Στυλιανού Δημήτρης
Π20004

21 Ιανουαρίου 2023

Λογικός Προγραμματισμός

3η προαιρετική εργασία

❖ Εκφώνηση εργασίας – [Για φοιτητές με επώνυμο από Ρ έως Ω] :

[Για φοιτητές με επώνυμο από Ρ έως Ω]: Develop a prolog program answering useful questions such as: what days of the week is there a direct flight from London to Athens? How can I get from Athens to Edinburgh on Thursday?
The program will be centred around a database holding the flight information. This will be represented as a three-argument relation timetable(Place1, Place2, List_of_flights) where Lisl of flights is a list of structured items of the form: Departure-time / Arrival-time / Flight-number / List-of-days
List of days is either a list of weekdays or the atom 'alldays'.
E.g.: timetable(london, edinburgh, [9:40/ 10:50 / ba4733 / alldays, 19:40 / 2O:50 / ba4833 / [mo,tu,we,th,fr,su]).
The times are represented as structured objects with two components, hours and minutes, combined by the operator ':'. (Alternatively, represent times as time(19, 40), without the need to define the operator ':').
The main problem is to find exact routes between two given cities on a given day of the week. This will be programmed as a four-argument relation:
route(Place1, Place2, Day, Route)
Here Route is a sequenceof flights that satisfies the following criteria:
(1) the start point of the route is Place1;
(2) the end point is Place2;
(3) all the flights are on the same day of the week, Day;
(4) all the flights in Route are in the timetable relation;
(5) there is enough time for transfer between flights.
Define the predicate route recursively:
route(….):- flight(….). when there is a direct flight
route(…):- route(…), flight(…), deptime(…), trandser(…) when there is no direct flight.
You will need to define and use the following auxiliary predicates:
flight(Place1, Place2, Day, Flight-num, Dep-time, Arr-time)
deptime(Route, Time)
transfer(Time1, Time2) (There should be at least 40 minutes between Time1 and Time2, which should be sufficient for transfer between two flights)
Example database:
timetable(edinburgh, london, [9:40/ 10:50 /ba4733/ alldays, 13:40 / 14:50/ ba4773 /alldays, 19:40 / 20:50 / ba4833 / [mo,tu,we,th,fr,suI]).
timetable(london, edinburgh, [9:40/ 10:50 /ba4732 / alldays, 11:40 / 12:50 / ba4752 / alldays, 18:40/ 19:50 / ba4822 / [mo,tu,we,th,fri]).
timetable(london, athens, [13:20 / 16:20 / ju201 / [fr], 13:20 / 16:20 / ju2l3 / [su]).
timetable(london, zurich, [9:10 / 11:45 /ba6l4 / alldays, 14:45 / 17:20/ sr805 / alldays).
timetable(london, milan, [8:30 / ll:20 / ba5l0 / alldays, 11:00 / 13:50 / a2459 / alldays).
timetable(athens, zurich, [11:30 / 12:40 / ju322/ [tu,th]).
timetable(athens, london, [11:10 / 12:20 / yu200 / [fr], 11:25 / 12:20 / yu212 / [sul]).
timetable(milan, london, [9:I0 / 10:00 / a2458 / alldays, 12:20 / 13:10 / ba5lI / alldays).

timetable(milar, zurich, 9:25 / l0:I5 / sr621 / alldays, 12:45 / 13:35 / sr623 / alldays).
timetable(zurich, athens, [13:30 / 14:4 / yu323 / [tu,th]).
timetable(zurich, london, [9:00 / 9:40 / ba6l3 / [mo,tu,we,th,fr,sa], 16:10 / 16:55 / sr806 / [mo,tu,we,th,fr,sul]).
timetable(zurich, milan, [7:55 / 8:45 / sr620 / alldays).

❖ Πηγαίος κώδικας:

```
% Database of flights between two cities.
timetable(edinburgh, london, [
                                [ depTime(09,40), arrTime(10,50), ba4773, alldays ],
                                [ depTime(19,40), arrTime(20,50), ba4833, [mo,tu,we,th,fr,su] ]
]).

timetable(london, edinburgh, [
                                [ depTime(09,40), arrTime(10,50), ba4732, alldays ],
                                [ depTime(18,40), arrTime(19,50), ba4822, [mo,tu,we,th,fr] ]
]).

timetable(london, athens, [
                                [ depTime(13,20), arrTime(16,20), ju201, [fr] ],
                                [ depTime(13,20), arrTime(16,20), ju213, [su] ]
]).

timetable(london, zurich, [
                                [ depTime(09,10), arrTime(11,45), ba614, alldays ],
                                [ depTime(14,45), arrTime(17,20), sr805, alldays ]
]).

timetable(london, milan, [
                                [ depTime(08,30), arrTime(11,20), ba510, alldays ],
                                [ depTime(11,00), arrTime(13,50), a2459, alldays ]
]).

timetable(athens, zurich, [
                                [ depTime(11,30), arrTime(12,40), ju322, [tu,th] ]
]).

timetable(athens, london, [
                                [ depTime(11,10), arrTime(12,20), yu200, [fr] ],
                                [ depTime(11,25), arrTime(12,20), yu212, [su] ]
]).

timetable(milan, london, [
                                [ depTime(09,10), arrTime(10,00), a2458, alldays ],
                                [ depTime(12,20), arrTime(13,10), ba511, alldays ]
]).

timetable(milan, zurich, [
                                [ depTime(09,25), arrTime(10,15), sr621, alldays ],
                                [ depTime(12,45), arrTime(13,35), sr623, alldays ]
]).

timetable(zurich, athens, [
                                [ depTime(13,30), arrTime(14,40), yu323, [tu,th] ]
]).
```

```
timetable(zurich, london, [
                                [ depTime(09,00), arrTime(09,40), ba613, [mo,tu,we,th,fr,sa] ],
                                [ depTime(16,10), arrTime(16,55), sr806, [mo,tu,we,th,fr,su] ]
]).

timetable(zurich, milan, [
                                [ depTime(07,55), arrTime(08,45), sr620, alldays ]
]).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Direct flight from Place1 to Place2 on given Day.
route(Place1, Place2, Day, [Place1, Place2, Day, FlightNum, DepTime]) :- flight(Place1, Place2, Day,
FlightNum, DepTime, _).

% flight(Place1, Place2, Day, FlightNum, DepTime, _) -> check if there is a flight from Place1
% to Place2 on the given Day and save the flight number and departure time in the variables
% FlightNum and DepTime.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Indirect flight from Place1 to Place2 on given Day.
route(Place1, Place2, Day, Routes) :-
    route(IntermediatePlace, Place2, Day, FinalFlight),
    flight(Place1, IntermediatePlace, Day, FlightNum, _, ArrTime),
    IntermediateFlight = [Place1, IntermediatePlace, Day, FlightNum, ArrTime],
    append([IntermediateFlight], [FinalFlight], Routes),
    deptime(FinalFlight, DepTime),
    transfer(ArrTime, DepTime), !.

% route(IntermediatePlace, Place2, Day, FinalFlight) -> check if there is a flight from
% IntermediatePlace to Place2 on the given Day and save the flight in the variable FinalFlight.

% flight(Place1, IntermediatePlace, Day, FlightNum, _, ArrTime) -> check if there is a flight
% from Place1 to IntermediatePlace on the given Day and save the flight number and arrival time
% in the variables FlightNum and ArrTime.

% IntermediateFlight = [Place1, IntermediatePlace, Day, FlightNum, ArrTime] -> create a list
% with the intermediate flight information.

% append([IntermediateFlight], [FinalFlight], Routes) -> append the intermediate flight list
% and the final flight list to the Routes list.

% deptime(FinalFlight, DepTime) -> save the departure time of the final flight in the variable
% DepTime.

% transfer(ArrTime, DepTime) -> % check if a transfer time of at least 40 minutes exists between
% the arrival time of the intermediate flight and the departure time of the final flight.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Return the flight between Place1 and Place2 on the given Day, FlightNum, DepTime and ArrTime.
flight(Place1, Place2, Day, FlightNum, DepTime, ArrTime) :-
    timetable(Place1, Place2, Flights),
    member([DepTime, ArrTime, FlightNum, DayTemp], Flights),
    (DayTemp == alldays, member(Day, [mo,tu,we,th,fr,sa,su]) ; member(Day, DayTemp)).

% timetable(Place1, Place2, Flights) -> check if the timetable contains a flight between
% Place1 and Place2 and save the list of flights in the variable Flights.
```

```
% member([DepTime, ArrTime, FlightNum, DayTemp], Flights) -> check if the Flights list
% contains a flight with the given DepTime, ArrTime and FlightNum and save in the variable
% DayTemp the days that the flight is available.

% (DayTemp == alldays, member(Day, [mo,tu,we,th,fr,sa,su]) ; member(Day, DayTemp)) -> check if
% the DayTemp list contains the given Day. If the DayTemp == alldays then the flight is available
% on all days of the week.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Return the departure time of a flight.
deptime([_, _, _, _, DepTime], DepTime).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Check if Time1 and Time2 have a transfer time of at least 40 minutes.
transfer(Time1, Time2) :-
    Time1 = arrTime(H1,M1), Time2 = depTime(H2,M2),
    TotalM1 is  H1*60 + M1, TotalM2 is H2*60 + M2,
    TotalM2-TotalM1 >= 40.

% Time1 = arrTime(H1,M1) -> extract the hour and minute values of Time1 which is
% in the form of arrTime(H1,M1) and assigns H1 to the hour and M1 to the minute.

% Time2 = depTime(H2,M2) -> extract the hour and minute values of Time2 which is
% in the form of depTime(H2,M2) and assigns H2 to the hour and M2 to the minute.

% TotalM1 is  H1*60 + M1 -> converts the hours and minutes of Time1 into total minutes.

% TotalM2 is  H2*60 + M2 -> converts the hours and minutes of Time2 into total minutes.

% M2-M1 >= 40 ->  check if the difference between M1 and M2 is greater than or equal to 40.
```

❖ Συνοπτική επεξήγηση κώδικα:

timetable(…): Περιέχει πληροφορίες για τις πτήσεις από ένα μέρος A σε ένα μέρος B, αποτελούν τη βάση δεδομένων.

flight(…): Επιστρέφει τη συγκεκριμένη πτήση (αν υπάρχει) από τη βάση δεδομένων (κατηγορήματα timetable), ανάλογα των δεδομένων που δίνει ο χρήστης.

route(…): Ο πρώτος κανόνας route αντιπροσωπεύει τις απευθείας πτήσεις από ένα μέρος A προς ένα μέρος B και χρησιμοποιεί τον κανόνα flight για να βρει την συγκεκριμένη πτήση. Ο δεύτερος κανόνας route είναι για τον εντοπισμό των ενδιάμεσων πτήσεων (μέσω αναδρομής) και τον έλεγχο αυτών εάν ικανοποιούν τις απαραίτητες προϋποθέσεις/συνθήκες της άσκησης.

deptime(…): Αποτελεί μέρος της διαδικασίας εντοπισμού των ενδιάμεσων πτήσεων. Επιστρέφει την ώρα αναχώρησης για μια συγκεκριμένη πτήση.

transfer(…): Αποτελεί μέρος της διαδικασίας εντοπισμού των ενδιάμεσων πτήσεων και ελέγχει εάν υπάρχει διάστημα το λιγότερο 40λεπτών, μεταξύ της ώρας άφιξης μιας ενδιάμεσης πτήσης και της ώρας αναχώρησης της επόμενης πτήσης.

❖ Παράδειγμα εκτέλεσης προγράμματος:

```
SWI-Prolog -- c:/Users/dimit/OneDrive - unipi.gr/Logikos_Programmatismos/Ergasies/Ergasia_3/ergasia_3.pl
File  Edit  Settings  Run  Debug  Help
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.0)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- route(athens,london,_,Route).
Route = [athens, london, fr, yu200, depTime(11, 10)] ;
Route = [athens, london, su, yu212, depTime(11, 25)] ;
Route = [[athens, zurich, tu, ju322, arrTime(12, 40)], [zurich, london, tu, sr806, depTime(16, 10)
]].

?- route(zurich,milan,fr,Route).
Route = [zurich, milan, fr, sr620, depTime(7, 55)] ;
Route = [[zurich, london, fr, ba613, arrTime(9, 40)], [london, milan, fr, a2459, depTime(11, 0)]].


?- route(zurich,milan,_,Route).
Route = [zurich, milan, mo, sr620, depTime(7, 55)] ;
Route = [zurich, milan, tu, sr620, depTime(7, 55)] ;
Route = [zurich, milan, we, sr620, depTime(7, 55)] ;
Route = [zurich, milan, th, sr620, depTime(7, 55)] ;
Route = [zurich, milan, fr, sr620, depTime(7, 55)] ;
Route = [zurich, milan, sa, sr620, depTime(7, 55)] ;
Route = [zurich, milan, su, sr620, depTime(7, 55)] ;
Route = [[zurich, london, mo, ba613, arrTime(9, 40)], [london, milan, mo, a2459, depTime(11, 0)]].


?-
```