CS206 Lecture 12
More Prolog Examples

G. Sivakumar
Computer Science Department
IIT Bombay
siva@iitb.ac.in
http://www.cse.iitb.ac.in/~siva

Fri, Jan 24, 2003

Plan for Lecture 12

- More Complex Examples

Some Interesting Problems

1. Coin-Change

   Given unlimited coins of some denominations (eg. 25,10,1), how to make change for some amount (58)?

2. Knight-Walk

   Given a starting square (i,j) on chessboard, how to generate all walks of length n.

3. P235 numbers

   Find all numbers (up to a bound) which have only 2, 3 or 5 as prime factors. Example such numbers: 36, 50.

Coin Change Run

```
| ?- cc(58,[25,10,1],Ans).

Ans = [0,0,58] ? ;
Ans = [0,1,48] ? ;
Ans = [0,2,38] ? ;
Ans = [0,3,28] ? ;
Ans = [0,4,18] ? ;
Ans = [0,5,8] ? ;
Ans = [1,0,33] ? ;
Ans = [1,1,23] ? ;
Ans = [1,2,13] ? ;
Ans = [1,3,3] ? ;
Ans = [2,0,8] ? ;

(10 ms) no
```

Does the solution order suggest a method?

Another Run

```
| ?- cc(58,[10,1,25],Ans).

Ans = [0,8,2] ? ;
Ans = [0,33,1] ? ;
Ans = [0,58,0] ? ;
Ans = [1,23,1] ? ;
Ans = [1,48,0] ? ;
Ans = [2,13,1] ? ;
Ans = [2,38,0] ? ;
Ans = [3,3,1] ? ;
Ans = [3,28,0] ? ;
Ans = [4,18,0] ? ;
Ans = [5,8,0] ? ;

no
```

Does this order make sense?

Coin Change Code

```
/*  written 8 sep 93 -- siva
    for finding ways to change an amount
    using given coin denonimations
     sample call   cc(58,[25,10,1],Ans)
*/

cc(0,[X],[0]). /* base case */

/* don't use first coin at all! */
cc(A,[C1|R],[0|L]) :- cc(A,R,L).

/* use first coin, if possible, at least once */
cc(A,[C1|R],[N|L]) :- A >= C1,
                      A1 is A - C1,
                      cc(A1,[C1|R],[N1|L]),
                      N is N1 + 1.
```

Counting Number of ways

```
| ?- cc(12,[5,3,2],Ans).

Ans = [0,0,6] ? ;
Ans = [0,2,3] ? ;
Ans = [0,4,0] ? ;
Ans = [1,1,2] ? ;
Ans = [2,0,1] ? ;

(2 ms) no
| ?- nways(12,[5,3,2],Ans).

Ans = 5 ? ;

no
| ?-
```

Write code for counting the number of ways as Assignment.

Knight's Walk

```
| ?- find_path(3,[1,1],Ans).
Time taken: 2
Ans = [[2,7],[1,5],[2,3],[1,1]] ? ;
Time taken: 2
Ans = [[3,4],[1,5],[2,3],[1,1]] ? ;
Time taken: 2
Ans = [[3,6],[1,5],[2,3],[1,1]] ? ;
Time taken: 2
Ans = [[4,3],[3,1],[2,3],[1,1]] ? ;
Time taken: 2
Ans = [[1,2],[3,1],[2,3],[1,1]] ? ;
Time taken: 4
Ans = [[5,2],[3,1],[2,3],[1,1]] ? ;
Time taken: 4
Ans = [[2,7],[3,5],[2,3],[1,1]] ? ;
Time taken: 4
Ans = [[4,3],[3,5],[2,3],[1,1]] ?
(4 ms) yes
```

Move Generation

An important subroutine

```
| ?- move([3,4],Ans).

Ans = [2,2] ? ;
Ans = [2,6] ? ;
Ans = [4,2] ? ;
Ans = [4,6] ? ;
Ans = [1,3] ? ;
Ans = [1,5] ? ;
Ans = [5,3] ? ;
Ans = [5,5]

(2 ms) yes
```

Move Generation Code

```
/* generate a move given X,Y.
   X1, Y1 assumed unbound. */
move([X,Y], [X1,Y1]) :- c1(X,X1), c2(Y,Y1).
move([X,Y], [X1,Y1]) :- c2(X,X1), c1(Y,Y1).

c1(X,X1) :- X > 1, X1 is X - 1.
c1(X,X1) :- X < 8, X1 is X + 1.

c2(X,X1) :- X > 2, X1 is X - 2.
c2(X,X1) :- X < 7, X1 is X + 2.
```

Code for Knight's Walk

```
/* knight's tour on a chess board */
find_path(N, S, Ans) :- cpu_time(Start),
                        path(N, [S], Ans),
                        cpu_time(End),
                        Time is End - Start,
                    write('Time taken: '), write(Time), nl.

path(0, L, L).
path(N, [C|Prev], Ans) :- N > 0, N1 is N - 1,
                                 move(C,C1),
                                 notin(Prev,C1),
                           path(N1, [C1 | [C | Prev]], Ans).

notin([], X).
notin([F|R], X) :- F \== X, notin(R,X).
```

Long Walks

```
| ?- find_path(60,[2,3],Ans).
Time taken: 1160

Ans = [[8,1],[6,2],[7,4],[8,6],[7,8],[6,6],[5,8],[7,7],[8,5],[
Time taken: 1512

Ans = [[8,1],[6,2],[7,4],[8,6],[7,8],[6,6],[5,8],[7,7],[8,5],[
Time taken: 1598

Ans = [[8,1],[6,2],[7,4],[8,6],[7,8],[6,6],[5,8],[7,7],[8,5],[
Time taken: 5643
```

Crashes for 61, 62 ...

Assigment: Use some heuristic and solve for 61, 62, 63!

P235 Numbers

Naive Solution: Generate and Test!

```
counter(1).
counter(A) :- counter(B), A is B + 1 .

p235(A) :-  counter(A), test_p235(A).

test_p235(1).
test_p235(A) :- B is A mod 2,
                B == 0, C is A // 2,
                test_p235(C).
test_p235(A) :- B is A mod 3,
                B == 0, C is A // 3,
                test_p235(C).
test_p235(A) :-
     B is A mod 5, B == 0,
     C is A // 5, test_p235(C).
```

Naive Solution

```
| ?- p235(Ans).

Ans = 1 ? ;
Ans = 2 ? ;
Ans = 3 ? ;
Ans = 4 ? ;
Ans = 5 ? ;
Ans = 6 ? ;
Ans = 6 ? ;
Ans = 8 ? ;
Ans = 9 ? ;
Ans = 10 ? ;
Ans = 10 ? ;
Ans = 12 ? ;
Ans = 12 ? ;
Ans = 12 ? ;
```

- Why duplicates?
- That's not only problem. Why? (Hint: Gaps).

Generate only Valid

```
p235(N) :- triple([X,Y,Z]),
           power(2,X,X1),
           power(3,Y,Y1),
           power(5,Z,Z1),
           Ans is (X1 * Y1) * Z1.
```

- Not in order!
- Assignment (Hard): Generate in order!
- Assignment (Hard): How many such numbers below N?