Two terms unify if and only if it follows from the previous three clauses that they unify.

We can test this by using a built in =/2 predicate
?- mia = mia

Proof Search

How does prolog search a knowledge base to see if a query is satisfied?
Consider the following knowledge base:
f(a).
f(b).
g(a).
g(b).
h(b).
k(X):- f(X), g(X), h(X).
Suppose we then query:
k(Y)

Prolog reads the knowledge base and tries to unify k(Y) with either a fact, or the head of a rule. It searches the knowledge base and carries out the unification, if it can, at the first place possible. Here there is only one possibility: it must unify k(Y) to the head of the rule k(X):- f(X), g(X), h(X).

When prolog unifies the variable in a query to a variable in a fact or rule, it generates a brand new variable (say _G34) to represent the shared variables. So the original query now reads:

k(_G34)

and Prolog knows that:

k(_G34) :- f(_G34), g(_G34), h(_G34).

The original query says "I want to find an indiviual that has property k". The rule says "they have property k if they have proprties f, g and h.". So if prolog can find an indivudual with these proprties, it will have satisfied the query.

Our original goal was to prove k(Y). When we unified k(Y) with the head of the rule, X, Y and _G34 were made to share the same value, giving us the goals of f(_G34), g(_G34), h(_G34).

Prolog will try to satisfy these goals one by one, left to right.

Prolog searches the knowledge base and the first item it finds that unifies with the goal of f(_sG34) is fact f(a). This satisfies the goal. Now when we unify f(_G34) to f(a), _G34 is instantiated ti a, and this instantiation applies to all occurences of _G34 in the list of goals. So the list now looks like this:

g(a), h(a)

g(a) is satisfied as above but there is no information matching h(a). We have only h(b), and this wont unify with h(a)