CUHK CSCI3230 Fall 2015
Prolog Programming Assignment - Tutorial Timeslot Scheduler

Due date: 23:59:59 (GMT +08:00), 13th November, 2015

Introduction

Scheduling tutorial timeslots for students is challenging especially for large or multi-disciplinary classes. Many heuristics have been proposed. For example, voting at the end of the first lesson and then selecting the tutorial time slot as greedy as possible; asking the busiest students to propose a certain timeslots first and let the other students vote; holding the tutorial lessons at night; using an online collaborative platform, like Google Docs, to let students mark their preferred time slots. All these methods work but cannot help tutors to automate the job.

In this assignment, you are required to write a **Prolog** program which is capable of proposing the **LEAST** number of tutorial time slots to ensure **EVERY** student can attend **AT LEAST ONE** tutorial and **EVERY** tutors can attend **ALL** the tutorials.

Requirement

In this assignment, you are required to implement a Prolog predicate:

find_time_slots(ConstraintLabels,StudentIDs,NumOfTutorials,TutorialTime).

Constraints on Tutorial Timeslots

Constraints will be imposed on timeslots to hold the tutorial sessions. They can represent the limitations in classroom booking or conflicts with tutors' timetables. Each constraint on tutorial timeslot is defined in the predicate:

constraints(Constraint_label,T).

,where

Constraint_label: a Prolog *atom* storing the label of the constraints,

T: a *list* of $\{a, n\}^N$ indicating the time slots allowed to hold tutorials, atom "a" and "n" means you can hold and cannot hold the tutorial session respectively. For example,

constraints(alice_tt,[n,n,a,n,n,a,a,a,a,a,a,n,n,n,a]).

constraints(bob_tt,[n,a,n,n,n,a,n,n,a,a,a,n,a,a,a]).

The above shows whether Alice and Bob can hold tutorials on a list of timeslots. Please note that there is no specific order and no specific requirement on the number of timeslots on the list. For example, the first element can represent "M1" or "T1" or "W5". Some timeslots has been omitted due to some reasons (say, conflicts with the lectures). However, it should be noted that any two lists must have the same representation and contain the same number of elements. For example, if the first element of Alice's Timetable indicates if she can hold tutorial on M4, the first element of Bob's Timetable (and the others') will have the same meaning.

Student Timetable

The timetable of each student is stored in the predicate:

student_timetable(Sid,T).

,where

Sid: a Prolog *atom* storing the student ID,

T: a *list* of $\{a, n\}^N$ corresponding to timetable of student with student ID Sid, "a" and "n" means a student can attend and cannot attend the tutorial session respectively, where the underlying representation is the same as the constraint lists. For example,

student_timetable(s1267633431,[a,a,a,n,n,a,n,n,a,a,a,n,n,a,a]).

student_timetable(s1246634324,[n,a,n,n,a,a,n,a,n,a,n,n,a,a,n]).

The above shows whether student s1267633431 and student s1246634324 can attend the tutorials on a list of suggested timeslots.

Problem Definition

Given a set of constraints on a tutorial time $C = [C_1\ C_2\ C_3\ \dots\ C_N]$,where C_i is a list for i = 1 to N; and a set of students $S = [S_1\ S_2\ S_3\ \dots\ S_N]$,where S_i is a *list* for i = 1 to N, who want to attend the tutorials, find a set of time slots T, for which the size of T is the minimum. Each student is required to attend **at least one** tutorial. The proposed timetable should **not** conflict with any of the constraints in C.

To find the minimum number of tutorials and tutorial time, we will use the following predicate:

find_time_slots(ConstraintLabels,StudentIDs,NumOfTutorials,TutorialTime).

where

ConstraintLabels: a *list* of the constraint labels C of constraints on the timetable,

StudentIDs: a *list* of the student identifiers S of students on the timetable,

NumOfTutorials: the minimum *number* of tutorials N_{min} offered,

TutorialTime: a *list* representing the tutorial timetable T with the minimum number of tutorials offered to students.

Please note that there can be more than one possible combination of the tutorial time slots, i.e more than one final answer, and your predicate should return **<u>any one</u>** of them.

Example

Two tutors, Alice and Bob, of a computer networking course ask for your help. You need to help them to find out the minimum number of timeslots to hold the tutorial session(s) on either Tuesday or Thursday. Assume there is only seven timeslots for you to choose on each day, you write a Prolog program, and encode their timetable in two lists in the format of [t1, t2, …, t7, h1, h2, …, h7], where t_i and h_j are either 'a' or 'n', to indicate their availability. These lists for Alice and Jimmy are labeled as *alice_tt* and *bob_tt* respectively. They are stored in two facts of a compound term *constraints*.

In their classes, there are only two students. Having obtained the students' timetables, you encode these timetables using the same method again and label them with their student IDs. These two facts are represented in the compound term *student_timetable*. Then you save these facts in a Prolog program.

```
%Facts about this course

constraints(alice_tt,[n,n,a,n,n,a,a,a,a,a,a,n,n,n]).

constraints(bob_tt,[n,a,n,n,n,a,n,n,a,a,a,n,a,a]).

student_timetable(s12676,[a,a,a,n,n,a,n,n,a,a,a,n,n,a]).

student_timetable(s12466,[n,a,n,n,a,a,n,a,n,a,n,n,a,a]).
```

Next you write a predicate *find_time_slots*. Invoking the predicate *find_time_slots* gives:

```
?- find_time_slots([alice_tt,
bob_tt],[s12676,s12466],NumTuto,Tuto).

NumTuto = 1,

Tuto = [n, n, n, n, n, a, n, n, n, n, n, n, n, n].
```

The result means that Alice and Bob only need to hold **one** tutorial session during the sixth lesson on Tuesday (because t6 is 'a') so that every student can attend one of the tutorials and they can both teach tutorials.

Marking Scheme

In the marking, we will use SWI-Prolog. There will be ten test cases and each case contributes 10% of the total marks of the assignment.

Submission

You should submit **a single file *student-id*.pl** through our online system, e.g. if your student ID is 1301234567, the name should be 1301234567.pl. Please limit the file size to be less than 1MB. You can submit multiple times (before the deadline), and the final grade will be the mark of the latest submission for this assignment. Make sure that you have **removed all the *constraints* predicates and *student_timetable* predicates** in your .pl file before submission.

Late submission will lead to marks deduction which is subject to the following penalty scheme.

Number of Days Late	Marks Deduction
1	10%
2	30%
3	60%
4 or more	100%

Plagiarism will be seriously punished.