University of Malta
BSc(Hons.) IT Year II

Problem Solving Techniques in Artificial Intelligence

Lecture 4

Search

- Agenda Based Search
- Implementation in Prolog

Department of Computer Science and AI
1998/9

Agenda Based Search

Many different search algorithms can be expressed with respect to an agenda of nodes using the following algorithm:

1. Take the next node from the agenda

   If there are no more nodes left, FAIL

2. If it is a goal node, SUCCEED

   Otherwise

3.(a) Expand it

  (b) Put its children on the agenda

  (c) Goto step 1

Implementing an Agenda

An agenda-handling mechanism is characterised by the following operations.

- **`agenda_new(A)`**: Assigns a brand-new agenda to the variable A.
- **`agenda_get(A,N,Anew)`**: Picks an item N from the agenda A, leaving the agenda in state Anew. Fails if there are no more items on the agenda.
- **`agenda_putl(A,L,Anew)`**: Puts a *list* of nodes on agenda A leaving it in state Anew

Nodes

The structure of nodes and the relations between them will depend on the specific problem. For the moment, we can abstract away from these details and concentrate on the following operations.

- **`node_initial(N)`**: assigns the start node to N. Note that the start node is not the same as the agenda. To get the initial agenda we must insert the start node into it.

- **`node_test(N)`**: Succeeds if N is a goal node, and fails otherwise.

- **`node_expand(N,L)`**: creates a list of N's childen and assigns it to L. If N has no children L will be the empty list.

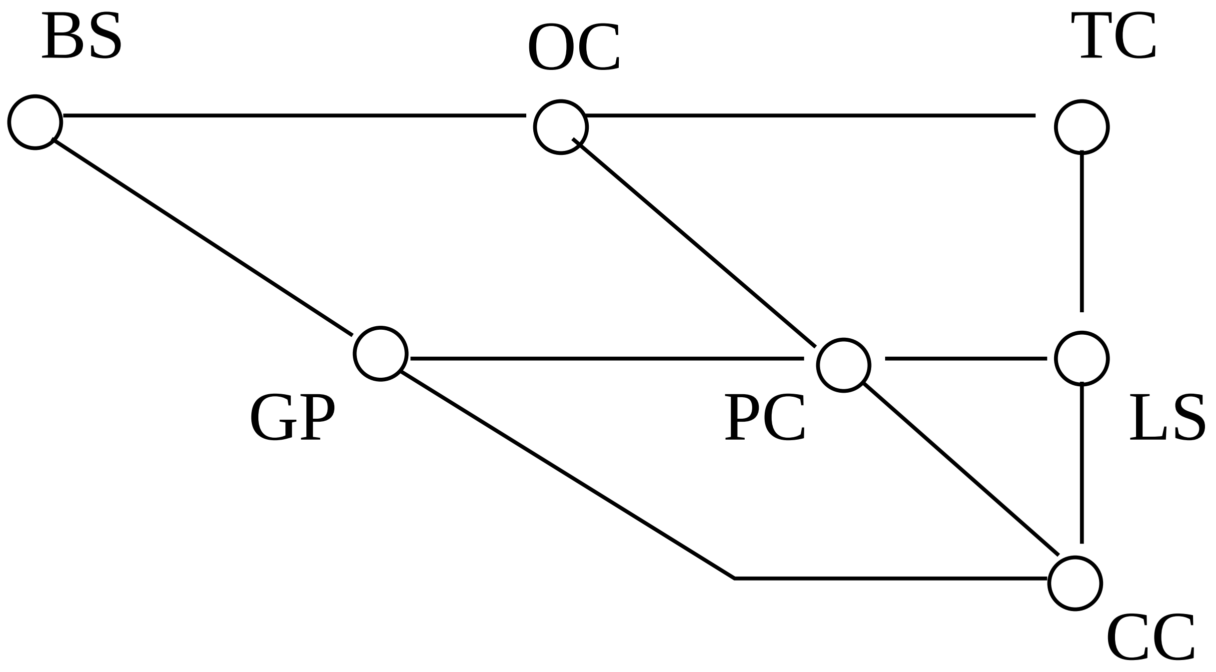
The Beef

```
agenda_search(A,Ans) :-
    agenda_get(A,N,Anew),
    agenda_search_1(N,Anew,Ans).

agenda_search_1(N,_,N) :-
    node_test(N),
    !.

agenda_search_1(N,A,Ans) :-
    node_expand(N,NodeList),
    agenda_putl(A,NodeList,NewA),
    agenda_search(NewA,Soln).
```

Encoding a Search Space



This will be encoded by associating with each node a name and the immediately adjacent nodes in alphabetical order

```
stn(bs,[gp,oc]).
stn(gp,[bs,cc,pc]).
stn(cc,[gp,ls,pc]).
stn(pc,[cc,gp,ls,oc]).
stn(ls,[cc,pc,tc]).
stn(tc,[ls,oc]).
stn(oc,[bs,pc,tc]).
```

Setting Things Up

We are now in a position to define a particular problem: getting from Bond Street (bs) to Charing Cross(cc):

```
node_initial(bs).

node_test(cc).

node_expand(N,L) :- stn(N,L).
```

At this point we can also define our agenda primitives making the simplest possible assumptions:

```
agenda_new([]).

agenda_get([X|Y],X,Y).

agenda_putl(A,L,Anew) :- append(A,L,Anew).
```

Initialising the Agenda and Go

We initialise the agenda by creating a new one, creating the initial node, and putting the initial node into the agenda. Recall that the putl operation requires a list of nodes, so we make a list of one element.

```
agenda_init(Ainit) :-
    agenda_new(A),
    node_initial(N),
    agenda_putl(A,[N],Ainit).
```

With all that loaded in, we perform a search with the following query:

```
?- agenda_init(A), agenda_search(A,N).
```

Exercise

Implement the above algorithm. How can the algorithm be modified so as to return the path from the start node to the goal?