CS3061 Artificial Intelligence

Submit to Blackboard by Mon, Feb 25.[1]

This assignment asks you to apply the A* search algorithm to the processing of propositional Prolog knowledge bases such as

```
q:- a,b.
q:- c.
a:- f.
c:- b.
c:- d,e,f.
d:- e.
e.
f:- e,d.
```

(call it `example`) which we can represent as the list

```
[[q,a,b], [q,c], [a,f], [c,b], [c,d,e,f], [d,e], [e], [f,e,d]]
```

and use in the clauses

```
arc([H|T],Node,Cost,KB) :- member([H|B],KB), append(B,T,Node),
                           length(B,L), Cost is L+1.

heuristic(Node,H) :- length(Node,H).

goal([]).
```

A file `incomplete` is supplied, defining a predicate `initKB/1` that initilizes a dynamic predicate `kb/1` to a list represeting a file, so that consulting this code, we have

```
| ?- initKB('example'), kb(KB).
KB = [[q,a,b],[q,c],[a,f],[c,b],[c,d,e,f],[d,e],[e],[f,e,d]]
```

Your task is to define the predicate

```
astar(+Node,?Path,?Cost,+KB)
```

that implements A*, returning a path to the goal node `[]` with minimal cost, given `Node` and `KB`. Among the clauses in `incomplete` is

[1] For any extensions beyond that date, email your demonstrator/marker, David Woods (dwoods@tcd.ie).

```
astar(Node,Path,Cost) :- kb(KB), astar(Node,Path,Cost,KB).
```

allowing you to test your code with queries such as

```
?- initKB('example'), astar([q],Path,Cost).
Path = [[],[e],[d],[e,d],[f],[e,f],[e,e,f],[d,e,f],[c],[q]],
Cost = 17.
```

See `hw-graph.pdf`.

Hint Modify the skeletal search algorithm

```
search([Node|_]) :-  goal(Node).
search([Node|More]) :-  findall(X,arc(Node,X),Children),
                        add-to-frontier(Children,More,New),
                        search(New).
```

so that the head of the list `New` obtained in `add-to-frontier` has f-value no larger than any in `New`'s tail, where

$$f(\text{node}) = \text{cost}(\text{node}) + h(\text{node}).$$

Let the frontier be a list of path-cost pairs (instead of just nodes), being careful to update path cost, and to bring in the heuristic function in forming the frontier `New`.

```
less-than([[Node1|_],Cost1],[[Node2|_],Cost2]) :-
   heuristic(Node1,Hvalue1), heuristic(Node2,Hvalue2),
   F1 is Cost1+Hvalue1, F2 is Cost2+Hvalue2,
   F1 =< F2.
```