Homework: Computing with Logic (Simple Database Queries)

Due-date: Apr 21 at 11:59pm
Submit online on Blackboard LS

Homework must be individual's original work. Collaborations and of any form with any students or other faculty members are not allowed. If you have any questions and/or concerns, post them on Piazza and/or ask our TA or me.

Learning Outcomes

- Knowledge and application of logic-based computing
- Knowledge and application of backtracking in logic programming
- Knowledge and application of encoding requirements as logical statements
- Ability to design software following requirement specifications

Question

In this assignment, you will be given a database (represented as logical facts) on airlines and cities (we will use the term city and airport interchangeably). You assignment is to write logical queries, which will use the database and produce the results as per the specification of the assignment.

In the file airlines.pl, you will see the following types of facts:

1.
   ```
   %% airport tax and security delay information
   airport(newyork, 50, 60).
   ```
   This represents the tax (50) and delay (60) associated with New York airport.

2.
   ```
   %% flight information
   flight(aircanada, london, newyork, 500, 6)
   ```
   This represents the fact that aircanada flies between London airport and New York airport. The cost of the flight is 500 and the time of flight is 6.

In the following, we will define relevant concepts for our problem.

1. Flight Leg: a flight leg is denoted by a source airport, a destination airport and an airline that flies from the source to the destination. We will represent it as `[City1, Airline, City2]`. For instance, `[london, aircanada, newyork]` is a flight leg.

2. Flight Path: a flight path is an ordered list of flight leg. It can be an empty list as well. For instance,

   ```
   [[newyork aircanada, madrid] [valencia, iberia, barcelona],
    [barcelona, iberia, valencia]]
   ```

3. Valid Flight Path: a flight path is valid if the destination of of each leg matches the the source of the next leg (when a next leg exists in the list). The above example is not a valid flight path, because the destination of the first leg does not match with the source the second. The following

   ```
   [[newyork aircanada, madrid] [madrid, iberia, barcelona],
    [barcelona, iberia, valencia]]
   ```

   is a valid flight path.

4. Price of a Path: The price of a path is the sum of the cost of each flight in each leg, the taxes at all airports, excluding the final destination airport. (For each intermediate airport, do not consider the tax twice).

5. Duration of a path is the sum of the duration of the legs in the path plus the security delays in each airport visited (including the origin/source airport, excluding the final destination airport). (For each intermediate airport, do not consider the security delay twice).

6. Number of different airlines in a path is the number of distinct airlines used by the path. For the above valid path, the number of distinct airlines in the path is 2.

7. Flight route is a list containing the following information

   ```
   [Price, Duration, NumAirlines, Path]
   ```

   where Path is a valid loop-free path (where any city is visited at most once) and price, duration and number of airlines are properties of that path (as defined above).

Write prolog predicates to answer the following queries:

1. Write a predicate `trip(Origin, Destination, Route)`, which relates an origin city, a destination city and a route containing a flight-path from the origin to destination.

2. Write a predicate `tripk(Origin, Destination, K, Route)`, which relates an origin city, a destination city and a route containing a flight-path from origin to destination such that the duration of the flight-path is less than given K.

3. Write a predicate `multicitytrip(Origin, Destination, Intermediate, Route)` which relates an origin city, a destination city, an intermediate city and a route such that the flight-path includes the intermediate city.

A. **Extra Credit: 10% of points for this assignment.** Write a predicate `findbesttrips(Origin, Route)` which relates an origin city with the best (lowest cost) route to any other city. When querying using this predication, you can assume that the Origin city will be given.

   One possible solution to the extra credit question will be use any of shortest path algorithms (e.g., Dijkstra's algorithm). You can use `assert` and `retract`/`retractall` for answering the extra credit question.

Please review few example results for the airlines.pl database provided at the end of this document. Your results may appear in a different order.

Submission Guidelines. You are required to submit the `<netid>.pl` file only on the blackboard. If you netid is asterix, then the name of the file should be asterix.pl. Note that, the database file containing information about the city-airports and flights is provided in airlines.pl. Please add the load command at the top of your file

```
:- [airlines.pl].
```

so that you can access the facts from the file. Your submission may be evaluated with a different database entries. For each debugging, write the relationship represented by each rule in your program.
You must follow the instructions for submission. For instance, do not submit the file with any other extension, do not submit any other file, do not submit any zip files, remove/comment any inclusion of test code, remove incomplete code.

```
?- trip(malaga, madrid, R).
R = [100, 1.0, 1, [[malaga, iberia, madrid]]] ;
R = [210, 3.73, 1, [[malaga, iberia, valencia], [valencia, iberia, madrid]]] ;
R = [420, 5.58, 2, [[malaga, iberia, valencia], [valencia, iberia, barcelona],
                    [barcelona, aircanada, madrid]]] ;
R = [440, 5.68, 1, [[malaga, iberia, valencia], [valencia, iberia, barcelona],
                    [barcelona, iberia, madrid]]] ;
R = [1965, 25.58, 2, [[malaga, iberia, valencia], [valencia, iberia, barcelona],
                      [barcelona, iberia, london], [london, aircanada, newyork],
                      [newyork, iberia, madrid]]] ;
R = [2065, 25.58, 2, [[malaga, iberia, valencia], [valencia, iberia, barcelona],
                      [barcelona, iberia, london], [london, aircanada, newyork],
                      [newyork, aircanada, madrid]]] ;
R = [2115, 26.58, 3, [[malaga, iberia, valencia], [valencia, iberia, barcelona],
                      [barcelona, iberia, london], [london, aircanada, newyork],
                      [newyork, united, madrid]]] ;
R = [2115, 26.58, 2, [[malaga, iberia, valencia], [valencia, iberia, barcelona],
                      [barcelona, iberia, london], [london, united, newyork],
                      [newyork, iberia, madrid]]] ;
R = [2215, 26.58, 3, [[malaga, iberia, valencia], [valencia, iberia, barcelona],
                      [barcelona, iberia, london], [london, united, newyork],
                      [newyork, aircanada, madrid]]] ;
R = [2265, 27.58, 2, [[malaga, iberia, valencia], [valencia, iberia, barcelona],
                      [barcelona, iberia, london], [london, united, newyork],
                      [newyork, united, madrid]]] ;
false.
```

```
?- tripk(malaga, madrid, 6, R).
R = [100, 1.0, 1, [[malaga, iberia, madrid]]] ;
R = [210, 3.73, 1, [[malaga, iberia, valencia], [valencia, iberia, madrid]]] ;
R = [420, 5.58, 2, [[malaga, iberia, valencia], [valencia, iberia, barcelona],
                    [barcelona, aircanada, madrid]]] ;
R = [440, 5.68, 1, [[malaga, iberia, valencia], [valencia, iberia, barcelona],
                    [barcelona, iberia, madrid]]] ;
false.


%% Destination is left unbound
?- tripk(malaga, Destination, 3, R).
Destination = valencia,
R = [130, 2.5, 1, [[malaga, iberia, valencia]]] ;
Destination = madrid,
R = [100, 1.0, 1, [[malaga, iberia, madrid]]] ;
Destination = barcelona,
R = [275, 2.75, 2, [[malaga, iberia, madrid], [madrid, aircanada, barcelona]]] ;
Destination = barcelona,
R = [295, 2.85, 1, [[malaga, iberia, madrid], [madrid, iberia, barcelona]]] ;
Destination = valencia,
R = [215, 2.65, 1, [[malaga, iberia, madrid], [madrid, iberia, valencia]]] ;
false.

%%
?- tripK(malaga, Destination, 1, R).
false
```

```
| ?- multicitytrip(barcelona, malaga, valencia, R).

R = [375,5.4800,2,[[barcelona,aircanada,madrid],[madrid,iberia,valencia],
                  [valencia,iberia,malaga]]];
R = [395,5.5800,1,[[barcelona,iberia,madrid],[madrid,iberia,valencia],
                  [valencia,iberia,malaga]]];
R = [1920,25.4800,2,[[barcelona,iberia,london],[london,aircanada,newyork],
                    [newyork,iberia,madrid],[madrid,iberia,valencia],
                    [valencia,iberia,malaga]]];
R = [2020,25.4800,2,[[barcelona,iberia,london],[london,aircanada,newyork],
                    [newyork,aircanada,madrid],[madrid,iberia,valencia],
                    [valencia,iberia,malaga]]];
R = [2070,26.4800,3,[[barcelona,iberia,london],[london,aircanada,newyork],
                    [newyork,united,madrid],[madrid,iberia,valencia],
                    [valencia,iberia,malaga]]];
R = [2070,26.4800,2,[[barcelona,iberia,london],[london,united,newyork],
                    [newyork,iberia,madrid],[madrid,iberia,valencia],
                    [valencia,iberia,malaga]]];
R = [2170,26.4800,3,[[barcelona,iberia,london],[london,united,newyork],
                    [newyork,aircanada,madrid],[madrid,iberia,valencia],
                    [valencia,iberia,malaga]]];
R = [2220,27.4800,2,[[barcelona,iberia,london],[london,united,newyork],
                    [newyork,united,madrid],[madrid,iberia,valencia],
                    [valencia,iberia,malaga]]];
R = [270,4.0800,1,[[barcelona,iberia,valencia],[valencia,iberia,malaga]]];
R = [355,4.2300,1,[[barcelona,iberia,valencia],[valencia,iberia,madrid],
                  [madrid,iberia,malaga]]];
false
```

```
?- findbesttrips(newyork, R).
R = [850, 9, 1, [[newyork, iberia, madrid]]] ;
R = [550, 7, 1, [[newyork, aircanada, london]]] ;
R = [975, 10.25, 1, [[newyork, iberia, madrid], [madrid, iberia, malaga]]] ;
R = [965, 10.65, 1, [[newyork, iberia, madrid], [madrid, iberia, valencia]]] ;
R = [845, 13, 2, [[newyork, aircanada, london], [london, iberia, barcelona]]] ;
false.

?- findbesttrips(london, R).
R = [575, 8, 1, [[london, aircanada, newyork]]] ;
R = [295, 6, 1, [[london, iberia, barcelona]]] ;
R = [435, 7.5, 2, [[london, iberia, barcelona], [barcelona, aircanada, madrid]]] ;
R = [445, 7.75, 1, [[london, iberia, barcelona], [barcelona, iberia, valencia]]] ;
false.
```