Declarative Programming
CS-205
Part II: Logic Programming (Prolog)

Monika Seisenberger, Ulrich Berger

Department of Computer Science
College of Science
Swansea University

CS-205 - Unification - Proof Search

Unification

Basic idea

Two terms unify if

- they are identical terms
- or if they contain variables that can be consistently instantiated with terms in such a way that the resulting terms are equal

SOME MORE EXAMPLES

`f(a,Y)` and `f(X,g(Z))` unify with ?

`f(h(a,b),f(U,V))` and `f(W,f(a,Z))` unify with ?

`f(a,g(b))` and `f(a,b)` – do they unify?

In Prolog `'=/2'` is used to unify two terms

Unification

Basic idea

Two terms unify if

- they are identical terms
- or if they contain variables that can be consistently instantiated with terms in such a way that the resulting terms are equal

SOME MORE EXAMPLES

`f(a,Y)` and `f(X,g(Z))` unify with `X=a` and `Y=g(Z)`

`f(h(a,b),f(U,V))` and `f(W,f(a,Z))` unify with

`U = a, W = h(a,b), Z = V`

`f(a,g(b))` and `f(a,b)` DON'T unify

In Prolog `'=/2'` is used to unify two terms

Unification - Recursive Definition

Two terms `T1` and `T2` unify

1. If `T1` and `T2` are atomic, then `T1` and `T2` unify if they are the same atom, or the same number
2. If `T1` is a variable and `T2` is any type of term, then `T1` and `T2` unify, and `T1` is instantiated to `T2` (and vice versa)
3. If `T1` and `T2` are complex terms then they unify if:
   a) They have the same functor and arity,
   b) and all their corresponding arguments unify,
   c) and the variable instantiations are compatible

(Note `atomic` means and atom or a number)

Unification - Occurs Check

What happens if we try to launch the goal
`X = f(X)` ?
Sicstus Prolog will actually make `X` an *infinite* term `f(f(...))`

- A standard unification algorithm carries out an occurs check
- If it is asked to unify a variable with another term it checks whether the variable occurs in the term to avoid these possible infinite terms
- Prolog doesn't perform the occurs check for efficiency
- If you want to enforce it use `unify_with_occurs_check`
  so `unify_with_occurs_check(f(X), X)` will fail

Proof Trees

```
% We will give full proof tree for program:
% the men
man(bill). % fact 1
man(joe).  % fact 2

% who's rich
rich(joe). % fact 3

% married men
married_to(bill,jill). % fact 4
married_to(joe,ann).   % tact 5

% the happy rules
happy(M) :- man(M),married_to(M,_). % rule 1
happy(M) :- man(M),rich(M).         % rule 2
```

Proof Trees

The proof of goal `happy(X)` is

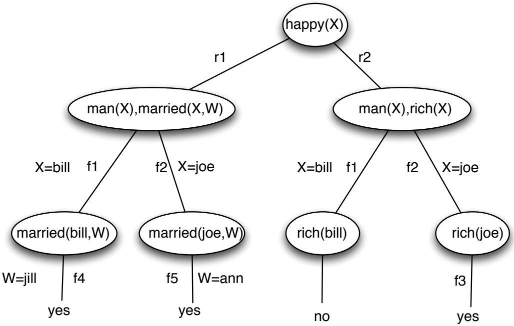

Figure 1: Proof Tree for goal `happy(X)`

Shows three ways of proving `happy(X)` with `X = bill; joe; joe`

Proof search: How about this example?

```
interesting(X) :- loop(X);fact.
loop(X) :- loop2(X).
loop2(X) :- loop(X).
fact.
```

It is recommended that you trace the query `?- interesting(X).` with having set trace on: `?-trace.`

Proof search

Prolog is using a depth-first strategy to find its responses.

Advantage: much less memory consumption than breadth-first search.

Disadvantage: Some solutions may not be found. (See previous slide.)