Unification

Unification

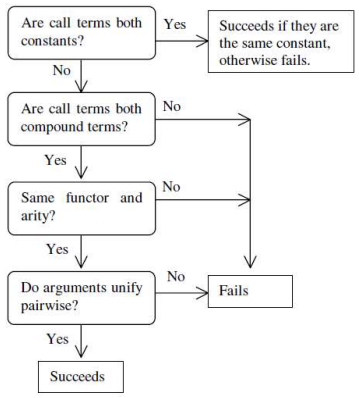
- Given a goal to evaluate, Prolog works through the clauses in the database trying to match the goal with each clause in turn, working from top to bottom until a match is found. If no match is found the goal fails.
- Prolog uses a very general form of matching known as ***unification***, which generally involves one or more variables being given values in order to make two call terms identical. This is known as ***binding*** the variables to values
- For example, the terms **dog(X)** and **dog(fido)** can be unified by binding variable *X* to atom **fido**, i.e. giving *X* the value **fido**
- The terms **owns(john,fido)** and **owns(P,Q)** can be unified by binding variables *P* and *Q* to atoms **john** and **fido**, respectively.

Unification…

- Initially all variables are ***unbound***, i.e. do not have any value
- Unlike for most other programming languages, once a variable has been bound it can be made unbound again and then perhaps be bound to a new value by ***backtracking***.

Unifying Call Terms

- The process is summarized in the following flowchart



Unifying Call Terms…

- There are three cases to consider. The simplest is when an atom is unified with another atom. This succeeds if and only if the two atoms are the same, so
  - unifying atoms **fido** and **fido** succeeds
  - unifying atoms **fido** and **'fido'** also succeeds, as the surrounding quotes are not considered part of the atom itself
  - unifying atoms **fido** and **rover** fails.

Unifying Call Terms…

- A second possibility is that an atom is unified with a compound term, e.g. **fido** with **likes(john,mary)**. This always fails.
- The third and by far the most common case is that two compound terms are unified, e.g. **likes(X,Y)** with **likes(john,mary)** or **dog(X)** with **likes(john,Y)**.
- Unification fails unless the two compound terms have the same functor and the same arity, i.e. the predicate is the same, so unifying **dog(X)** and **likes(john,Y)** inevitably fails.

Unifying Call Terms…

- Unifying two compound terms with the same functor and arity, e.g. the goal **person(X,Y,Z)** with the head **person(john,smith,27)**, requires the arguments of the head and clause to be unified 'pairwise', working from left to right, i.e. the first arguments of the two compound terms are unified, then their second arguments are unified, and so on.
- So *X* is unified with **john**, then *Y* with **smith**, then *Z* with 27
- If all the pairs of arguments can be unified (as they can in this case) the unification of the two compound terms succeeds. If not, it fails.

Unifying Call Terms…

- The arguments of a compound term can be terms of any kind, i.e. numbers, variables and lists as well as atoms and compound terms.
- Unifying two terms of this unrestricted kind involves considering more possibilities than unifying two call terms.

Unifying Call Terms: Rules

- Two atoms unify if and only if they are the same
- Two compound terms unify if and only if they have the same functor and the same arity (i.e. the predicate is the same) and their arguments can be unified pairwise, working from left to right
- Two numbers unify if and only if they are the same, so 7 unifies with 7, but not with 6.9
- Two unbound variables, say *X* and *Y* always unify, with the two variables bound to each other

Unifying Call Terms: Rules…

- An unbound variable and a term that is not a variable always unify, with the variable bound to the term
  - *X* and **fido** unify, with variable *X* bound to the atom **fido**
  - *X* and **[a,b,c]** unify, with *X* bound to list **[a,b,c]**
  - *X* and **mypred(a,b,P,Q,R)** unify, with *X* bound to **mypred(a,b,P,Q,R)**
- A bound variable is treated as the value to which it is bound

Unifying Call Terms: Rules…

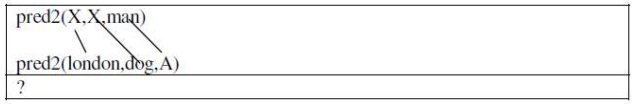
- Two lists unify if and only if they have the same number of elements and their elements can be unified pairwise, working from left to right
  - **[a,b,c]** can be unified with **[X,Y,c]**, with *X* bound to **a** and *Y* bound to **b**
  - **[a,b,c]** cannot be unified with **[a,b,d]**
- **[a,mypred(X,Y),K]** can be unified with **[P,Z,third]**, with variables *P*, *Z* and *K* bound to atom **a**, compound term **mypred(X,Y)** and atom **third**, respectively
- All other combinations of terms fail to unify

Unification Example

person(X,Y,Z) person(john,smith,27) Succeeds with variables *X*, *Y* and Z bound to *john*, *smith* and 27, respectively.
person(john,Y,23) person(X,smith,27) Fails because 23 cannot be unified with 27
pred1(X,Y,[a,b,c]) pred1(A,prolog,B) Succeeds with variables *X* and *A* bound to each other, *Y* bound to atom *prolog* and *B* bound to list *[a,b,c]*.

Unification Example: Repeated Variables

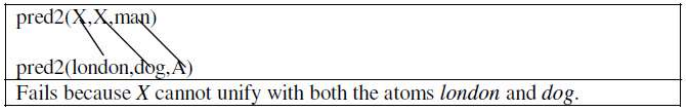
- A slightly more complicated case arises when a variable appears more than once in a compound term



- Here the first arguments of the two compound terms are unified successfully, with *X* bound to the atom **London**
- All other values of *X* in the first compound term are also bound to the atom **london** and so are effectively replaced by that value before any subsequent unification takes place

Unification Example: Repeated Variables…

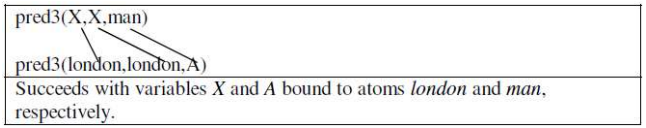
- When Prolog comes to examine the two second arguments, they are no longer *X* and **dog** but **london** and **dog**. These are different atoms and so fail to unify.



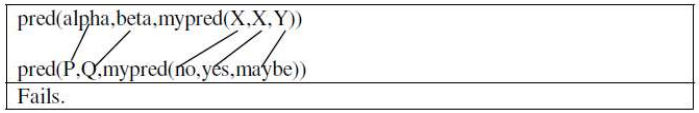
- In general, after any pair of arguments are unified, all bound variables are replaced by their values

Unification Example: Repeated Variables…

- The example below shows a successful unification involving repeated variables



Unification Example: Repeated Variables



- *P* successfully unifies with **alpha**. Next *Q* unifies with **beta**. Then Prolog attempts to unify the two third arguments, i.e. **mypred(X,X,Y)** and **mypred(no,yes,maybe)**
- The first step is to unify variable *X* with the atom **no**. This succeeds with *X* bound to **no**
- Next the two second arguments are compared. As *X* is bound to **no**, instead of *X* and **yes** the second arguments are now **no** and **yes**, so the unification fails

Unification Example: Repeated Variables…

- In the example below, the second **mypred** argument is now **no** rather than **yes**, so unification succeeds

pred(alpha,beta,mypred(X,X,Y)) pred(P,Q,mypred(no,no,maybe))
Succeeds with variables *P*, *Q*, *X* and *Y* bound to atoms *alpha*, *beta*, *no* and *maybe*, respectively.

END