German University in Cairo
Media Engineering and Technology
Prof. Dr. Slim Abdennadher

Concepts of Programming Languages, Summer 2021
Project Description
“Scheduling”
Deadline: 18/8/2020

Project Description

The scheduling problem is a very popular problem in computer science, Say that you must find a schedule of classes for the entire university to satisfy some reasonable constrains, such as that no two classes take place in the same room at the same time. If you have just a thousand classes, finding the best schedule may require centuries, even with a supercomputer.

This is where we come in and we try to take a shot at the scheduling problem. We will have a small knowledge base not a thousand classes and a small number of constrains which will make this project computable in a reasonable time.

Requirements/constrains

Lets start with our constrains, each slot can fit only one subject at a time, each subject needs 1-3 slots, maximize the number of subjects in the schedule and use all the slots available in your schedule

Facts

You will be provided by two facts these will represent our knowledge base of courses and the available schedule/slots

courses/1

```
courses([(csen403,2),(csen905,2),(csen709,1),(csen601,2),(csen301,3),(csen701,2)
,(csen503,3),(csen501,2)]).
```

The fact `courses` contains one argument which is a list of pairs(sequence of 2 items) each pair contains the course name and the number slots that this course needs.

slots/1

```
slots([slot(sunday,1),slot(sunday,2),slot(sunday,3),slot(monday,1),slot(monday,2),
slot(monday,3),slot(tuesday,1),slot(tuesday,2),slot(tuesday,3),slot(wednesday,1),
slot(wednesday,2),slot(wednesday,3)]).
```

The fact slots contains one argument which is the list of structures, each structure is `slot(Day,SlotNumber)` has 2 arguments the day and slot number in that day respectively

Predicates to be added

You are going to implement this system purely through Prolog, you can add as many predicates as you need to make sure the following predicates work correctly. This means that you are **not allowed to use any clpfd libaries**.

We describe the predicates with a bottom up approach so it's recommended to start implementing predicates in order

putSlots/1

`putSlots(L)` will succeed when the list L has `N` pairs, `N` denotes the number of slots in `slots/1` and for each pair the first argument is the slot structure of a specific slot in the knowledge base and the second argument is an unassigned variable.

A slot can't be repeated more than once and it should be in the same order as the `slots\1` fact. Example:

```
?- putSlots(L).

L = [ (slot(sunday, 1), _G375), (slot(sunday, 2), _G384), (slot(sunday, 3), _G396),
(slot(monday, 1), _G411), (slot(monday, 2), _G429), (slot(monday, 3), _G450),
(slot(tuesday, 1), _G474), (slot(tuesday, 2), _G501), (slot(tuesday, 3), _G531),
(slot(wednesday, 1), _G564), (slot(wednesday, 2), _G600), (slot(wednesday, 3), _G639)] ;
false.
```

courseNotDone/2

`courseNotDone(L,Subject)` will succeed if the second argument `Subject` is not in the list of courses `L`

Example:

```
?- courseNotDone([csen502,csen205,csen506],csen603).
true.

?- courseNotDone([csen502,csen205,csen506,csen503],csen503).
false.
```

pickAnotDoneCourse/3

The functionality of `pickAnotDoneCourse(NotDoneCourses,DoneCourses,Course)` is to pick a course (order: left to right) from the list `NotDoneCourses` and make sure it's not in the list of `DoneCourses` and it should always give me alternative answers if there are any. (note that the list is in the pair structure not a list of course names like `DoneCourses`)

Example:

?- pickAnotDoneCourse([(csen502,2),(csen302,3),(csen603,1),(csen705,2)],[csen503],C).

C = (csen502, 2) ;

C = (csen302, 3) ;

C = (csen603, 1) ;

C = (csen705, 2) ;

false.

scheduleCourse/3

This `scheduleCourse(L,Name,N)` predicate will recursively iterate over the schedule `L` and put the course `Name` in the available slots of the schedule `N` times (it should put the course in consecutive slots).

Example:

```
scheduleCourse([ (slot(sunday, 1), X), (slot(sunday, 2), Y), (slot(sunday, 3), Z)],csen502,2).
X = Y, Y = csen502 ;
false.
```

```
?- scheduleCourse([ (slot(sunday, 1), csen301), (slot(sunday, 2), Y), (slot(sunday, 3),
Z)],csen502,2).
Y = Z, Z = csen502 ;
false.
```

removeFromNotDone/2

`removeFromNotDone(C,L,L2)` will remove the Course `C` from the list `L` and will return the list `L2` which is the same list `L` without the course `C`

Example:

```
?- removeFromNotDone((csen502,3),[(csen301,1),(csen501,3),(csen502,3)],L).

L = [ (csen301, 1), (csen501, 3)] .
```

schedule/5

This predicate is where you put all pieces of the puzzle together.

`schedule(L,DoneSubjects,NotDoneSubjects,AvailableSlots,DoneSubjN)` you want to pick a course that you made sure it can fit in our schedule i.e the schedule has enough space `availableSlots`, update the schedule itself by putting the course in the schedule `L`, update the list of `NotDoneSubjects` courses, update the list of `DoneSubjects` and update the number of available slots and finally when the schedule is full you count the `DoneSubjects` and feed that value to the `DoneSubjN`

solve/2

We will provide you with the implementation of `solve` which will help you understand the implementation better and you can get an idea of how the predicate `schedule/5` works

```
solve(L,DoneSubjN):-
    putSlots(L),
    length(L,N),
    courses(NotDone),
    sort(2,@=<,NotDone,SortedCourses), %A greedy approach to the maximization problem at hand
    schedule(L,[],SortedCourses,N,DoneSubjN).
```

When we test the solve predicate with 2 variables it will create a schedule according to the facts in our knowledge base (`slots/1` and `courses/1`) in the first argument and give us the number of subjects done as follows.

```
?- solve(L,N).
L = [ (slot(sunday, 1), csen709), (slot(sunday, 2), csen403), (slot(sunday, 3), csen403),
(slot(monday, 1), csen905), (slot(monday, 2), csen905), (slot(monday, 3), csen601),
(slot(tuesday, 1), csen601), (slot(tuesday, 2), csen701), (slot(tuesday, 3), csen701),
(slot(wednesday, 1), csen301), (slot(wednesday, 2), csen301), (slot(wednesday, 3), csen301)],

N = 6
```