Programming Paradigms – Prolog Homework

Elisa Marengo Pietro Galliani

1st Semester 2020/21

Consider the Prolog knowledge base in `basegraph.pl`. The knowledge base consists of two parts.

The first part encodes a *weighted undirected graph* using `edge/3` facts, where `edge(v,w,c)` indicates that nodes `v` and `w` are connected through an edge whose associated cost is the positive, real number `c`. In particular, `basegraph.pl` encodes one of the following graphs:

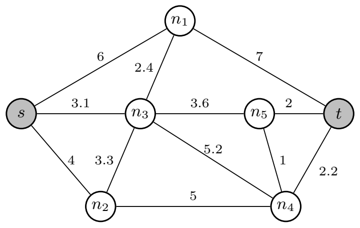

Figure 1: Graph 1

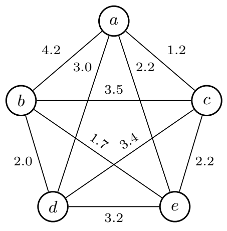

Figure 2: Graph 2

Note that `basegraph.pl` can only encode a single graph at a time, but the first part of the knowledge base can be changed so as to encode a different graph. For example, `basegraph.pl` initially encodes Graph 1, while the encoding for Graph 2 is commented in the code. You can un-comment and comment Graph 2 and Graph 1 encodings, respectively, and then re-compile it in your SWI-PL to make `basegraph.pl` encodes Graph 2 instead.

The second part of the knowledge base is instead fixed, and encodes two useful additional predicates:

- A predicate `connection(N1,N2,C)` that is true whenever `N1` and `N2` are two different nodes from the graph, such that there is an edge between `N1` and `N2` (with cost `C`) or the other way around. The graph, indeed, is undirected. Therefore, having an edge from one node to another means that the edge can be traversed in both directions, which is expressed by means of the `connection` predicate.
- A predicate `graph_nodes(L)` that is true when `L` is a list containing the set of all nodes in the graph.

By exploiting these two predicates, you have to add the following predicates.

1. A predicate `complete` that is true if the graph encoded in the knowledge base is *complete*, that is, when every pair of distinct nodes in the graph is connected by a unique edge. In the above examples, Graph 2 is complete whereas Graph 1 is not.

2. A predicate `path(S,T,P,C)` that is true if `P` is a path connecting `S` and `T` with total cost `C`. For example, by asking all solutions of the query ?- `path(s,t,P,C)`, we should obtain all paths connecting `s` with `t` (with their respective costs).

3. A predicate `min_traversal(S,T,P,C,N)` that is true if `P` is (one of) the path(s) connecting `S` and `T`, with total cost `C` and such that the number of nodes in `P` is `N`. The path `P` is such that there is no other path connecting `S` and `T` having fewer nodes. For example, by querying the knowledge base with `min_traversal(s,t,P,C,N)`, we should get the following:

```
?- min_traversal(s,t,P,C).
P = [s, n1, t],
C = 13
N = 3
```

4. A predicate `connected` that is true if the graph encoded in the knowledge base is *connected*, that is, guarantees the existence of a path between every pair of vertices.

5. A predicate `hcycle(P,C)` that is true if P is a Hamiltonian cycle in the graph with total cost `C`. A Hamiltonian cycle is a path in the graph that visits each vertex exactly once, and forms a cycle. For example, `[s, n2, n4, n5, t, n1, n3]` is a Hamiltonian cycle of Graph 1 with total cost 24.5. It is not the only one, though.

   Note. The exercise requires you to check that all nodes are traversed.

   Workarounds are not considered correct (e.g., counting that the length of a path is equal to the number of nodes in the graph is not considered valid).

6. A predicate `shcycle(P,C)` that is true if P is (one of) the shortest Hamiltonian cycle with total cost `C`. For example, by querying the knowledge base of Graph 1 with `shcycle(P,C)`, we should get the following:

```
?- shcycle(P,C).
P = [s, n2, n4, n5, t, n1, n3],
C = 24.5.
```

Deadline: 19.11.2020, 7 AM (on OLE)