COM S-342

Recitation 11/26/18 – 11/28/18

Today

- Logic programming examples and exercises
- Logic Programming homework Q&A

Finding all solutions

- Recall the food.pl example
- We want to find all the dishes that sam likes:
  - ?- likesall(sam, L).
  - L = [dahl, tandoori, kurma, chow_mein, chop_suey, sweet_and_sour, pizza, spaghetti, chips].
- findall/3
  - findall(Template, Goal, Bag)

Finding All Solutions

- We create a rule called likesall where X is the individual and L is the list of dishes:
  - likesall(X, L) :- findall(D, likes(X, D), L).
- Where:
  - D is the template and represents the element returned from the goal likes(X, D)
  - Each element D is put in the Bag (list) L

Negation

- Negation as a failure:
  - not(P) :- call(P), !, fail.
  - not(P).
- In SWI you can use the operator '\+'

Family Example

```
male(james1).
male(charles1).
male(charles2).
male(james2).
male(george1).
male(paul).
male(sam).

female(catherine).
female(elizabeth).
female(sophia).
female(claudia).
female(fay).

/*  parent ( child, parent).  */
parent(charles1, james1).
parent(elizabeth, james1).
parent(charles2, charles1).
parent(catherine, charles1).
parent(james2, charles1).
parent(sophia, elizabeth).
parent(george1, sophia).
parent(george1, sam).
parent(catherine, fay).
parent(charles2, fay).
parent(james2, fay).
parent(sophia, paul).
parent(elizabeth,claudia).
parent(charles1, claudia).
```

```
/* married ( A,B)  - A is married to B */

married(james1, claudia).
married(claudia, james1).
married(charles1, fay).
married(fay, charles1).
married(elizabeth, paul).
married(paul, elizabeth).
married(sophia, sam).
married(sam, sophia).
```

Use of Negation

- bachelor(P) :- ??
- Where X is an individual
- Returns true or false

Use of Negation

- bachelor(P) :- male(P), not(married(P, _)).
- bachelor2(P) :- male(P), \+ married(P, _).

Subset

- Can we decide whether a list of subset of another list
  - ?- subset([1, 2], [1, 2 ,3, 4]).
  - True.
- subset(L1, L2) :-

Subset

- subset([X|R],S) :- member(X,S), subset(R,S).
- subset([],_).

Logic Programming Examples

Food Example

```
%% food.pl
indian(curry).
indian(dahl).
indian(tandoori).
indian(kurma).
mild(dahl).
mild(tandoori).
mild(kurma).
chinese(chow_mein).
chinese(chop_suey).
chinese(sweet_and_sour).
italian(pizza).
italian(spaghetti).

likes(sam,Food) :-
    indian(Food),
    mild(Food).
likes(sam,Food) :-
    chinese(Food).
likes(sam,Food) :-
    italian(Food).
likes(sam,chips).
```

Loading Food Example

- Go to the directory where the file was stored or pass the complete path
- **prolog** food.pl
  - If you install SWI-Prolog you can use **swipl**
- Find out what food sam likes
  - ?- **likes(sam, X).**
  - Type semi-colon (;) to see more values for X or type point (.) to stop the query
  - ?- likes(dan, X). %% should return **false.**

Hello World

- prolog %% start prolog
  - **[user].** %% start writing user rules and facts
  - **hello :- format(‘Hello World~n’).**
  - Type Ctrl-d
  - **hello.** %% query hello.
    - **Hello World**
    - **true.**

Debugging

- Use the built-in structure **trace** to display instantiations of values at each step:
  - prolog distance.pl
  - ?- trace.
  - ?- distance(volvo, Volvo_Distance).
- Tracing model describe prolog programs in four events:
  - 1) Call, attemps to satisfy a goal,
  - 2) Exit, when a goal has been satisfied
  - 3) Redo, when backtrack causes an attempt to resatisfy a goal
  - 4) Fail, when a goal fails

Arithmetic Expressions

- Prolog supports integer variables and integer arithmetic
- Use the **is** operator
  - Takes an arithmetic expression as right operand
  - A variable as left operand
  - C **is** 17 + 10.

Arithmetic Expressions

- Examples:
  - ?- X is 10 + 5.
  - X = 15.
  - true.
  - ?- X is 10 * 5.
  - X = 50.
  - true.

Arithmetic Expressions

- Define a predicate pow/3 that takes numbers as its first two arguments X and Y and returns as the value of its third argument a number which is X to the power of Y (pow(X, Y, Z)):
  - ?- pow(2, 3, Z).
  - Z = 8.
  - true.

Arithmetic Expressions

- Base case:
  - pow(_,0,1).
- Inductive case:
  - pow(X, Y, Z) :-
    Y1 is Y – 1,
    pow(X, Y1, Z1),
    Z is Z1 * X.
- Another solution:
  - pow(X, Y, Z) :- Z is X ** Y.

Arithmetic Expressions

- The semantics are not the same of assignment. Example:
  - Sum **is** Sum + 5. %% error in prolog
- Sum is not instantiated, reference in right side is undefined
- If Sum is instantiated, the clause fails because the left operand cannot have the current instantiation

Arithmetic Expressions

```
%% distance.pl
speed(ford, 100).
speed(chevy, 105).
speed(dodge, 95).
speed(volvo, 80).
time(ford, 20).
time(chevy, 21).
time(dodge, 24).
time(volvo, 24).
distance(X, Y) :- speed(X, Speed),
                  time(X, Time),
                  Y is Speed * Time.
```

Arithmetic Expressions

- prolog distance.pl
  - ?- distance(chevy, Chevy_Distance).
  - Chevy_Distance = 2205.
  - ?- distance(volvo, Volvo_Distance).
  - Volvo_Distance = 1920.

Debugging

```
distance(volvo, Volvo_Distance).
   Call: (8) distance(volvo, _4870) ? creep
   Call: (9) speed(volvo, _5094) ? creep
   Exit: (9) speed(volvo, 80) ? creep
   Call: (9) time(volvo, _5094) ? creep
   Exit: (9) time(volvo, 24) ? creep
   Call: (9) _4870 is 80*24 ? creep
   Exit: (9) 1920 is 80*24 ? creep
   Exit: (8) distance(volvo, 1920) ? creep
Volvo_Distance = 1920.
```

List Structures

- Lists are sequences of any number of elements
- Lists can be composed by:
  - Atoms
  - Atomic prepositions
  - Any other terms, including lists
- Syntax:
  - [apple, prune, grape, kumquat]
  - [] %% empty list
  - [X | Y] %% denotes head X and tail Y (car and cdr in LISP)

List Structures

- Check if a term is a member of a list:
  - ?- member(a, [b, c, d]).
  - false.
  - ?- member(b, [a, b, c]).
  - True.
- member(X, L) :- ??

List Structures

- member(X, [X | _]).
- member(X, [_ | L]) :-
  member(X, L).

Debugging

```
[trace]  ?- member(a, [a, b, c]).
   Call: (8) member(a, [a, b, c]) ? creep
   Exit: (8) member(a, [a, b, c]) ? creep
true .

[trace]  ?- member(d, [a, b, c]).
   Call: (8) member(d, [a, b, c]) ? creep
   Call: (9) member(d, [b, c]) ? creep
   Call: (10) member(d, [c]) ? creep
   Call: (11) member(d, []) ? creep
   Fail: (11) member(d, []) ? creep
   Fail: (10) member(d, [c]) ? creep
   Fail: (9) member(d, [b, c]) ? creep
   Fail: (8) member(d, [a, b, c]) ? creep
false.
```

Typelang homework Q&A

Questions?