GRADUATION ELEGIBILITY TOOL

Computational Logic Spring 2018
Project Report

Dhwani Kaneria – drk170130
Hardik Bhadja– hxb162230

Introduction:
The project aims to create an autonomous eligibility tool that assistant grad students to check their eligibility to graduate based on their current record. Project also includes the degree plan according to current course enrollment i.e. unofficial transcript and course structure. We have used python to generate a file containing facts for the sASP code to use.

Methodology:
s(ASP) system is used for implementation of the system. Facts is generated from student's transcript using automation script. Criteria to be able to graduate is encoded via rules in logic file, using those rules and the facts, answer set containing predicates described below:

_hasTaken(Student,Course).

This predicate is automatically generated by automation script as a fact from the CSV transcript. It denotes that "Student has taken this Course".

Example: _hasTaken(hardik,cs6363). %hardik has taken cs6363

hasNotTaken(Student,Course).

This predicate encodes CWA. If we don't know that student has taken this course, this predicate is true.

_t(Student,Course).

This predicate creates an even loop so that it will always be true regardless whether or not the student has taken the course. If the student has not taken the course, the answer set will show the hasNotTaken predicate in the answer set. If the student has taken the course, it will hide the _hasTaken and _t predicates.

_meetsHours(Student,Req,Min)

This predicate will count the amount of credit hours a Student has taken of a certain Req and will be true if the amount of credit hours is greater than the Min.

Example:
_meetsHours(Student,required,15) :- 15 credit hours of core courses is required.
_meetsHours(Student,elective,18) :- 18 credit hours of elective and prerequisite is required.

_auditHours(Student,Req,Min).

Semantic of this is same as meetsHours except it will always be true and if the total hours is less than the Min, needsHours(Student,Req,Diff) will appear in the answer set, with Diff as the hours needed in the requirement.

Example:
needsHours(hardik,elective,6):- 6 credit hours of elective and/or prerequisite is needed.
needsHours(hardik,required,3):- 3 credit hours of core is needed.

Exact flow of the system is described using one test case in readme file.

Assumption made for the projects (closely matches with graduation requirement at UTDallas):

1. Irrespective of the track, student has to complete 15 core subject credit hours and 18 elective credit hours. Each of the track has its own list of core and elective course structure as well as prerequisite.
   e.g Data Science track has predefined 5 core course, student have no choice for core subject, while Interactive Computing Track has only 2 course predefined as core, student have choice to select 3 among 5 other core course,

2. Student have to maintain 3.18 GPA for core subject of its track, while 3.0 GPA for non-core subjects. If student haven't maintain the above GPA, program will add the predicate hasNotTaken(Student,Course) where course would be from other track that would basically advice to change the track.

3. If the transcript is partial, e.g student haven't taken sufficient credit hours for its track to graduate, it will suggest to take particular course either core, elective or prerequisites, using hasNotTaken(X,Y) predicate in answer set.

Issue faced during implantations:

1. Transcript parsing from pdf format to csv automatically, we were using ghostscrit python library, where the script couldn't find the required functions to convert the pdf format.
   Solution: we have made csv file for few students manually.

2. There are many assumptions that needs to be encoded using rule, we found it difficult to implement the choice of selecting core subject among available core electives.
   Solution: we have tested few student transcripts with just one core subject from available core electives.

Potential Improvement:

1. Overlapping tracks: Assumption number 2 is partially implemented but rest is the future scope. e.g student can change its track based on combination of core and elective courses it has taken. Major work would be to convert directory structure to flat format.

2. Logic rules for the transfer student and FastTrack student category is yet to encode in the system.