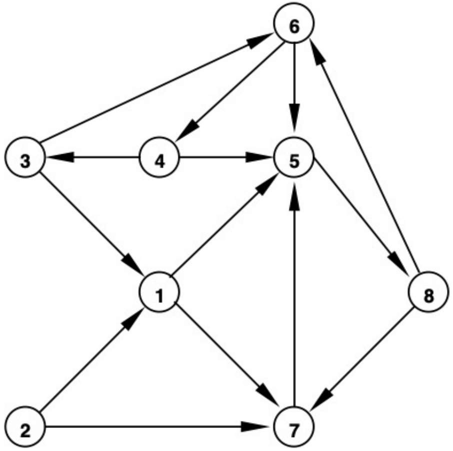
COMP4418: Knowledge Representation and Reasoning

Prolog III — Problem Solving

Maurice Pagnucco
School of Computer Science and Engineering

COMP4418, Week 3

Graph Search in Prolog



Binary Trees

- A graph may be represented by a set of edge predicates and a list of vertices

```
edge(1, 5).                  edge(1, 7).
edge(2, 1).                  edge(2, 7).
edge(3, 1).                  edge(3, 6).
edge(4, 3).                  edge(4, 5).
edge(5, 8).
edge(6, 4).                  edge(6, 5).
edge(7, 5).
edge(8, 6).                  edge(8, 7).
vertices([1, 2, 3, 4, 5, 6, 7, 8]).
```

Finding a Path

- Write a program to find a path from one node to another
- Must avoid cycles (i.e., going around in a circle)
- A template for the clause is

```
    path(Start, Finish, Visited, Path).
```

`Start` is the name of the starting node
`Finish` is the name of the finishing node
`Visited` is the list of nodes already visited
`Start` is the list of nodes on the path, including `Start` and `Finish`

The `path` Program

- The search for a path terminates when we have nowhere to go

```
    path(Node, Node, _, [Node]).
```

- A path from `Start` to `Finish` starts with a node, `X`, connected to `Start` followed by a path from `X` to `Finish`

```
path(Start, Finish, Visited, [Start|Path]) :-
    edge(Start, X),
    not(member(X, Visited)),
    path(X, Finish, [X|Visited], Path).
```

Hamiltonian Paths

- A *Hamiltonian path* is a path which spans the entire graph without any repetition of nodes in the path

```
hamiltonian(P) :-
    vertices(V),
    member(S, V),
    path(S, _, [S], P),
    subset(V, P).

subset([], _) :- !.
subset([A|B], C) :-
    member(A, C),
    subset(B, C).

: hamiltonian(P)?
P = [2, 1, 7, 5, 8, 6, 4, 3]
P = [2, 7, 5, 8, 6, 4, 3, 1]
```

Missionaries and Cannibals

- There are three missionaries and three cannibals on the left bank of a river
- They wish to cross over to the right bank using a boat that can only carry two at a time
- The number of cannibals on either bank must never exceed the number of missionaries on the same bank, otherwise the missionaries will become the cannibal’s dinner
- Plan a sequence of crossings that will take everyone safely across

Representing the State

- A state is one “snapshot” in time
- For this problem, the only information we need to fully characterise the state is:
  - the number of missionaries on the left bank
  - the number of cannibals on the left bank
  - the side the boat is on
- All other information can be deduced from these three items
- In Prolog, the state can be represented by a 3-ary term,

```
state(Missionaries, Cannibals, Side)
```

Representing the Solution

- The solution consists of a list of moves, e.g.,

```
    [move(1, 1, right), move(2, 0, left)]
```

  which we will take to mean that 1 missionary and 1 cannibal moved to the right bank, then 2 missionaries moved to the left bank
- Like the graph search problem, we must avoid returning to a state we have visited before
- The visited list will have the form:

```
    [MostRecentState | ListOfPreviousStates]
```

Overview of Solution

- We follow a simple graph search procedure
    - Start from an initial state
    - Find a neighbouring state
    - Check that the new state has not been visited before
    - Find a path from the neighbour to the goal
- The search terminates when we have found the state:

```
state(0, 0, right).
```

Top-level Prolog Code

```
% mandc(CurrentState, Visited, Path)

mandc(state(0, 0, right), _, []).
mandc(CurrentState, Visited, [Move|RestOfMoves]) :-
    newstate(CurrentState, NextState),
    not(member(NextState, Visited)),
    make_move(CurrentState, NextState, Move),
    mandc(NextState, [NextState|Visited], RestOfMoves).

make_move(state(M1,C1,left), state(M2,C2,right), move(M,C,right)) :-
    M is M1 - M2,
    C is C1 - C2.

make_move(state(M1,C1,right), state(M2,C2,left), move(M,C,left)) :-
    M is M2 - M1,
    C is C2 - C1.
```

Possible Moves

- A move is characterised by the number of missionaries and the number of cannibals taken in the boat at one time.
- Since the boat
  can carry no more than two people at once, the only possible combinations are:

```
    carry(2, 0).
    carry(1, 0).
    carry(1, 1).
    carry(0, 1).
    carry(0, 2).
```

- Where `carry(M, C)` means the boat will carry `M` missionaries and `C` cannibals on one trip

Feasible Moves

- Once we have found a possible move, we have to confirm that it is feasible
- I.e., it is not feasible to move more missionaries or more cannibals than are present on one bank
- When the state is `state(M1, C1, left)` and we try `carry(M, C)` then

  `M <= M1` *and* `C <= C1`

  must be true
- When the state is `state(M1, C1, right)` and we try `carry(M, C)` then

  `M + M1 <= 3` *and* `C + C1 <= 3`

  must be true

Legal Moves

- Once we have found a feasible move, we must check that it is legal
- I.e., no missionaries must be eaten

```
legal(X, X) :- !.
legal(3, X) :- !.
legal(0, X).
```

Generating the Next State

```
newstate(state(M1, C1, left), state(M2, C2, right)) :-
    carry(M, C),
    M <= M1,
    C <= C1,
    M2 is M1 - M,
    C2 is C1 - C,
    legal(M2, C2).
newstate(state(M1, C1, right), state(M2, C2, left)) :-
    carry(M, C),
    M2 is M1 + M,
    C2 is C1 + C,
    M2 <= 3,
    C2 <= 3,
    legal(M2, C2).
```

Logic Puzzles

Flatmates, from *Logic Problems*, Issue 10, page 35.
Six people live in a three-storey block of studio flats laid out as in the plan. From the clues given, work out the name and situation of the resident of each flat.

Flat 5	Flat 6
Flat 3	Flat 4
Flat 1	Flat 2

1. Ivor and the photographer live on the same floor.
2. Edwina lives immediately above the medical student.
3. Patrick, who is studying law, lives immediately above Ivor, and opposite the air hostess.
4. Flat 4 is the home of the store detective.
5. Doris lives in Flat 2.
6. Rodney and Rosemary are 2 of the residents in the block of flats.
7. One of the residents is a clerk.

Logic Puzzles

Residents: Doris, Edwina, Ivor, Patrick, Rodney, Rosemary.
Professions: air hostess, clerk, law student, medical student, photographer, store detective.

```
male(ivor).  male(patrick).  male(rodney).
female(doris).  female(edwina).  female(rosemary).
```

Logic Puzzle Solution

```
flatmates([L,R]) :-
    L = [[5,_,_], [3,_,_], [1,_,_]],
    R = [[6,_,_], [4,_,_], [2,_,_]],

    opposite([_,ivor,_], [_,_,photographer], L, R),

    member(C1, [L,R]),
    nextto([_,edwina,_], [_,_,medical_student], C1),

    member(C2, [L,R]),
    nextto([N1,patrick,law_student], [_,ivor,_], C2),
    opposite([N1,_,_], [_,H,air_hostess], L, R),
    female(H),

    member([4,_,store_detective], R),

    member([2,doris,_], R),

    append(L, R, A),
    member([_,rodney,_], A),
    member([_,rosemary,_], A),

    member([_,_,clerk], A).
```

Logic Puzzle Solution

```
nextto(X, Y, [X, Y|_]).
nextto(X, Y, [_|R]) :-
    nextto(X, Y, R).

opposite(X, Y, [X|_], [Y|_]).
opposite(X, Y, [Y|_], [X|_]).
opposite(X, Y, [_|R1], [_|R2]) :-
    opposite(X, Y, R1, R2).
```