PROLOG PROGRAMS AND SEARCH TREE

Comp3031 Lab 06
Fall 2014

JIA Xiaoying, SU Zhiyang

Review of comparisons

- Two kinds of comparisons:
  - Term comparison: == \==
  - Arithmetic comparison: =:= =\=
- Unification:
  - E1 = E2 returns true if E1 and E2 can be unified.
  - ?- X = a.
  - X = a.

Unification Examples

- ?-a = a.
- ?-a = b.
- ?-X = a.
- ?-foo(a,Y) = foo(X,b).
- ?-2*3+4 = X + Y.
- ?-[a,b,c] = [X,Y,Z].
- ?-[a,b,c] = [X | Y].

Prolog Search tree

- A tree representing the search process of Prolog.
- If a node N1 is a child of the node N2, then the problem of proving the goal for N2 can be solved by (reducing to) proving the goal for N1.

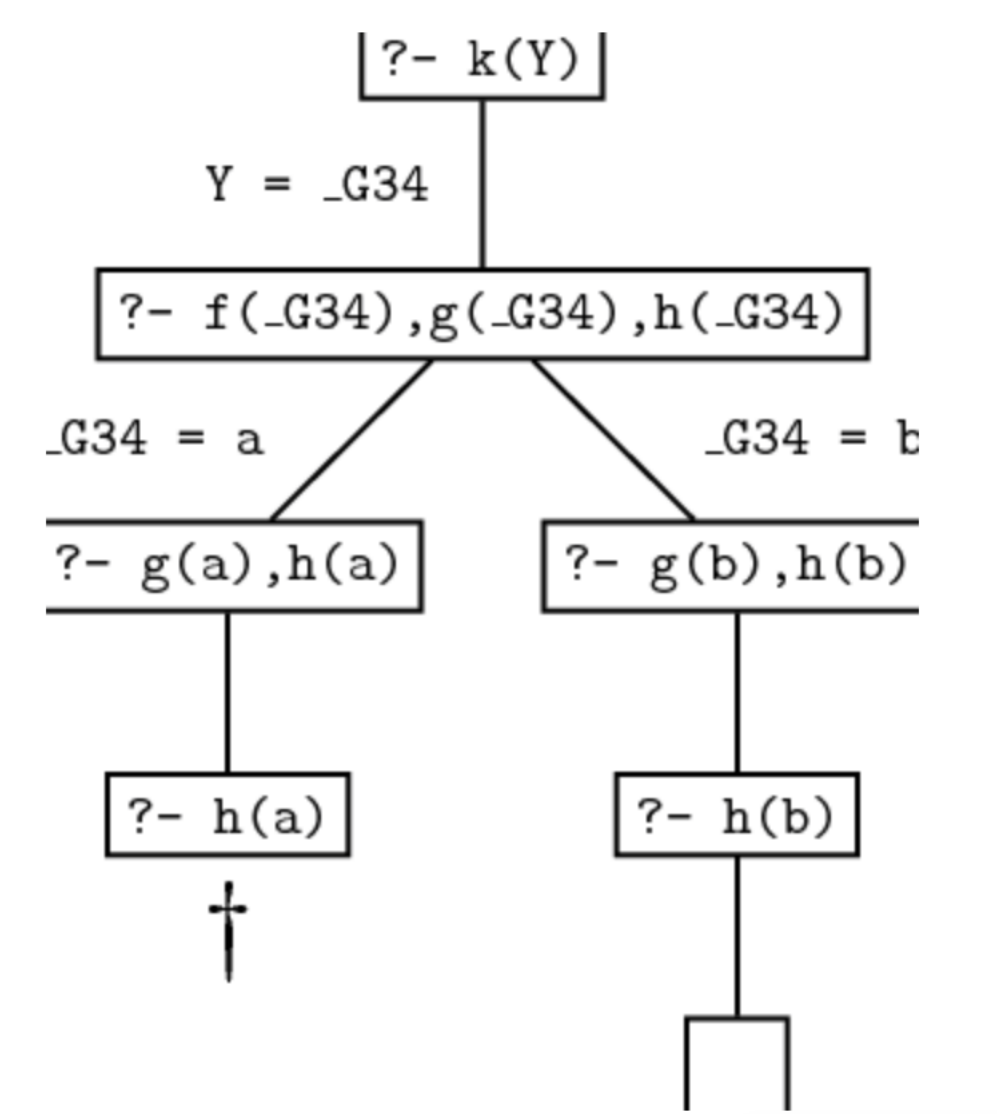
Prolog Search tree(cont.)

- The empty goal means nothing to prove, thus is ”succeeded”.
- A leaf, which is a node without children, with non-empty goal is a dead-end: there is no way to prove the goal, and is thus ”failed”.
- When drawing the tree, use different variables in the child node from those in the parent node, just to distinguish them.

Example

- Suppose we have the following database:
- f(a).
- f(b).
- g(a).
- g(b).
- h(b).
- k(X) :- f(X),g(X),h(X).
- Query:
- ?- k(Y).

Search Tree



Trace mode

- Search process can be checked using Trace mode
  - ?- trace.
  - [trace] ?- animal(X).
  - Call: (7) animal(_G335) ? Creep
  - …
- Hitting return will show you the next step

Trace mode(cont.)

- Call: Prolog tells what is the goal.
- Fail: the specified goal failed.
- Exit: the goal succeeded.
- Redo: Prolog is trying to find an alternative way of proving the goal.

Trace Example

- familyTree

```
 1 mother('Elizabeth','Susan').
 2 mother('Elizabeth','James').
 3 mother('Elizabeth','Joyce').
 4 mother('Susan','Kathryn').
 5 mother('Sue','Robert').
 6 father('Robert','Kathryn').
 7 female('Kathryn').
 8 male('James').
 9 parent(X,Y) :- father(X,Y).
10 parent(X,Y) :- mother(X,Y).
11 grandparent(X,Z) :- parent(X,Y),parent(Y,Z).
12 ancester(X,Y) :- parent(X,Y).
13 ancester(X,Z) :- parent(X,Y),ancester(Y,Z).
14
```

Trace Example

- trace query

```
?- [familyTree].
% familyTree compiled 0.00 sec, 4,368 bytes
true.

?- trace.
Unknown message: query(yes)
[trace]  ?- ancester(X,'James').
   Call: (7) ancester(_G335, 'James') ? creep
   Call: (8) parent(_G335, 'James') ? creep
   Call: (9) father(_G335, 'James') ? creep
   Fail: (9) father(_G335, 'James') ? creep
   Redo: (8) parent(_G335, 'James') ? creep
   Call: (9) mother(_G335, 'James') ? creep
   Exit: (9) mother('Elizabeth', 'James') ? creep
   Exit: (8) parent('Elizabeth', 'James') ? creep
   Exit: (7) ancester('Elizabeth', 'James') ? creep
X = 'Elizabeth' .
```

Exercise example

- Define a relation **count(X,L,N)** where N is the number of occurrences of X in L.
- Answer:
  - % base case
  - count(_,[],0).
  - % inductive case
  - count(X, [X|L], N) :- count(X, L, N1), N is N1+1.
  - count(X,[Y|L], N) :- X\==Y, count(X,L,N).
  - %query
  - count(5,[1,4,5,5,5],N).

Exercise1

- Given the **append** relation below:
  - append([], L, L).
  - append([H|T], L, [H|L1]) :- append(T, L, L1).
- Use **append(X,L1,L2)** to define **reverse(L1,L2)** where L2 is the reverse of L1.
- Example:
  - ?- reverse([7,up,8,down], L).
  - L = [down, 8, up, 7].

Exercise2

- Write a Prolog relation prefix(L1,L) to define the following relation:
  - First we define sublist: L1 is a sublist of L2 if and only if all elements of L1 appear consecutively in the same order as in L2
  - L1 is a prefix of L2 if and only if L1 is a sublist of L2 and the first element of L1 is the first element of L2.

Exercise2

- Examples:
  - ?- prefix(X,[1,2,3]).
  - X = [];
  - X = [1];
  - X = [1,2];
  - X = [1,2,3];