Logic Languages

Michael L. Scott, in Programming Language Pragmatics (Third Edition), 2009

11.2.1 Resolution and Unification

Example 11.8

Resolution in Prolog

The resolution principle, due to Robinson [Rob65], says that if C_1 and C_2 are Horn clauses and the head of C_1 matches one of the terms in the body of C_2, then we can replace the term in C_2 with the body of C_1. Consider the following example:

```
takes(jane_doe, his201).
takes(jane_doe, cs254).
takes(ajit_chandra, art302).
takes(ajit_chandra, cs254).
classmates(X, Y) :- takes(X, Z), takes(Y, Z).
```

If we let X be `jane_doe` and Z be `cs254`, wecan replace the first term on the right-hand side of the last clause with the (empty) body of the second clause, yielding the new rule

```
classmates(jane_doe, Y) :- takes(Y, cs254).
```

In other words, Y is a classmate of `jane_doe` if Y takes `cs254`. Note that the last rule has a variable (Z) on the right-hand side that does not appear in the head. Such variables are existentially quantified: for all X and Y, X and Y are classmates if there exists a class Z that they both take.

The pattern-matching process used to associate X with `jane_doe` and Z with `cs254` is known as unification. Variables that are given values as a result of unification are said to be instantiated.

The unification rules for Prolog state that

- A constant unifies only with itself.
- Two structures unify if and only if they have the same functor and the same arity, and the corresponding arguments unify recursively.

▪ A variable unifies with anything. If the other thing has a value, then the variable is instantiated. If the other thing is an uninstantiated variable, then the two variables are associated in such a way that if either is given a value later, that value will be shared by both.

Example 11.9
Unification in Prolog and ML

Unification of structures in Prolog is very much akin to ML's unification of the types of formal and actual parameters. A formal parameter of type `int * 'b list`, for example, will unify with an actual parameter of type `'a * real list` in ML by instantiating `'a` to `int` and `'b` to `real`.

Example 11.10
Equality and Unification

Equality in Prolog is defined in terms of "unifiability." The goal `=(A, B)` succeeds if and only if A and B can be unified. For the sake of convenience, the goal may be written as `A = B`; the infix notation is simply syntactic sugar. In keeping with the rules above, we have

```
?- a = a.
Yes % constant unifies with itself
?- a = b.
No % but not with another constant
?- foo(a, b) = foo(a, b).
Yes % structures are recursively identical
?- X = a.
X = a ; % variable unifies with constant
No % only once
?- foo(a, b) = foo(X, b).
X = a ; % arguments must unify
No % only one possibility
```

Example 11.11

Unification Without Instantiation

It is possible for two variables to be unified without instantiating them. If we type

```
?- A = B.
```

the interpreter will simply respond

```
A = B
```

If, however, we type

```
?- A = B, A = a, B = Y.
```

(unifying A and B before binding a to A) the interpreter will respond

```
A = a
B = a
Y = a
```

In a similar vein, suppose we are given the following rules:

```
takes_lab(S) :- takes(S, C), has_lab(C).
has_lab(D) :- meets_in(D, R), is_lab(R).
```

(S takes a lab class if S takes C and C is a lab class. Moreover D is a lab class if D meets in room R and R is a lab.) An attempt to resolve these rules will unify the head of the second with the second term in the body of the first, causing C and D to be unified, even though neither is instantiated.