Programming Paradigms Prolog Homework

Johann Gamper   Radityo Eko Prasojo   Thomas Tschager

1st Semester 2018/19

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:

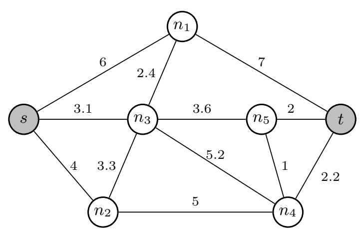

Figure 1: Graph 1

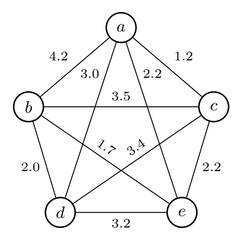

Figure 2: Graph 2

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 them with cost `C`. Notice that this predicate is symmetric, i.e., `connection(N1,N2,C)` is true if and only if `connection(N2,N1,C)` is true.
- 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:

1. Write 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. Write a predicate `path(S,T,P,L)` that is true if `P` is a path connecting `S` and `P` with total length `L`. For example, by asking all solutions of the query ?- `path(s,t,P,L)`, we should obtain all paths connecting `s` with `t` (with their respective lengths).

3. Write a predicate `spath(S,T,P,L)` that is true if `P` is (one of) the shortest path(s) connecting `S` and `P`, with total length `L`. For example, by querying the knowledge base with `spath(s,t,P,L)`, we should get the following:

```
?- spath(s,t,P,L).
P = [s, n3, n5, t],
L = 8.7.
```

4. Write 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. Write a predicate `hcycle(P,L)` that is true if P is a Hamiltonian cycle in the graph with total length `L`. 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 length of 24.5.

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

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

Deadline: 04.12.2018, 23:55 (OLE)