REG-EXPERT: A Knowledge-Based Intelligent Agent for Course Registration

İmren Toprak

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the Degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
February, 2015
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Serhan Çiftçioğlu
Acting Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Assoc. Prof. Dr. Zeki Bayram
Supervisor

Examining Committee

1. Prof. Dr. Marifi Güler
2. Assoc. Prof. Dr. Zeki Bayram
3. Asst. Prof. Dr. Önsen Toygar

ABSTRACT

In this study, we present the design and implementation of a knowledge-based intelligent software agent for helping advisors and students in the choice of courses during the course registration period every semester in a university setting. The agent has knowledge of university rules and regulations, the curricula in the university, prerequisite information about courses, as well as information about students, including their academic history. Once the courses are selected, the agent then finds an optimal schedule that minimizes the number of clashes during the week by considering different sections of the courses.

Keywords: University, Course Registration, Software Intelligent Agent, Knowledge-Based.

ÖZ

Bir üniversite ortamında, her yarı yıl başında ders kayıt dönemi sürecinde ders seçme konusunda danışmanlara ve öğrencilere yardım amaçlı bir uygulama tasarladık. Bu uygulama, bilgi tabanlı akıllı bir yazılım etmenidir. Bu etmen, üniversitenin kuralları ve düzenlemeleri hakkında bilgiye sahiptir. Üniversitenin müfredatı, derslerin önkoşul düzenlemesi bu kurallara örnek verilebilir. Ayrıca, etmenimiz akademik geçmişleri dahil olmak üzere öğrenciler hakkında bilgiye de sahiptir. Dersler seçildikten sonra, akıllı etmen bir derse ait diğer grupları ve aynı dönemde seçilmiş diğer ders gruplarını göz önünde bulundurarak, mümkün olduğunca çakışmasız veya çakışan saat sayısı en düşük seviyede olan ders programını bulmaktadır.

Anahtar Kelimeler: Üniversite, Ders Programı, Akıllı yazılım ajanı, Bilgi tabanlı.

To My Father: Mustafa Toprak

To My Mother: Fatma Toprak

To My Sister: Semra Toprak

To My Sister: Meryem Toprak

To My Brother: Kader Toprak

To My Little Princess: Sude Toprak

ACKNOWLEDGMENT

Foremost, I would like to express my special appreciation and thanks to my supervisor Assoc. Prof. Dr. Zeki Bayram for his help, motivation, supervision, support and guidance. He has enriched me with his huge knowledge and experience.

My sincere thanks to my family. Words cannot express how gratefull I am to my father Mustafa Toprak, to my mother Fatma Toprak and to my sisters and to my brother for their support throughout my whole life generally and for my education life specifically.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Registering students to courses in a university setting without violating any of the university rules is a significant challenge, given the complexity of those rules and regulations as they evolve over time. In order to do the job of assigning courses to a student in the proper way, advisors have to constantly consult curricula, documentation about university rules and regulations, as well as specific students' academic histories of which courses they have taken and the grades with which they passed the courses etc. Furthermore, once the courses have been decided upon, the number of clashes in the students' timetables should be minimized through consideration of different sections opened for each course. It is clear that these activities place a heavy cognitive burden on the advisor, and often mistakes happen inadvertently.

In this thesis, we describe the design and implementation of an intelligent, Frame-logic based software agent (REG-EXPERT) that can tackle this complex registration problem in the computer engineering department at Eastern Mediterranean University (North Cyprus) [1]. REG-EXPERT is designed to help advisors, as well as students in the preregistration period in the task of course registration each semester. It is implemented in Flora-2 [2], a sophisticated frame-logic based system with transactional capabilities. All information needed for the operation of REG-EXPERT, such as student data, curricula for programs taught in the department, specific course information, as well as transcript data is stored in the internal knowledge base of Flora-

2 in the form of frame-based logic statements. Implicit information, such as student grade point average (GPA) and cumulative grade point average (CGPA) are computed automatically through logic rules defining attributes of objects.

REG-EXPERT has three main components. The first component (transcript computation) prepares a transcript for a student, given the raw data of the courses the student has taken previously, as well as the grades obtained in these courses. This module also determines a student's status, which is a function of the student's current academic term and CGPA. The status determines how many new courses (if any) a student can take. The second module (course finder) is the heart of the agent. It considers the program in which the student is enrolled (which determines the curriculum the student is following), the status of the student, the transcript of the student, and the courses offered in the current semester, and makes a suggestion for a full load of courses for the student, making sure that all relevant university rules and regulations are applied. The last module (optimization) takes the suggestion of the course finder module and finds a timetable with the minimum number of clashes by considering different sections of each course the student should take.

The remainder of this thesis is organized as follows. In chapter 2, we describe the relevant rules and registrations at Eastern Mediterranean University that must be obeyed when registering a student, as well as the rules for determination of student status and GPA/CGPA computations. Chapter 3 contains information about the Flora-2 logic system and the course registration problem in general. The architecture, design and relevant implementation details of REG-EXPERT are presented in chapter 4. Chapter 5 contains a realistic registration scenario for a fictionary computer engineering student that demonstrates the workings of the agent. Chapter 6 contains related work (other course registration assistants, optimization, advising). Finally, in chapter 7 we have the conclusion and future research directions for further enhancement of REG-EXPERT. We also talk about the improvements that can be made to REG-EXPERT in terms of user-interface and web connectivity to make it usable on a large scale.

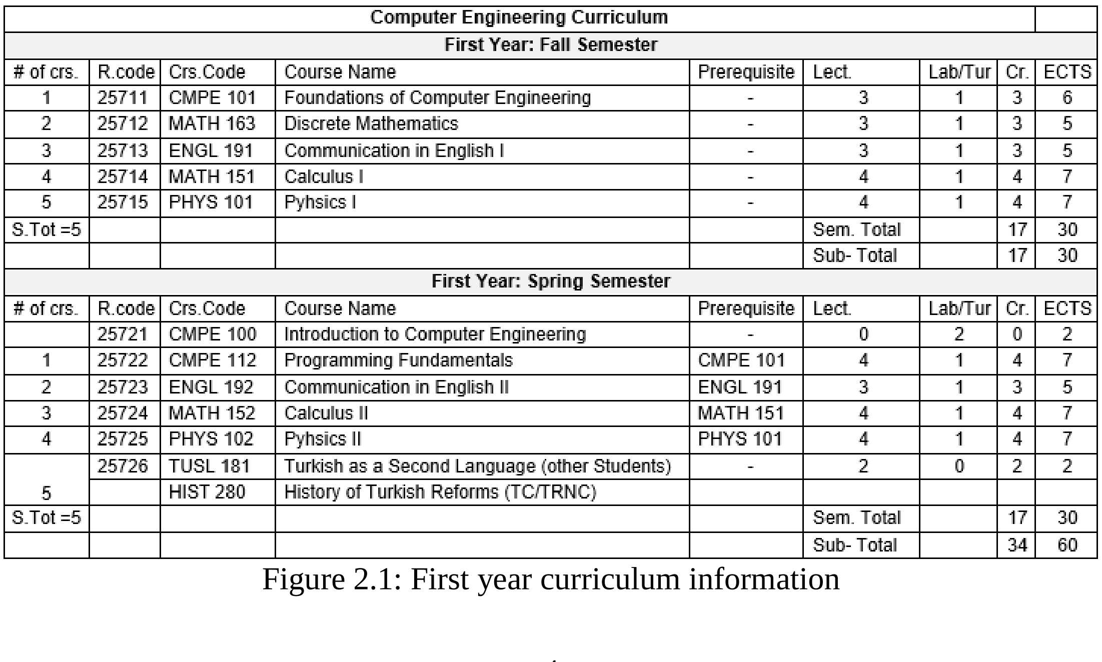
Chapter 2

RULES AND REGULATIONS OF EMU RELEVANT TO REGISTRATION

In this chapter we give explanations of rules and regulations relevant to registration and general information about curriculum of the department of computer engineering.

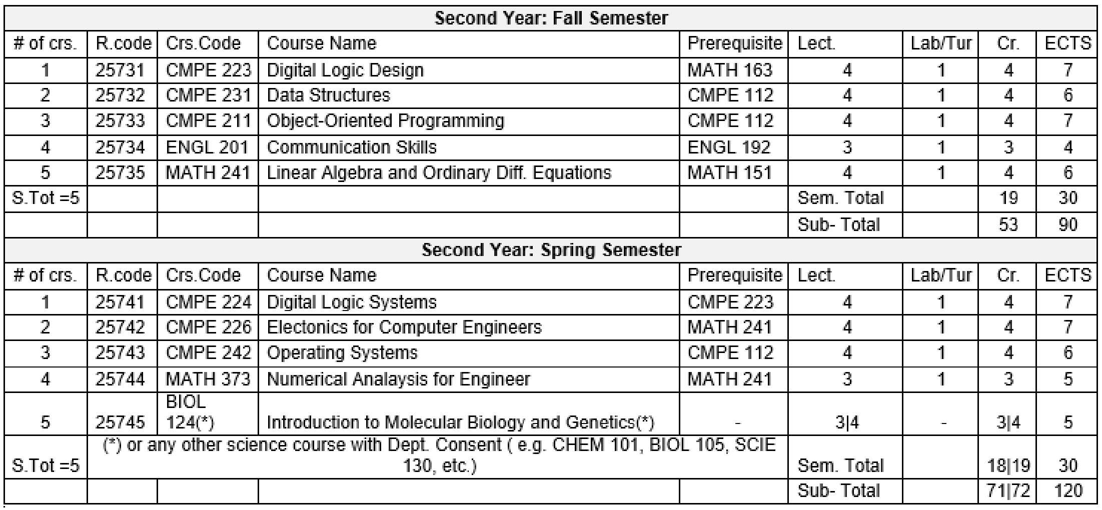
2.1 Computer Engineering Curriculum

There are forty three courses in computer engineering curriculum. Forty which have 3 or 4 credits (to be called full-credited courses from now on), one is one credited and two are non credited courses. Curriculum of computer engineering contains eight semesters. Five 3 or 4 credited courses appear in every semester. The summer training course as well as the "introduction to computer engineering" course are zero credited. The first part of the graduation project course has one credit. Details of the 4 year curriculum are shown in the figures 2.1 to 2.4 [1].



Computer Engineering Curriculum								
First Year: Fall Semester								
# of crs.	R.code	Crs.Code	Course Name	Prerequisite	Lect.	Lab/Tur	Cr.	ECTS
1	25711	CMPE 101	Foundations of Computer Engineering	-	3	1	3	6
2	25712	MATH 163	Discrete Mathematics	-	3	1	3	5
3	25713	ENGL 191	Communication in English I	-	3	1	3	5
4	25714	MATH 151	Calculus I	-	4	1	4	7
5	25715	PHYS 101	Pyhsics I	-	4	1	4	7
S.Tot =5					Sem. Total		17	30
					Sub- Total		17	30
First Year: Spring Semester								
# of crs.	R.code	Crs.Code	Course Name	Prerequisite	Lect.	Lab/Tur	Cr.	ECTS
	25721	CMPE 100	Introduction to Computer Engineering	-	0	2	0	2
1	25722	CMPE 112	Programming Fundamentals	CMPE 101	4	1	4	7
2	25723	ENGL 192	Communication in English II	ENGL 191	3	1	3	5
3	25724	MATH 152	Calculus II	MATH 151	4	1	4	7
4	25725	PHYS 102	Pyhsics II	PHYS 101	4	1	4	7
	25726	TUSL 181	Turkish as a Second Language (other Students)	-	2	0	2	2
5		HIST 280	History of Turkish Reforms (TC/TRNC)					
S.Tot =5					Sem. Total		17	30
					Sub- Total		34	60

Figure 2.1: First year curriculum information



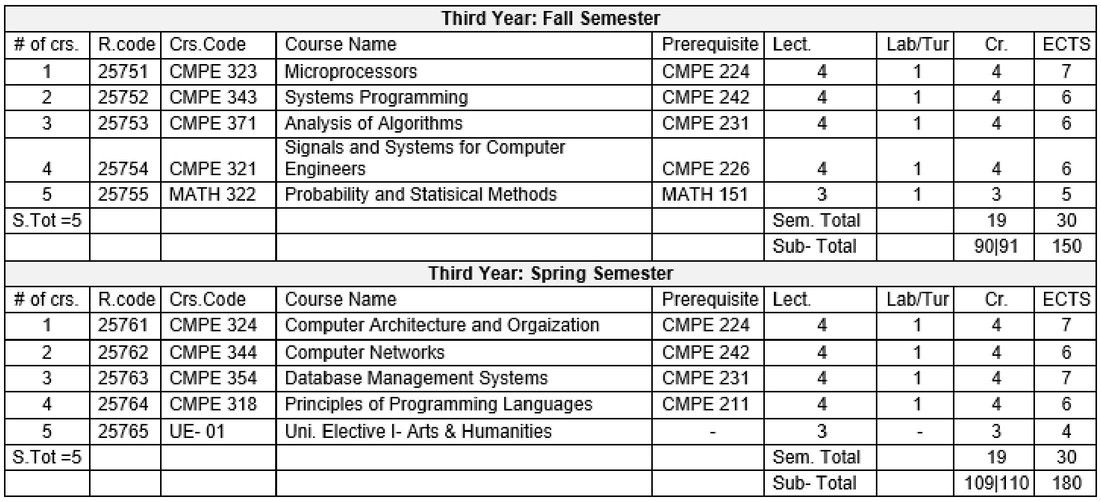
Second Year: Fall Semester

# of crs.	R.code	Crs.Code	Course Name	Prerequisite	Lect.	Lab/Tur	Cr.	ECTS
1	25731	CMPE 223	Digital Logic Design	MATH 163	4	1	4	7
2	25732	CMPE 231	Data Structures	CMPE 112	4	1	4	6
3	25733	CMPE 211	Object-Oriented Programming	CMPE 112	4	1	4	7
4	25734	ENGL 201	Communication Skills	ENGL 192	3	1	3	4
5	25735	MATH 241	Linear Algebra and Ordinary Diff. Equations	MATH 151	4	1	4	6
S.Tot =5					Sem. Total		19	30
					Sub- Total		53	90

Second Year: Spring Semester

# of crs.	R.code	Crs.Code	Course Name	Prerequisite	Lect.	Lab/Tur	Cr.	ECTS
1	25741	CMPE 224	Digital Logic Systems	CMPE 223	4	1	4	7
2	25742	CMPE 226	Electonics for Computer Engineers	MATH 241	4	1	4	7
3	25743	CMPE 242	Operating Systems	CMPE 112	4	1	4	6
4	25744	MATH 373	Numerical Analaysis for Engineer	MATH 241	3	1	3	5
5	25745	BIOL 124(*)	Introduction to Molecular Biology and Genetics(*)	-	3|4	-	3|4	5
S.Tot =5	(*) or any other science course with Dept. Consent (e.g. CHEM 101, BIOL 105, SCIE 130, etc.)				Sem. Total		18|19	30
					Sub- Total		71|72	120

Figure 2.2: Second year curriculum information



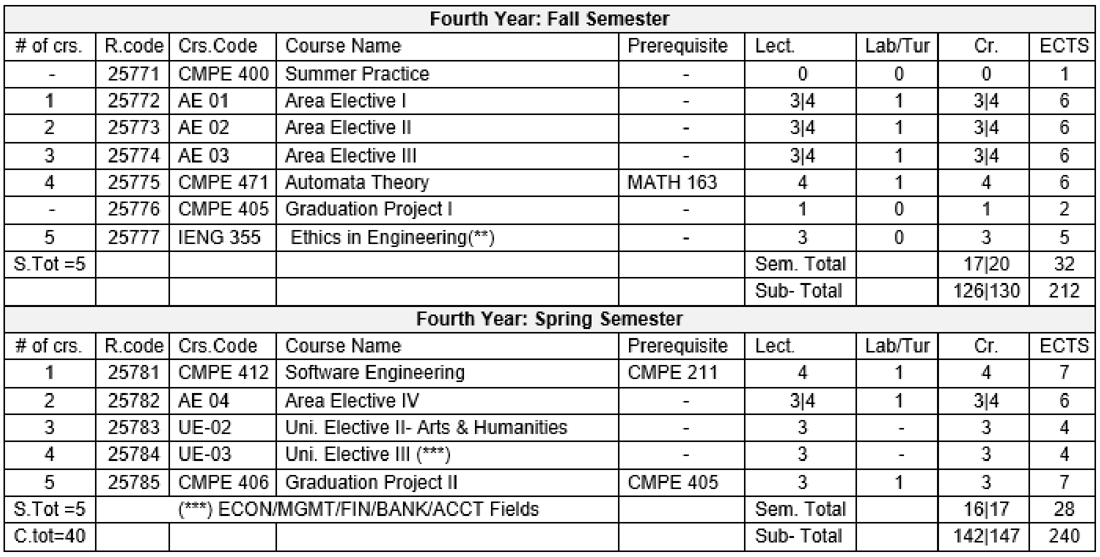
Third Year: Fall Semester

# of crs.	R.code	Crs.Code	Course Name	Prerequisite	Lect.	Lab/Tur	Cr.	ECTS
1	25751	CMPE 323	Microprocessors	CMPE 224	4	1	4	7
2	25752	CMPE 343	Systems Programming	CMPE 242	4	1	4	6
3	25753	CMPE 371	Analysis of Algorithms	CMPE 231	4	1	4	6
4	25754	CMPE 321	Signals and Systems for Computer Engineers	CMPE 226	4	1	4	6
5	25755	MATH 322	Probability and Statisical Methods	MATH 151	3	1	3	5
S.Tot =5					Sem. Total		19	30
					Sub- Total		90|91	150

Third Year: Spring Semester

# of crs.	R.code	Crs.Code	Course Name	Prerequisite	Lect.	Lab/Tur	Cr.	ECTS
1	25761	CMPE 324	Computer Architecture and Orgaization	CMPE 224	4	1	4	7
2	25762	CMPE 344	Computer Networks	CMPE 242	4	1	4	6
3	25763	CMPE 354	Database Management Systems	CMPE 231	4	1	4	7
4	25764	CMPE 318	Principles of Programming Languages	CMPE 211	4	1	4	6
5	25765	UE- 01	Uni. Elective I- Arts & Humanities	-	3	-	3	4
S.Tot =5					Sem. Total		19	30
					Sub- Total		109|110	180

Figure 2.3: Third year curriculum information



Fourth Year: Fall Semester								
# of crs.	R.code	Crs.Code	Course Name	Prerequisite	Lect.	Lab/Tur	Cr.	ECTS
-	25771	CMPE 400	Summer Practice	-	0	0	0	1
1	25772	AE 01	Area Elective I	-	3|4	1	3|4	6
2	25773	AE 02	Area Elective II	-	3|4	1	3|4	6
3	25774	AE 03	Area Elective III	-	3|4	1	3|4	6
4	25775	CMPE 471	Automata Theory	MATH 163	4	1	4	6
-	25776	CMPE 405	Graduation Project I	-	1	0	1	2
5	25777	IENG 355	Ethics in Engineering(**)	-	3	0	3	5
S.Tot =5					Sem. Total		17|20	32
					Sub- Total		126|130	212
Fourth Year: Spring Semester								
# of crs.	R.code	Crs.Code	Course Name	Prerequisite	Lect.	Lab/Tur	Cr.	ECTS
1	25781	CMPE 412	Software Engineering	CMPE 211	4	1	4	7
2	25782	AE 04	Area Elective IV	-	3|4	1	3|4	6
3	25783	UE-02	Uni. Elective II- Arts & Humanities	-	3	-	3	4
4	25784	UE-03	Uni. Elective III (***)	-	3	-	3	4
5	25785	CMPE 406	Graduation Project II	CMPE 405	3	1	3	7
S.Tot =5		(***) ECON/MGMT/FIN/BANK/ACCT Fields			Sem. Total		16|17	28
C.tot=40					Sub- Total		142|147	240

Figure 2.4: Fourth year curriculum information

2.1.1 Turkish versus Foreign Students

The curriculum entry with reference code is 25726 has two options of courses. One of them is history course (HIST280) and the second one is Turkish course (TUSL181). Students who have nationality of Republic of Turkey and Turkish Republic of Nothern Cyprus (TRNC) take the history course. Students from other countries have to take the Turkish course.

2.1.2 Restricted Electives

There are three restricted elective courses whose reference codes are 25745, 25777 and 25784. For reference code of 25745 students can take only science courses such as chemistry or biology. For reference code 25777 students can take only ethics courses. For reference code of 25784 students can take only business courses that are offered in that semester, such as economy and management.

2.1.3 Area Electives

Area elective courses offered by the computer engineering department are announced by the department at the beginning of each semester. There are four reference codes (25772, 25773, 25774 and 25782) that must be taken as area elective courses. Students can take whatever course they want that are offered as a technical elective in that semester.

2.1.4 Summer Training

Every student should do summer practice for 40 working days in a company in order to graduate from computer engineering department. While doing summer training, there are rules and regulations that every student should follow. Course code of summer training is CMPE400 and it's reference code is 25771.

2.1.5 Graduation Project

Every student must do a graduation project in order to graduate from computer engineering department. Graduation project is divided into two semesters that has course codes of CMPE405 and CMPE406 which have reference codes of 25776 and 25785 respectively. CMPE405 is prerequisite of CMPE406 and it is one credited. Therefore, every student can take CMPE405 course as a sixth course. However, CMPE406 is three credited, so it cannot be taken as a sixth course.

2.2 Grade Point Average (GPA) – Cumulative Grade Point Average (CGPA)

The Grade Point Average (GPA) is a student's academic achievement for each semester and is expressed numerically by a real number. In order to compute GPA, firstly, credits earned for each course should be calculated. It is computed by the formula:

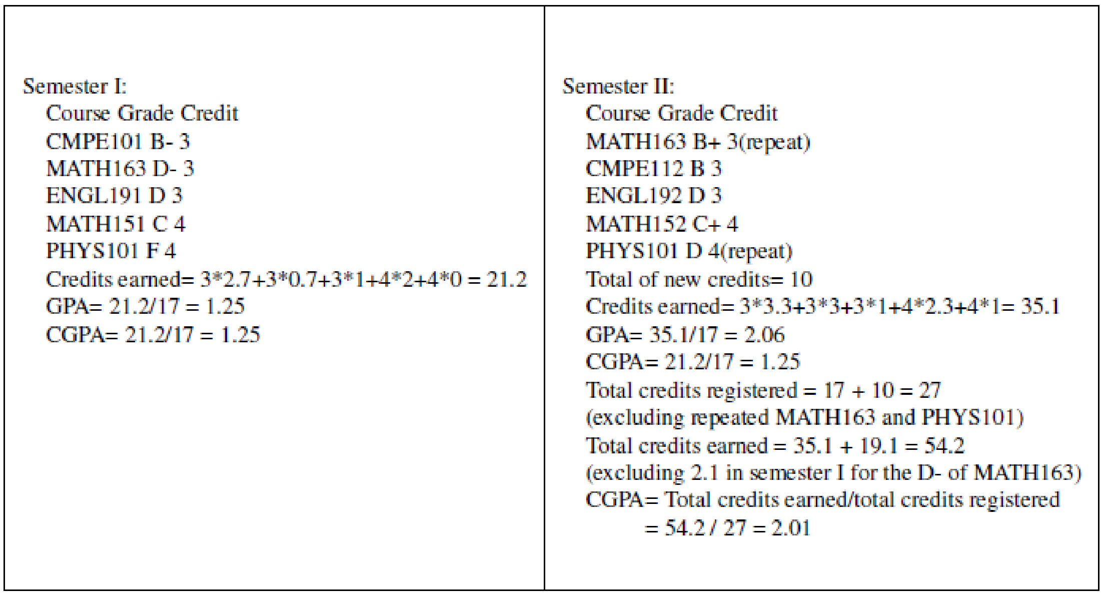
$$\frac{\sum_{i\,\in\,courses\ taken\ during\ the\ semester} c_i \times g_i}{\sum_{i\,\in\,courses\ taken\ during\ the\ semester} c_i}$$

In the above formula, c_i represents the credit of i-th course and g_i represents the numeric equivalent of the grade obtained for the i-th course. Numeric equivalents of letter grades are as follows: A = 4, A- = 3.7, B+ = 3.3, B = 3, B- = 2.7, C+ = 2.3, C = 2, C- = 1.7, D+ = 1.3, D = 1, D- = 0.7 and F = 0.

The cumulative grade point average (CGPA) is a student's overall academic achievement is expressed by a real number. It is given by the formula:

$$\frac{\sum_{i \in all\ courses\ taken} c_i \times g_i}{\sum_{i \in all\ courses\ taken} c_i}$$

In the above formula, c_i and g_i have the same meaning as in the computation of the GPA. Illustration of GPA and CGPA computations are given in figure 2.5. Suppose a student complete two semester.



Semester I:
Course Grade Credit
CMPE101 B- 3
MATH163 D- 3
ENGL191 D 3
MATH151 C 4
PHYS101 F 4
Credits earned= 3*2.7+3*0.7+3*1+4*2+4*0 = 21.2
GPA= 21.2/17 = 1.25
CGPA= 21.2/17 = 1.25

Semester II:
Course Grade Credit
MATH163 B+ 3(repeat)
CMPE112 B 3
ENGL192 D 3
MATH152 C+ 4
PHYS101 D 4(repeat)
Total of new credits= 10
Credits earned= 3*3.3+3*3+3*1+4*2.3+4*1= 35.1
GPA= 35.1/17 = 2.06
CGPA= 21.2/17 = 1.25
Total credits registered = 17 + 10 = 27
(excluding repeated MATH163 and PHYS101)
Total credits earned = 35.1 + 19.1 = 54.2
(excluding 2.1 in semester I for the D- of MATH163)
CGPA= Total credits earned/total credits registered
= 54.2 / 27 = 2.01

Figure 2.5: Example of GPA and CGPA Computation

2.3 Prerequisite Courses

Some courses have prerequisite course. When course X is a prerequisite of Y, student cannot take course Y without obtaining at least a D- grade from X.

2.4 Course Load Rules

The semester course load is defined as the number of credit courses for which a student is registered in a semester. The regular course load for computer engineering students is 5 credited courses. This load can increase by at most one credited course, if he/she has a GPA and CGPA that are greater than 3.00. This load can be decreased at most two credited courses. If a student is in the last academic semester (graduation semester), in other words if he/she is left with 7 courses in last semester, he/she can register to all 7 credited courses with the approval his/her of advisor and chairman of the department. A student can take at most 2 credited courses in the summer session.

2.5 Student Status

Student can pass a course successfully, if he/she gets a grade of A, A-, B+, B, B-, C+ ,C , C-, D+, D or S. Otherwise, he/she will be unsuccessful from that course with grade of D-, F or U and the student must repeat the course if it is offered the following semester. If a student passes the course with grade of D, he/she may repeat the course in some situations. A student may have different type of status according to his/her CGPA.

There is a CGPA expectation from a student at every actual academic term excluding the first two semesters. Actual academic term refers to the number of semesters that a student has registered so far, excluding English preparatory school and summer semesters.

There are four different type of status. First one is *satisfactory* which means that student has sufficient success. In order to have satisfactory status, student's CGPA must be greater that 1.50 out of 4 from the end of second actual academic term until the end of the fourth actual academic term. Also, CGPA must be greater than 1.80 out of 4 from end of the fifth actual academic term until the end of the seventh actual academic term to have satisfactory status. In addition to this, CGPA must be greater than 2.0 out of 4 at the end of the the eight actual academic term and above to be in satisfactory status. If a student’s status is satisfactory, he/she can register five new credited courses in the following semester.

Secondly, there is a status of *on probation*. A student’s status will be on probation if his/her CGPA is between 1.50 and 1.0 in the end of second actual academic term until the end of the fourth actual academic term, if his/her CGPA is between 1.80 and 1.50 at the end of the fifth actual academic term until the end of the seventh actual academic term and if his/her CGPA is between 2.0 and 1.80 at the end of the eight actual academic term and above. In on probation status, a student can register only two new credited courses in the following semester.

Another status is *unsatisfactory*. A student’s status will be unsatisfactory if his/her CGPA is between 1.0 and 0.0 at the end of the second actual academic term until the end of the fourth actual academic term, if his/her CGPA is between 1.5 and 0.0 at the end of the fifth actual academic term until the end of the seventh actual academic term and if his/her CGPA is between 1.80 and 0.00 at the end of the eighth actual academic term and above. If a student’s status is unsatisfactory, he/she cannot register to any new credited courses in the following semester.

Finally, there is status of *compulsory transfer*, in other words dismiss which means student should transfer to another department. If a student’s CGPA is between 1.0 and

0.0 after third actual academic term, his/her status will be compulsory transfer. These rules are tabulated in table 1 [3].

Table 1: Satisfactory / On Probation / Unsatisfactory

End of Actual Academic Term	Satisfactory (S)	Satisfactory Progress (Y)	On Probation (P)	Unsatisfactory (U)	Compulsory Transfer/DISMISS ©
2	4.00≥CGPA≥2.00	2.00>CGPA≥1.50	1.50>CGPA≥1.00	1.00>CGPA≥0.00	-----
3	4.00≥CGPA≥2.00	2.00>CGPA≥1.50	1.50>CGPA≥1.00	1.00>CGPA≥0.00	-----
4	4.00≥CGPA≥2.00	2.00>CGPA≥1.50	1.50>CGPA≥1.00	1.00>CGPA≥0.00	***1.00>CGPA≥0.00
5	4.00≥CGPA≥2.00	2.00>CGPA≥1.80	1.80>CGPA≥1.50	1.50>CGPA≥0.00	***1.00>CGPA≥0.00
6	4.00≥CGPA≥2.00	2.00>CGPA≥1.80	1.80>CGPA≥1.50	1.50>CGPA≥0.00	***1.00>CGPA≥0.00
7	4.00≥CGPA≥2.00	2.00>CGPA≥1.80	1.80>CGPA≥1.50	1.50>CGPA≥0.00	***1.00>CGPA≥0.00
≥8	4.00≥CGPA≥2.00	-----	2.00>CGPA≥1.80	1.80>CGPA≥0.00	***1.00>CGPA≥0.00

2.6 Extra Course

Generally, course load of every student is five full-credited courses in every semester. However, this may vary in some situations. A student who obtains a GPA between 3.0 and 3.49 with a normal course load is designated an honor student. A student who obtains a GPA between 3.50 and 4.00 with a normal course load is designated a high honor student. If a student's GPA and CGPA is higher than 3.00, he/she can register to one extra full-credited courses. Also, if a student left with seven credited courses in the academic term, this means that this academic term is graduation term for this student. Therefore, he/she can register to all seven credited courses with the approval of his/her advisor and the chairman of the department.

2.7 Rules for Registration

A student who starts regularly to the department will register his/her courses as given in the curriculum. Advisor of a student should be careful about rules of registering the courses. If a student's status is sufficient, he/she may give the courses that appear in the curriculum accordingly. Order of registering the courses as follows. Firstly, advisor should register the courses that are failed if they offered in the new semester. Secondly, the courses which are withdrawed in the last semester should be registered in the new semester. Courses from previous academic semester should be registered. After these have been taken care of, the advisor should be careful about status of the student. If a student's status is satisfactory, new courses may be taken. However, if a student's status is on probation, only two new courses may be taken and if a student's status is unsatisfactory, no new courses can be taken.

2.8 Rules for Compulsory Transfer / Dismiss

At the end of fourth academic semester, if a student has CGPA below 1.00, he/she will be dismissed from department of computer engineering. In this status, student should transfer to another faculty or he/she can continue computer engineering department with new student registration fees.

Chapter 3

BACKGROUND

In this chapter, we provide information about intelligent agents, knowledge-based systems, university automation systems and Flora-2.

3.1 Intelligent Agent

Firstly, we should give definition of an agent in order to describe the intelligent agent. There are many definitions for an agent. Basically, they meet at a common point. An agent is "a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives" [4]. A software agent is "An artificial agent which operates in a software environment" [5]. An intelligent software agent is "a software agent that uses Artificial Intelligence (AI) in the pursuit of the goals of its clients" [6]. An agent percepts the input from environment by its sensors, decides on its action and gives output to the environment by its effectors. It should exhibit some form of intelligence and independent action on behalf of its user and owner. Also, it may be situated (embedded in some environment which it can sense and act), reactive (has ability to respond the sensed input), autonomous (independent and make its own decision), social (interacts with other agents), and pro-active (it has one or more goals to achieve) [4]. Furthermore, it should be rational while decide on its action and achieving its goals. It should have internal architecture so that it can be understood in terms of mentalistic notions. We can give example to mentalistic notions, beliefs, desires, intentions and obligations. Finally, an agent is adaptive. In other words, we can update the agent according to new technology or we can adapt the agent to new requirements of environment [7].

There are industrial control applications that are difficult to manage. In other words it is hard to specify all cases. In such situations agents are desirable. Since they can think and act like a human with rational mentality [8].

3.2 Knowledge-Based Systems and Artificial Intelligence (AI)

“Intelligence is the computational part of the natural ability to achieve desired goals in the world” [9]. Generally, artificial intelligence is implemented on computers with artificial simulations of human brain function. In other words, AI is a term to express an area of science and engineering whose aim is to produce computational understanding and intelligent behavior, resulting for example in robots and speaking computers. Therefore, AI systems think and act like a human. Also, they think and act rationally.

Although modern computers are highly developed computational devices, unfortunately do not have the learning skill. The field of knowledge-based systems, which is a branch of artificial intelligence, tries to change this phenomenon. Researchers of AI try to gain two main abilities to knowledge-based systems. One of them is the ability to mimic human reasoning system. Secondly, ability to learning. Knowledge-based system is a computer system that is programmed to imitate the ability of solving problem of human with help of artificial intelligence and reference of knowledge base that is related with any issue [10]. For example, consider a system that diagnose disease of patient. This system is formed with the rules of diagnosis for that disease. If we can diagnose disease of patient with this system, then we can say that this system is knowledge-based system [9].

3.3 University Automation Systems

University automation systems may contain different modules with functions that are different but important individually. These modules may be an exam scheduling, course registration, course time tabling or student information system etc. [11].

3.4 Flora-2

"Flora-2 is a knowledge base engine and a complete environment for developing knowledge intensive applications" [12]. F-logic, Hilog and Transaction logic form the language of the Flora-2 [13]. Mainly, it contains F-logic that "extends classical predicate calculus with the concepts of objects, classes and types, which are adapted from object-oriented programming" [12]. Hilog "is a programming logic with higher-order syntax, which allows arbitrary terms to appear in predicate and function position" [14]. "Transaction logic provides logical foundations for state changes and side effects in a logic programming language" [15].

Chapter 4

IMPLEMENTATION

In this chapter we describe the design and implementation of REG-EXPERT in some detail.

4.1 Architecture

Figure 4.1 depicts the overall architecture of REG-EXPERT. Raw student data in the form of courses that have already been taken by the student (semester in which the courses were taken and grades obtained) are fed into the transcript generator module that computes the grade point average (GPA) and cumulative grade point average (CGPA), as well as the status of the student determined by the university rules, for each semester. The registration advisor module considers the student's transcript, curriculum of the program the student belongs to, as well as the actual courses opened during the semester to come up with a suggestion for courses to be taken by the student. Various registration rules of the university are taken into account by this module when it generates its suggestion. Finally, the optimizer module takes the output of the registration advisor module (a list of courses) and finds a schedule consisting of course openings with a minimal number of clashes.

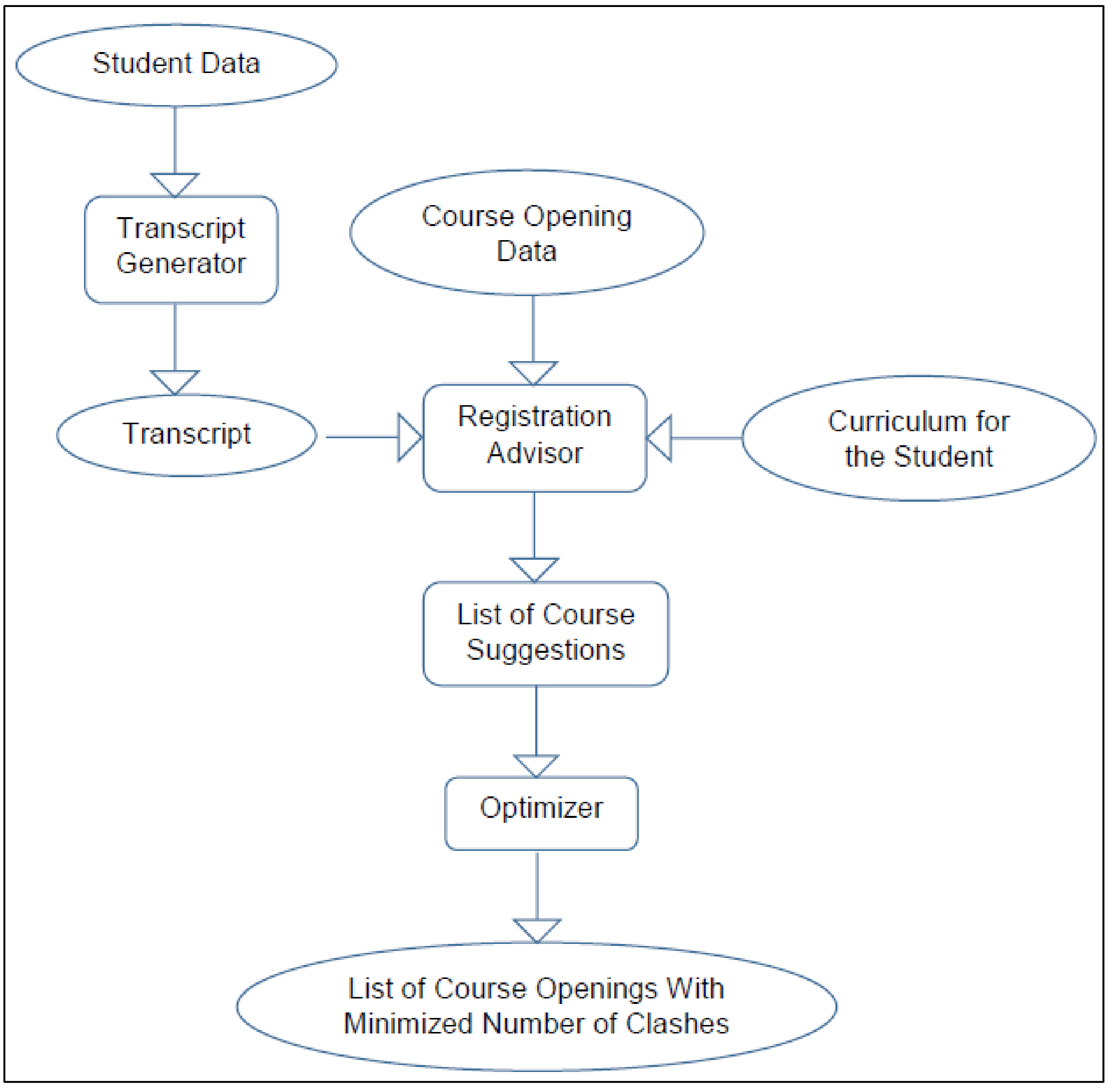

Student Data
Transcript Generator
Course Opening Data
Transcript
Registration Advisor
Curriculum for the Student
List of Course Suggestions
Optimizer
List of Course Openings With Minimized Number of Clashes

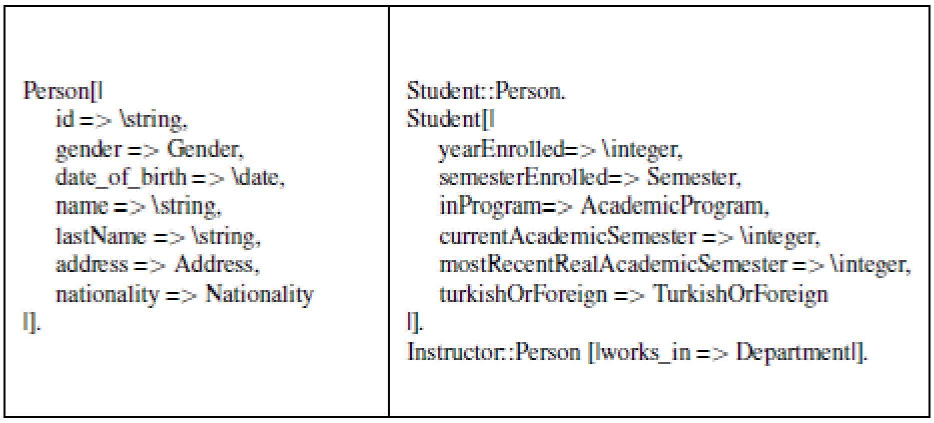
Figure 4.1: Overall architecture of REG-EXPERT

4.2 Concept Definitions

In Flora-2, concepts are similar to classes in programming language terminology. They define the internal structure of objects, and act as the knowledge-base schema. Concepts can have attributes as in classes, however, in Flora-2, attribute values can be defined by logic rules as opposed to being explicitly specified. We describe below the Concepts utilized in the implementation of REG-EXPERT.

4.2.1 Concepts Related to Students

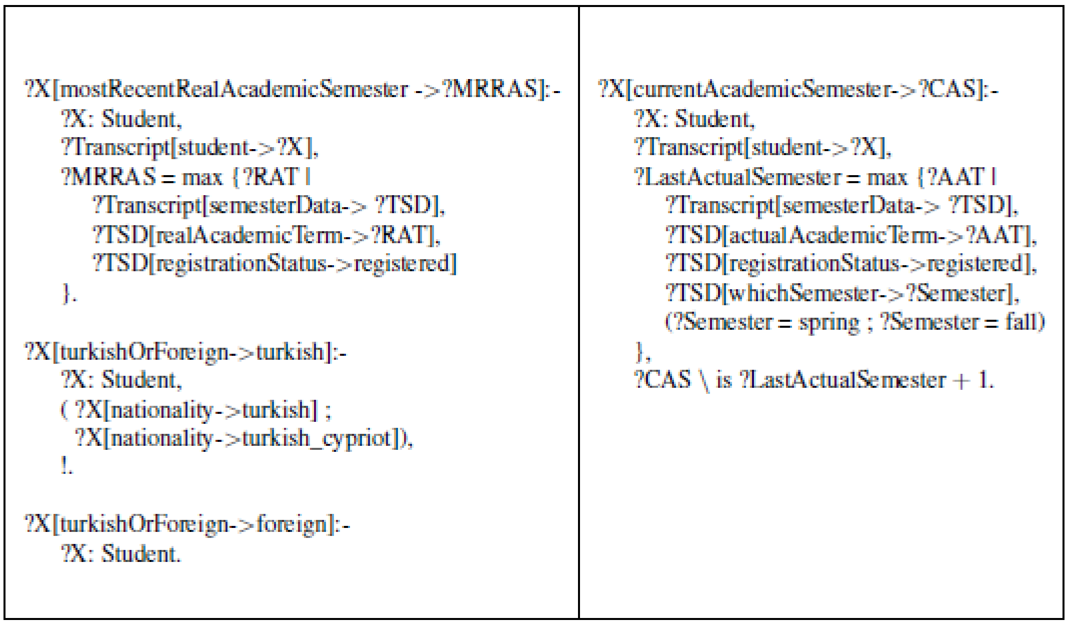
In figure 4.2 we have the *Person*, *Student* and *Instructor* concepts. The *Student* and *Instructor* concepts are sub-concepts of the *Person* concept. The notation [| |] means that a concept is inheritable. The notation *attributeName* => *someType* means that the value in *attributeName* must be of type *someType*. Built-in types start with a backslash (\), and other types are concept names, which by convention start with a capital letter. The *currentAcademicSemester* attribute contains the number of academic terms in which the student was enrolled, plus one, excluding leave of absences and summer terms. The *realAcademicTerm* attribute is similar, except it includes the summer semesters in which the student took courses. Both attributes are computed through the rules given in figure 4.3. A student's current academic term is found in the student's transcript, and is one more than the actual academic term in which s/he was registered. The student's most recent “real” academic semester is also found in the student's transcript. Also in figure 4.3 is the definition of the rule for determining the value of the *turkishOrForeign* attribute. If a student is from Turkey or North Cyprus, he/she is assumed to be Turkish; otherwise he/she is taken to be a foreigner. Foreigners take a Turkish language course, however Turkish students take a Turkish history course instead.



```
Person[|
    id => \string,
    gender => Gender,
    date_of_birth => \date,
    name => \string,
    lastName => \string,
    address => Address,
    nationality => Nationality
|].
```

```
Student::Person.
Student[|
    yearEnrolled=> \integer,
    semesterEnrolled=> Semester,
    inProgram=> AcademicProgram,
    currentAcademicSemester => \integer,
    mostRecentRealAcademicSemester => \integer,
    turkishOrForeign => TurkishOrForeign
|].
Instructor::Person [|works_in => Department|].
```

Figure 4.2: Person, Student and Instructor Concepts



```
?X[mostRecentRealAcademicSemester ->?MRRAS]:-
    ?X: Student,
    ?Transcript[student->?X],
    ?MRRAS = max {?RAT |
        ?Transcript[semesterData-> ?TSD],
        ?TSD[realAcademicTerm->?RAT],
        ?TSD[registrationStatus->registered]
    }.

?X[turkishOrForeign->turkish]:-
    ?X: Student,
    ( ?X[nationality->turkish] ;
      ?X[nationality->turkish_cypriot]),
    !.

?X[turkishOrForeign->foreign]:-
    ?X: Student.
```

```
?X[currentAcademicSemester->?CAS]:-
    ?X: Student,
    ?Transcript[student->?X],
    ?LastActualSemester = max {?AAT |
        ?Transcript[semesterData-> ?TSD],
        ?TSD[actualAcademicTerm->?AAT],
        ?TSD[registrationStatus->registered],
        ?TSD[whichSemester->?Semester],
        (?Semester = spring ; ?Semester = fall)
    },
    ?CAS \ is ?LastActualSemester + 1.
```

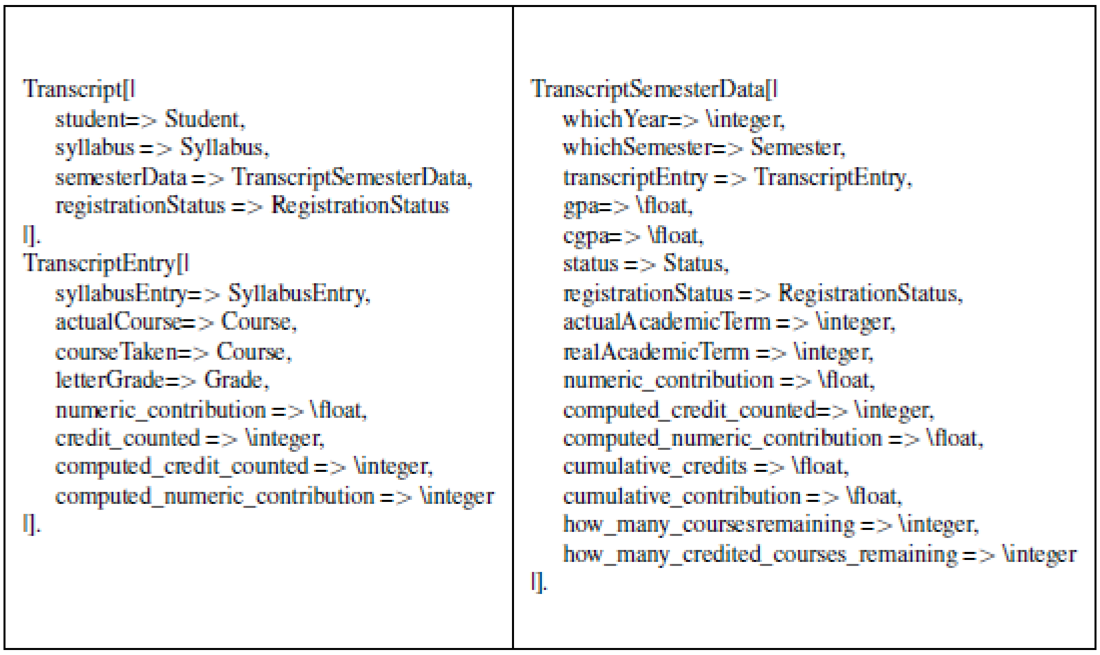
Figure 4.3: Rules defining the currentAcademicSemester, realAcademicTerm and turkishOrForeign attributes of the Student

4.2.2 Concepts Related to Transcripts

In figure 4.4 we have three concepts used in the description of a transcript. A transcript has a hierarchical structure, whereby at the top level there is the *Transcript* concept, containing references to *Student*, *Syllabus*, *Transcript-Semester-Data* and *Registration-Status* concepts. The *semester-Data* attribute is multi-valued.

Instances of *TransctiptSemesterData* contain information about a specific semester. Most of the *TransctiptSemesterData* concept attributes are computed. It has attributes for the year, semester, transcript entry, GPA, CGPA, status of the student (satisfactory, on probation or unsatisfactory) , registration status of the student (on leave of absence or registered), actual academic term of the student (how many semesters he/she spent in the university plus one, excluding the summer semesters) , real academic term (same as actual academic term except the summer terms are now included), as well as other auxiliary attributes for computing the GPA and CGPA. Its *transcript-Entry* attribute is multivalued, and contains references to *Transcript-Entry* instances.

A *Transcript-Entry* instance contains information about the course the student has taken, reference to the syllabus entry for the course, the grade obtained, and other attributes used for computing the GPA/CGPA of the student.



```
Transcript[|
    student=> Student,
    syllabus => Syllabus,
    semesterData => TranscriptSemesterData,
    registrationStatus => RegistrationStatus
|].
TranscriptEntry[|
    syllabusEntry=> SyllabusEntry,
    actualCourse=> Course,
    courseTaken=> Course,
    letterGrade=> Grade,
    numeric_contribution => \float,
    credit_counted => \integer,
    computed_credit_counted => \integer,
    computed_numeric_contribution => \integer
|].
```

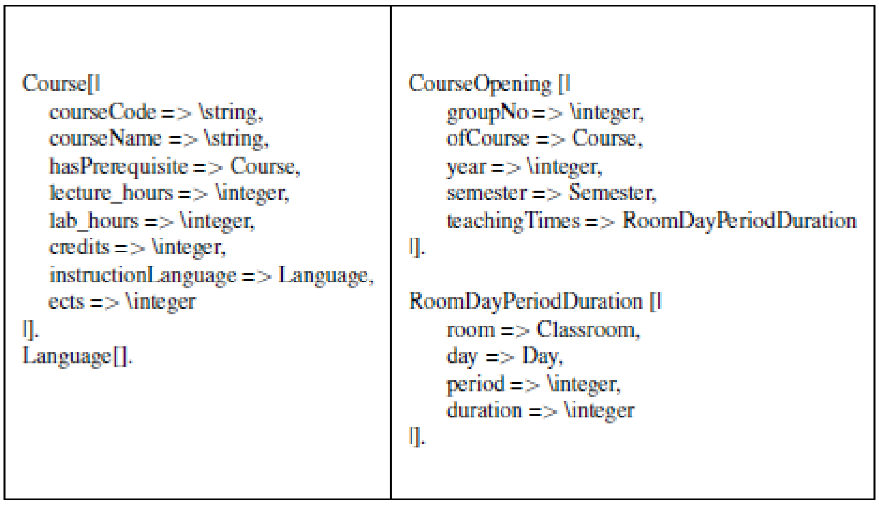
```
TranscriptSemesterData[|
    whichYear=> \integer,
    whichSemester=> Semester,
    transcriptEntry => TranscriptEntry,
    gpa=> \float,
    cgpa=> \float,
    status => Status,
    registrationStatus => RegistrationStatus,
    actualAcademicTerm => \integer,
    realAcademicTerm => \integer,
    numeric_contribution => \float,
    computed_credit_counted=> \integer,
    computed_numeric_contribution => \float,
    cumulative_credits => \float,
    cumulative_contribution => \float,
    how_many_coursesremaining => \integer,
    how_many_credited_courses_remaining => \integer
|].
```

Figure 4.4: Concepts related to transcript

4.2.3 Concepts Related to Courses

In figure 4.5 we have the concepts for defining courses and course openings. A *Course* concept contains attributes for the course code, course name, prerequisite courses, lecture hours, lab hours, credits, instruction language and European Credit Transfer System (ECTS) credits. A *CourseOpening* concept has attributes for describing a specific course offered in a specific semester. Its instances contain information about the group (section) number, the course to which it belongs, year, semester and times/places at which it is taught. *RoomDayPeriodDuration* is used to specify which room is required to teach a course, as well as the day, period and duration for which the room is required.

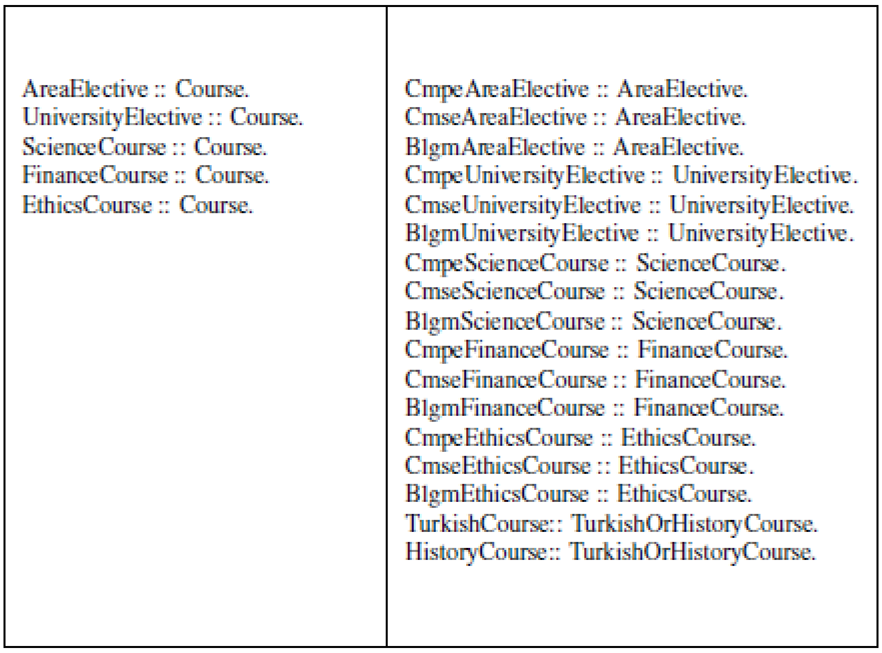
Figure 4.6 depicts the concepts related to various course types in a hierarchy. For example, *CmpeAreaElective* is an *AreaElective*.



```
Course[|
    courseCode => \string,
    courseName => \string,
    hasPrerequisite => Course,
    lecture_hours => \integer,
    lab_hours => \integer,
    credits => \integer,
    instructionLanguage => Language,
    ects => \integer
|].
Language[].
```

```
CourseOpening [|
     groupNo => \integer,
     ofCourse => Course,
     year => \integer,
     semester => Semester,
     teachingTimes => RoomDayPeriodDuration
|].

RoomDayPeriodDuration [|
     room => Classroom,
     day => Day,
     period => \integer,
     duration => \integer
|].
```

Figure 4.5: Concepts related to course



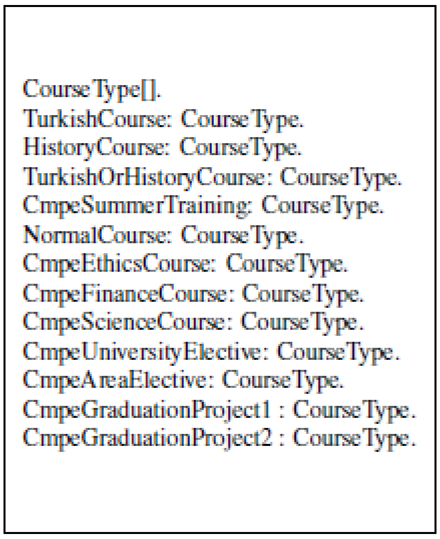
```
AreaElective :: Course.
UniversityElective :: Course.
ScienceCourse :: Course.
FinanceCourse :: Course.
EthicsCourse :: Course.
```

```
CmpeAreaElective :: AreaElective.
CmseAreaElective :: AreaElective.
BlgmAreaElective :: AreaElective.
CmpeUniversityElective :: UniversityElective.
CmseUniversityElective :: UniversityElective.
BlgmUniversityElective :: UniversityElective.
CmpeScienceCourse :: ScienceCourse.
CmseScienceCourse :: ScienceCourse.
BlgmScienceCourse :: ScienceCourse.
CmpeFinanceCourse :: FinanceCourse.
CmseFinanceCourse :: FinanceCourse.
BlgmFinanceCourse :: FinanceCourse.
CmpeEthicsCourse :: EthicsCourse.
CmseEthicsCourse :: EthicsCourse.
BlgmEthicsCourse :: EthicsCourse.
TurkishCourse:: TurkishOrHistoryCourse.
HistoryCourse:: TurkishOrHistoryCourse.
```

Figure 4.6: Hierarchy of concepts related to courses

4.2.4 Concepts that are Also Instances of the *CourseType* Concept

In figure 4.7 we have concepts that are also instances of the *CourseType* concept. Flora 2 permits this kind of usage, which is very useful in modeling certain structures. In our case, concepts used for describing course types (such as *CmpeEthicsCourse*) also appear as values of attributes inside a syllabus.

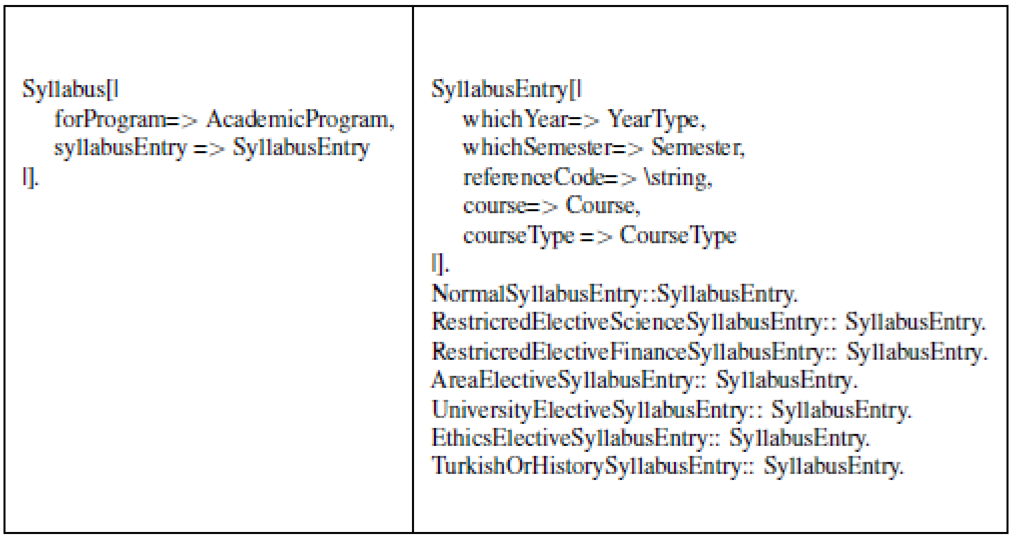


```
CourseType[].
TurkishCourse: CourseType.
HistoryCourse: CourseType.
TurkishOrHistoryCourse: CourseType.
CmpeSummerTraining: CourseType.
NormalCourse: CourseType.
CmpeEthicsCourse: CourseType.
CmpeFinanceCourse: CourseType.
CmpeScienceCourse: CourseType.
CmpeUniversityElective: CourseType.
CmpeAreaElective: CourseType.
CmpeGraduationProject1 : CourseType.
CmpeGraduationProject2 : CourseType.
```

Figure 4.7: Course type concepts

4.2.5 Concepts Related to Syllabi

Figure 4.8 contains the concepts that are used to describe syllabus objects. At the top level is the *Syllabus* concept with attributes *forProgram* and *syllabusEntry* (a multi-valued attribute). The *SyllabusEntry* concept is used to describe individual entries in a syllabus and has information such as the year (i.e. freshman, sophomore, junior or senior), the semester (i.e. fall or spring), reference code, course and type of course. The type of course determines the actual course the student is allowed to take. If the course type is *NormalCourse*, the student should take the exact course specified in the *SyllabusEntry* instance. Otherwise, s/he should take one of the courses that belong to the *courseType* attribute (e.g. if the course type is *CmpeFinanceCourse* then the student should take of of the courses in this category).



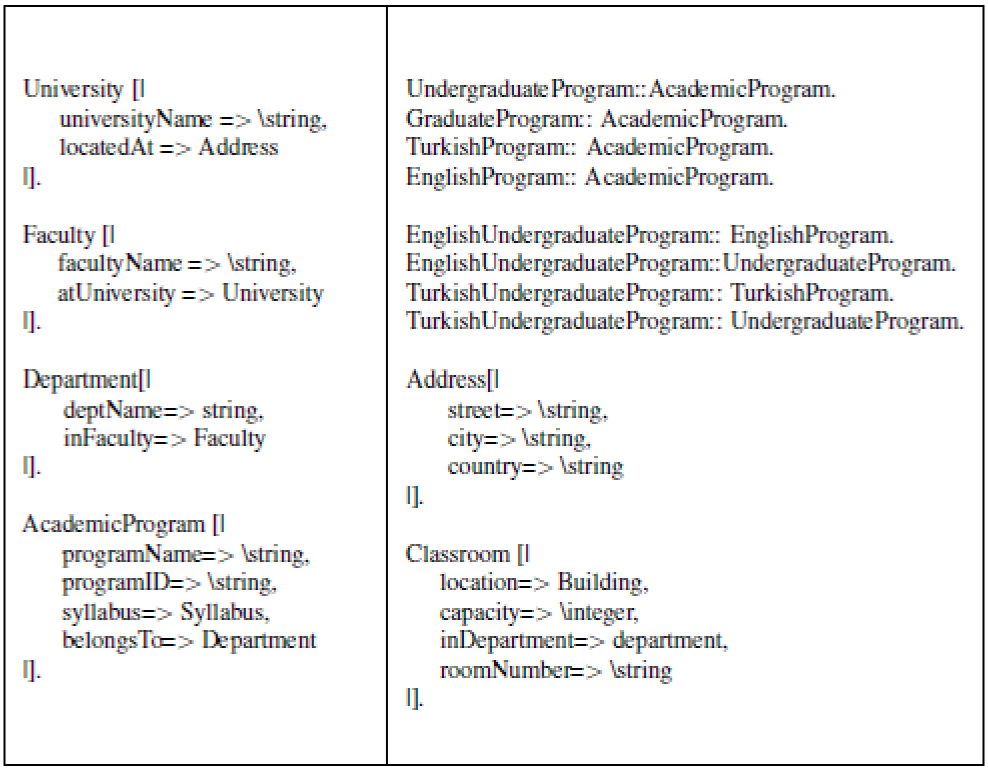
```
Syllabus[|
    forProgram=> AcademicProgram,
    syllabusEntry => SyllabusEntry
|].
```

```
SyllabusEntry[|
    whichYear=> YearType,
    whichSemester=> Semester,
    referenceCode=> \string,
    course=> Course,
    courseType => CourseType
|].
NormalSyllabusEntry::SyllabusEntry.
RestricredElectiveScienceSyllabusEntry:: SyllabusEntry.
RestricredElectiveFinanceSyllabusEntry:: SyllabusEntry.
AreaElectiveSyllabusEntry:: SyllabusEntry.
UniversityElectiveSyllabusEntry:: SyllabusEntry.
EthicsElectiveSyllabusEntry:: SyllabusEntry.
TurkishOrHistorySyllabusEntry:: SyllabusEntry.
```

Figure 4.8: Concepts related to the syllabus of a program

4.2.6 Concepts Related to the University Academic, Administrative and Physical Structure

In figure 4.9 we have the concepts for describing the hierarchical academic and administrative structure of a university, as well as its physical structure. At the top level, there is the *University* concept. Below it are faculties that house departments that run various academic programs. Each academic program implements a syllabus (in this thesis, the words “curriculum” and “syllabus” are used interchangeably, and they are both taken to mean the order and types of courses the student should take).



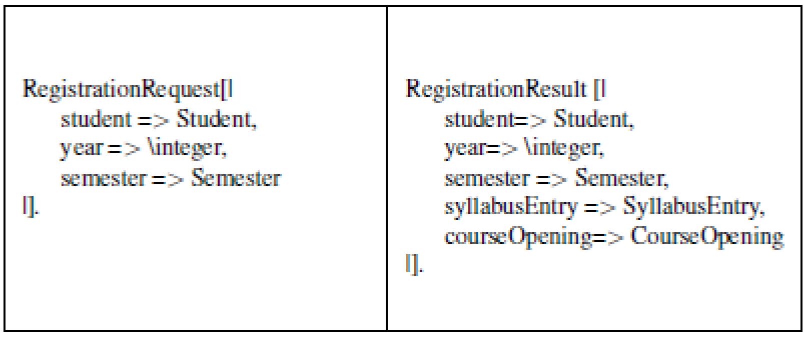
```
University [|
     universityName => \string,
     locatedAt => Address
|].

Faculty [|
     facultyName => \string,
     atUniversity => University
|].

Department[|
     deptName=> string,
     inFaculty=> Faculty
|].

AcademicProgram [|
     programName=> \string,
     programID=> \string,
     syllabus=> Syllabus,
     belongsTo=> Department
|].
```

```
UndergraduateProgram::AcademicProgram.
GraduateProgram:: AcademicProgram.
TurkishProgram:: AcademicProgram.
EnglishProgram:: AcademicProgram.

EnglishUndergraduateProgram:: EnglishProgram.
EnglishUndergraduateProgram::UndergraduateProgram.
TurkishUndergraduateProgram:: TurkishProgram.
TurkishUndergraduateProgram:: UndergraduateProgram.

Address[|
     street=> \string,
     city=> \string,
     country=> \string
|].

Classroom [|
    location=> Building,
    capacity=> \integer,
    inDepartment=> department,
    roomNumber=> \string
|].
```

Figure 4.9: University structure (administrative, academic and physical)

4.2.7 Concepts Related to Registration

Figure 4.10 contains the definition of *RegistrationRequest* and *RegistrationResult* concepts. *RegistrationRequest* contains information about the student, year and semester for which we need to make a registration. *RegistrationResult* contains information about a specific course opening which the student has been registered to. It is the job of the registration agent REG-EXPERT to recommend, given a request for registration, a list of *RegistrationResult* instances that are appropriate under the circumstances (i.e. the current status of the student, the courses the student has already taken, the courses that have been opened in the current semester, and the courses that remain). The recommendation must comply with all of the rules and regulations of the university, as well as the program and its curriculum.



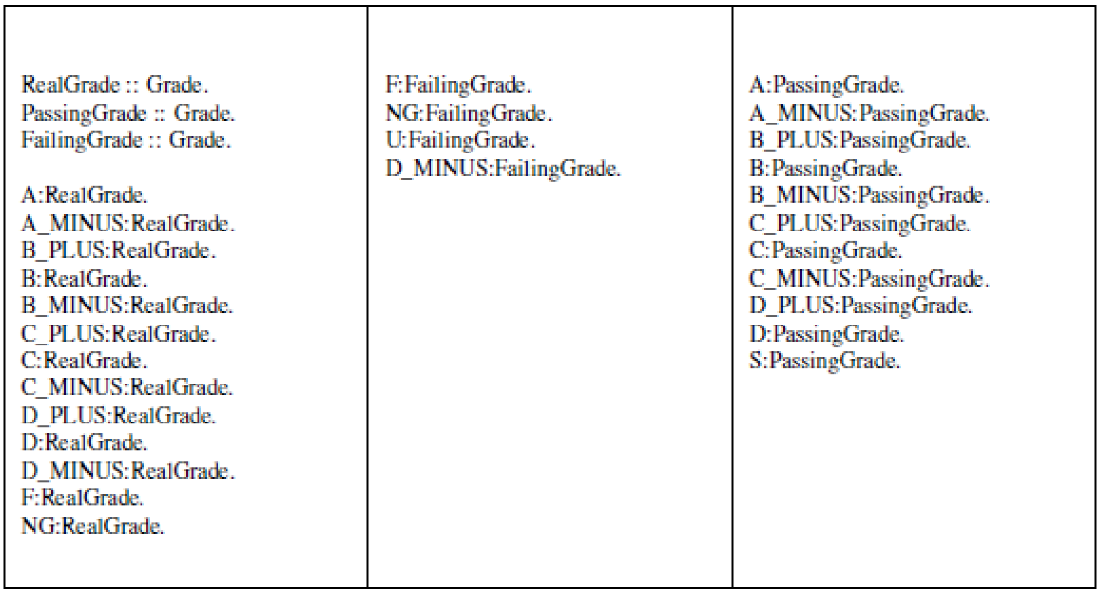
```
RegistrationRequest[|
     student => Student,
     year => \integer,
     semester => Semester
|].
```

```
RegistrationResult [|
     student=> Student,
     year=> \integer,
     semester => Semester,
     syllabusEntry => SyllabusEntry,
     courseOpening=> CourseOpening
|].
```

Figure 4.10: Concepts of registration

4.3 Classifying Grades

Figure 4.11 classifies grades according to whether they count as passing or failing, and whether they contribute towards the GPA/CGPA computations or not. *RealGrade*s contribute towards the GPA/CGPA computations whereas grades that are not *RealGrade* do not.



```
RealGrade :: Grade.
PassingGrade :: Grade.
FailingGrade :: Grade.

A:RealGrade.
A_MINUS:RealGrade.
B_PLUS:RealGrade.
B:RealGrade.
B_MINUS:RealGrade.
C_PLUS:RealGrade.
C:RealGrade.
C_MINUS:RealGrade.
D_PLUS:RealGrade.
D:RealGrade.
D_MINUS:RealGrade.
F:RealGrade.
NG:RealGrade.
```

```
F:FailingGrade.
NG:FailingGrade.
U:FailingGrade.
D_MINUS:FailingGrade.
```

```
A:PassingGrade.
A_MINUS:PassingGrade.
B_PLUS:PassingGrade.
B:PassingGrade.
B_MINUS:PassingGrade.
C_PLUS:PassingGrade.
C:PassingGrade.
C_MINUS:PassingGrade.
D_PLUS:PassingGrade.
D:PassingGrade.
S:PassingGrade.
```

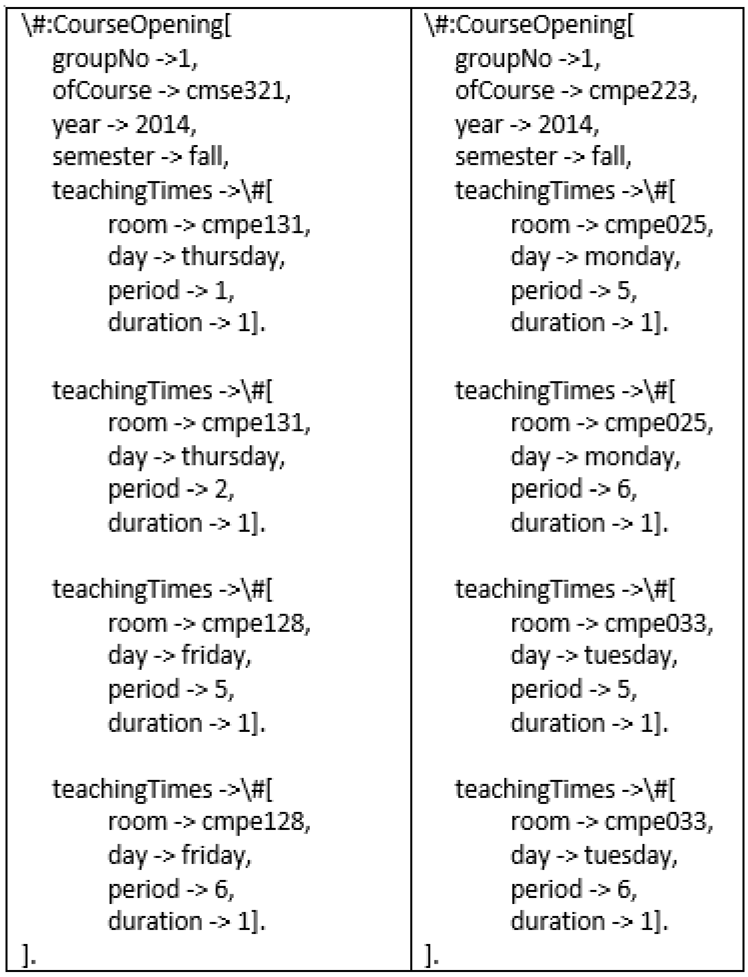
Figure 4.11 Classifying grades

4.4 Representative Concept Instances

Instance is a specific realization of any object. In our study, we create objects for concepts. In this section, we show representative instances of concepts.

4.4.1 Representative Instances of CourseOpening

Figure 4.12 shows the instance of two *CourseOpening* conepts. All attribues of concept of course opening are initialized.



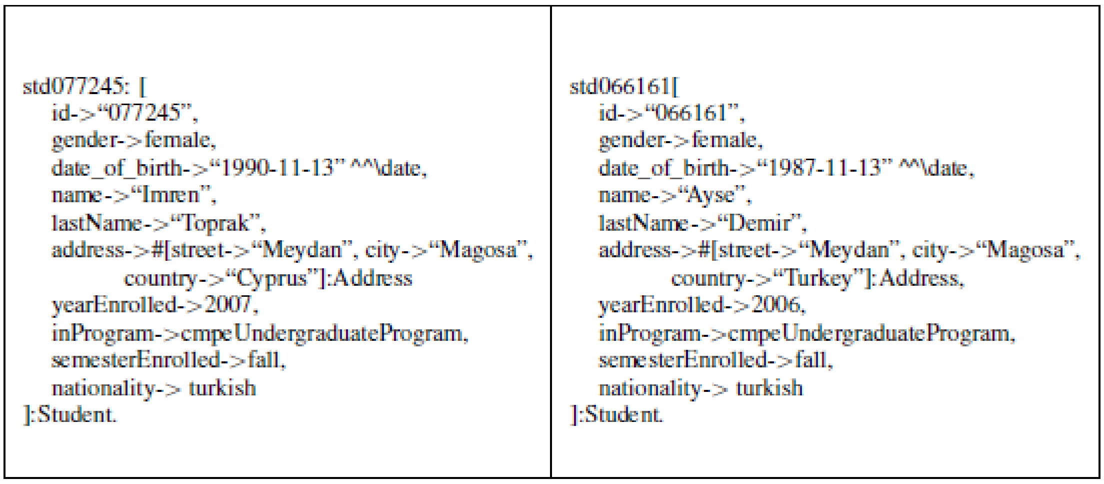
```
\#:CourseOpening[
    groupNo ->1,
    ofCourse -> cmse321,
    year -> 2014,
    semester -> fall,
    teachingTimes ->\#[
            room -> cmpe131,
            day -> thursday,
            period -> 1,
            duration -> 1].

    teachingTimes ->\#[
            room -> cmpe131,
            day -> thursday,
            period -> 2,
            duration -> 1].

    teachingTimes ->\#[
            room -> cmpe128,
            day -> friday,
            period -> 5,
            duration -> 1].

    teachingTimes ->\#[
            room -> cmpe128,
            day -> friday,
            period -> 6,
            duration -> 1].
].
```

```
\#:CourseOpening[
    groupNo ->1,
    ofCourse -> cmpe223,
    year -> 2014,
    semester -> fall,
    teachingTimes ->\#[
            room -> cmpe025,
            day -> monday,
            period -> 5,
            duration -> 1].

    teachingTimes ->\#[
            room -> cmpe025,
            day -> monday,
            period -> 6,
            duration -> 1].

    teachingTimes ->\#[
            room -> cmpe033,
            day -> tuesday,
            period -> 5,
            duration -> 1].

    teachingTimes ->\#[
            room -> cmpe033,
            day -> tuesday,
            period -> 6,
            duration -> 1].
].
```

Figure 4.12: Representative instances for CourseOpening concept

4.4.2 Representative Instances of Student

Figure 4.13 shows the instances of two *Student* concepts. All attribues of concept of student are initialized.



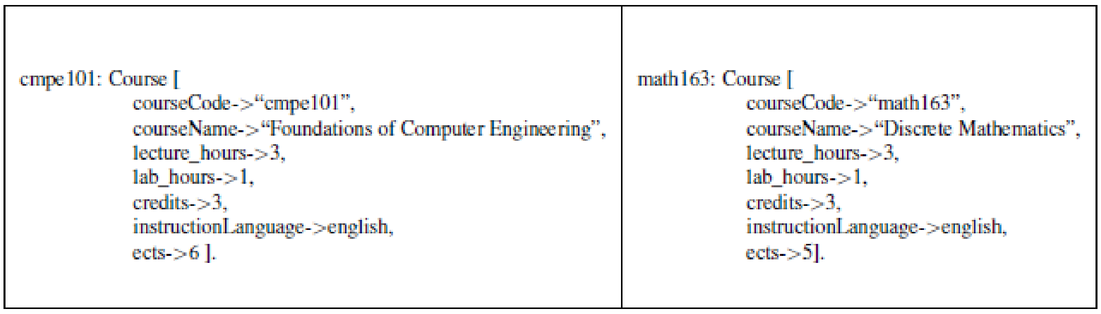
```
std077245: [
    id->"077245",
    gender->female,
    date_of_birth->"1990-11-13" ^^\date,
    name->"Imren",
    lastName->"Toprak",
    address->#[street->"Meydan", city->"Magosa",
              country->"Cyprus"]:Address
    yearEnrolled->2007,
    inProgram->cmpeUndergraduateProgram,
    semesterEnrolled->fall,
    nationality-> turkish
]:Student.
```

```
std066161[
    id->"066161",
    gender->female,
    date_of_birth->"1987-11-13" ^^\date,
    name->"Ayse",
    lastName->"Demir",
    address->#[street->"Meydan", city->"Magosa",
              country->"Turkey"]:Address,
    yearEnrolled->2006,
    inProgram->cmpeUndergraduateProgram,
    semesterEnrolled->fall,
    nationality-> turkish
]:Student.
```

Figure 4.13 Representative instances of Student concept

4.4.3 Representative Instances of Course

Figure 4.14 shows the instances of the *Course* concept.



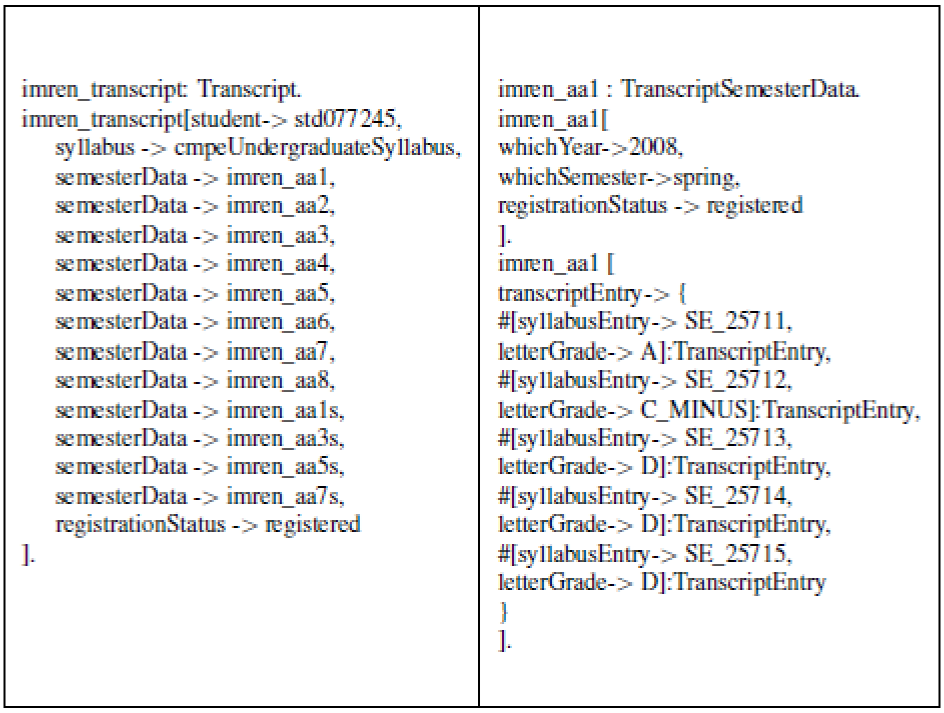
```
cmpe101: Course [
            courseCode->"cmpe101",
            courseName->"Foundations of Computer Engineering",
            lecture_hours->3,
            lab_hours->1,
            credits->3,
            instructionLanguage->english,
            ects->6 ].
```

```
math163: Course [
            courseCode->"math163",
            courseName->"Discrete Mathematics",
            lecture_hours->3,
            lab_hours->1,
            credits->3,
            instructionLanguage->english,
            ects->5].
```

Figure 4.14: Representative instance of Course concept

4.4.4 Representative Instance of Transcript

Figure 4.15 shows an instance of the *Transcript* concept.



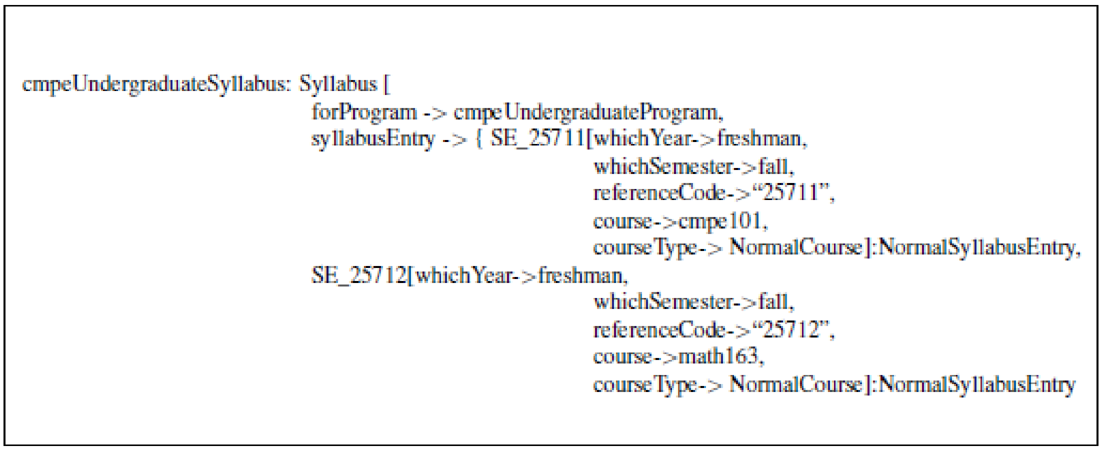
```
imren_transcript: Transcript.
imren_transcript[student-> std077245,
    syllabus -> cmpeUndergraduateSyllabus,
    semesterData -> imren_aa1,
    semesterData -> imren_aa2,
    semesterData -> imren_aa3,
    semesterData -> imren_aa4,
    semesterData -> imren_aa5,
    semesterData -> imren_aa6,
    semesterData -> imren_aa7,
    semesterData -> imren_aa8,
    semesterData -> imren_aa1s,
    semesterData -> imren_aa3s,
    semesterData -> imren_aa5s,
    semesterData -> imren_aa7s,
    registrationStatus -> registered
].
```

```
imren_aa1 : TranscriptSemesterData.
imren_aa1[
whichYear->2008,
whichSemester->spring,
registrationStatus -> registered
].
imren_aa1 [
transcriptEntry-> {
#[syllabusEntry-> SE_25711,
letterGrade-> A]:TranscriptEntry,
#[syllabusEntry-> SE_25712,
letterGrade-> C_MINUS]:TranscriptEntry,
#[syllabusEntry-> SE_25713,
letterGrade-> D]:TranscriptEntry,
#[syllabusEntry-> SE_25714,
letterGrade-> D]:TranscriptEntry,
#[syllabusEntry-> SE_25715,
letterGrade-> D]:TranscriptEntry
}
].
```

Figure 4.15: Representative instance of Transcript concept

4.4.5 Representative Instance of Syllabus

Figure 4.16 shows an instance of the *Syllabus* concept. (In this thesis, we use the terms syllabus and curriulum interchangeable).

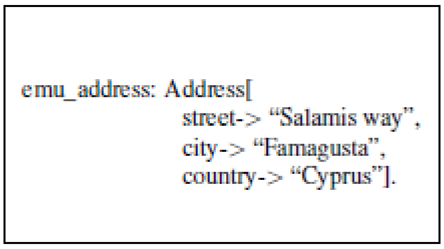


```
cmpeUndergraduateSyllabus: Syllabus [
                           forProgram -> cmpeUndergraduateProgram,
                           syllabusEntry -> { SE_25711[whichYear->freshman,
                                                       whichSemester->fall,
                                                       referenceCode->"25711",
                                                       course->cmpe101,
                                                       courseType-> NormalCourse]:NormalSyllabusEntry,
                           SE_25712[whichYear->freshman,
                                                       whichSemester->fall,
                                                       referenceCode->"25712",
                                                       course->math163,
                                                       courseType-> NormalCourse]:NormalSyllabusEntry
```

Figure 4.16: Representative instance of Syllabus concept

4.4.6 Representative Instance of Address

Figure 4.17 shows an instance of the *Address* concept.

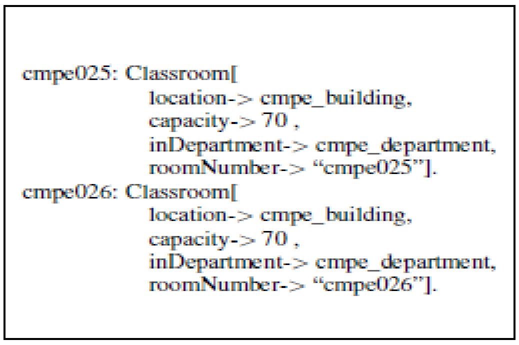


```
emu_address: Address[
                    street-> "Salamis way",
                    city-> "Famagusta",
                    country-> "Cyprus"].
```

Figure 4.17: Representative instance of Address concept

4.4.7 Representative Instances of Classroom

Figure 4.18 shows the instances of the *Classroom* concept.

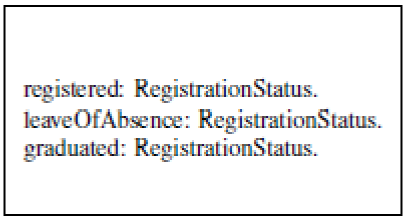


```
cmpe025: Classroom[
                location-> cmpe_building,
                capacity-> 70 ,
                inDepartment-> cmpe_department,
                roomNumber-> "cmpe025"].
cmpe026: Classroom[
                location-> cmpe_building,
                capacity-> 70 ,
                inDepartment-> cmpe_department,
                roomNumber-> "cmpe026"].
```

Figure 4.18: Representative instances of Classroom

4.4.8 Representative Instance of RegistrationStatus

Figure 4.19 shows the instances of the *RegistrationStatus* concept.

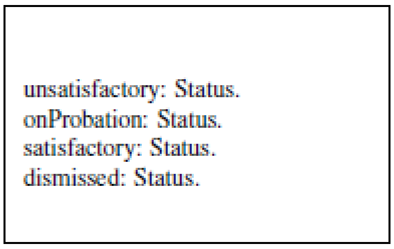


```
registered: RegistrationStatus.
leaveOfAbsence: RegistrationStatus.
graduated: RegistrationStatus.
```

Figure 4.19: Representative instances of RegistrationStatus concept

4.4.9 Representative Instance of Student Status

Figure 4.20 shows the instances of the *Status* concept.

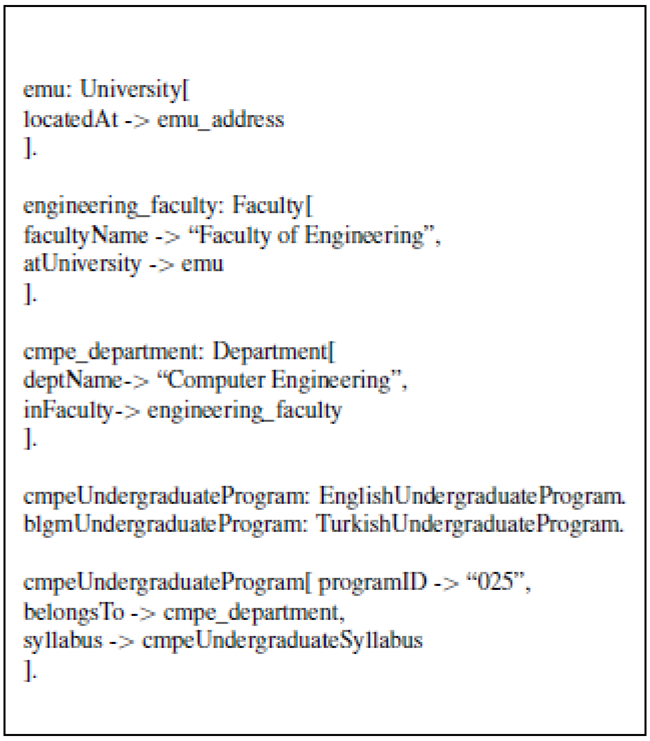


```
unsatisfactory: Status.
onProbation: Status.
satisfactory: Status.
dismissed: Status.
```

Figure 4.20: Representative instance of Status concept

4.4.10 Representative Instance of University, Faculty, Department and Program

Figure 4.21 shows instances of *University*, *Faculty*, *Department and Program* concepts.

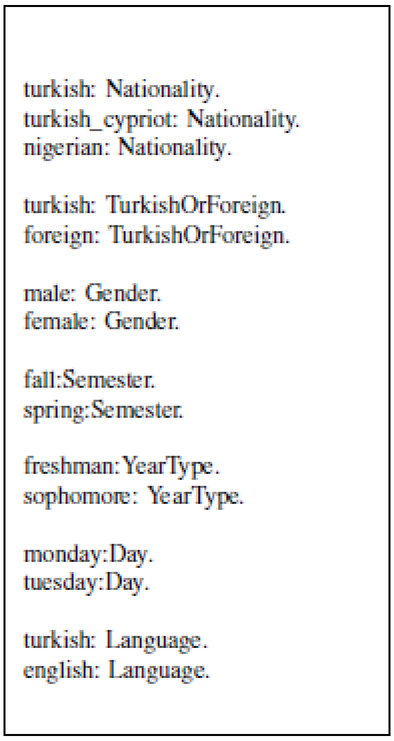


```
emu: University[
locatedAt -> emu_address
].

engineering_faculty: Faculty[
facultyName -> "Faculty of Engineering",
atUniversity -> emu
].

cmpe_department: Department[
deptName-> "Computer Engineering",
inFaculty-> engineering_faculty
].

cmpeUndergraduateProgram: EnglishUndergraduateProgram.
blgmUndergraduateProgram: TurkishUndergraduateProgram.

cmpeUndergraduateProgram[ programID -> "025",
belongsTo -> cmpe_department,
syllabus -> cmpeUndergraduateSyllabus
].
```

Figure 4.21: Representative instance of University, Faculty, Department and Program concept

4.4.11 Representative Instance of Other Miscellaneous Concepts

Figure 4.22 shows instance of other concepts such as *Nationality*, *Gender*, *Semester*, *YearType*, *Day*, *Language* and *TurkishOrForeign*. *TurkishOrForeign* is used to separate turkish and foreign students.



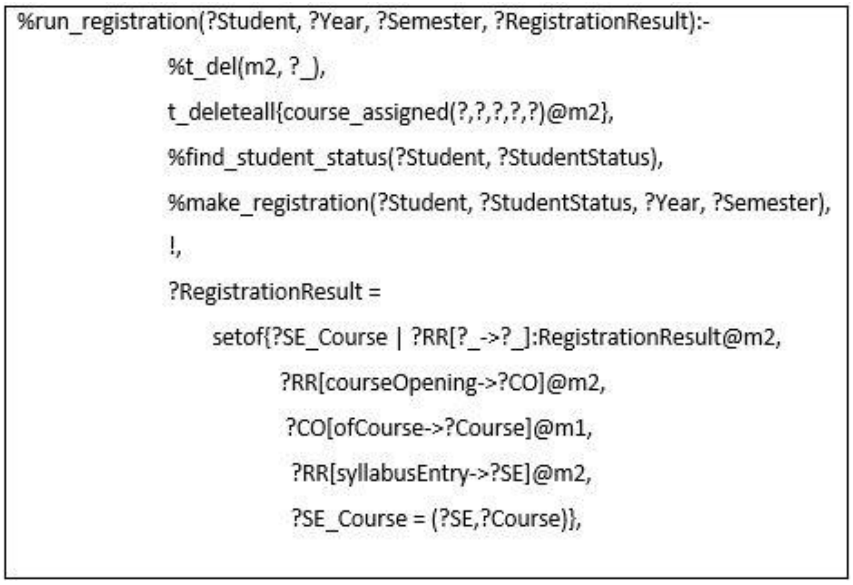
```
turkish: Nationality.
turkish_cypriot: Nationality.
nigerian: Nationality.

turkish: TurkishOrForeign.
foreign: TurkishOrForeign.

male: Gender.
female: Gender.

fall:Semester.
spring:Semester.

freshman:YearType.
sophomore: YearType.

monday:Day.
tuesday:Day.

turkish: Language.
english: Language.
```

Figure 4.22: Representative instance of other miscellaneous concepts

4.5 High-Level Explanation of the Predicates Used in the Implementation

4.5.1 The Main Predicate

The main predicate is *%run_registration(?Student, ?Year, ?Semester, ?RegistrationResult).* The name of the predicate is *run_registration.* It has 4 parameters that are *?Student, ?Year, ?Semester,* and *?RegistrationResult.* Firstly, it calls *t_del* and *t_deleteall* primitives to update the runtime database. Then, predicate *find_student_status* is called to find status of the student. Finally, it calls *make_registration* which finds the suggested course list and put them temporarily into the run time knowledge base. ?*RegistrationResult* gives to us the list of courses, as shown in figure 4.23.



```
%run_registration(?Student, ?Year, ?Semester, ?RegistrationResult):-
                %t_del(m2, ?_),
                t_deleteall{course_assigned(?,?,?,?,?)@m2},
                %find_student_status(?Student, ?StudentStatus),
                %make_registration(?Student, ?StudentStatus, ?Year, ?Semester),
                !,
                ?RegistrationResult =
                     setof{?SE_Course | ?RR[?_->?_]:RegistrationResult@m2,
                            ?RR[courseOpening->?CO]@m2,
                            ?CO[ofCourse->?Course]@m1,
                             ?RR[syllabusEntry->?SE]@m2,
                             ?SE_Course = (?SE,?Course)},
```

Figure 4.23: Main predicate for course finder module

4.5.2 Predicates for Transcript Generation

There are no predicates used in the generation of transcripts. All computation is done through rules that define the values of attributes, which were shown in section 4.2.2.

4.5.3 Predicates for Registration Advisor

%r2: Loads the separate program files into the Flora-2 environment.

%run(?RR): Executes the agent and returns in ?RR the courses recommended for the student. The curriculum of the program to which the student is registered, the transcript of the student, status of the student, as well as the courses that are opened for the current semester are taken into account.

%run_registration(?Std, ?Year, ?Sem, ?RR): Called by %run to find the result for a specific student, year and semester.

%find_student_status(?Std, ?StdStatu): Finds the student's status based upon his/her CGPA and actual academic semester. The status can be satisfactory, on probation, unsatisfactory and dismissed.

%make_registration(?Std, ?Statu,?Year,?Sem): Finds a registration suggestion for a student based upon his/her status.

%how_many_zero_one_credit(?LFC, ?H): Determines how many of the student’s failed courses have zero or one credit.

%maxNumberOfCoursesStudentCanTake(?Std,?M): How many credited courses a student can take. This depends upon the students current semester, GPA and how many courses the student has remaining until graduation.

%make_registration_repeat_courses_list(?List, ?_Std,?_Year,?_Sem): Registers a student to failed courses using the list of failed courses.

%make_registration_repeat_courses(?N, ?_Std,?_Year,?_Sem): Registers a student to previously taken courses if the student needs to repeat courses due to his/her status (on probation or unsatisfactory).

%make_registration_new_courses(?N, ?_Std,?_Year,?_Sem): Registers a student to courses that have not been taken before.

%make_single_registration_specific_course(?SE,?Std, ?Year, ?Sem): Attempts a registration for a specific course that was taken before.

%make_single_registration_repeat_courses(?Std, ?Year, ?Sem): Suggests registration for non-failed courses that the student should take again.

%make_single_registration_new_courses(?Std, ?Year, ?Sem): Suggests registration for new courses.

%courses_not_taken_by_student(?Std,?CNT): Finds the courses that have not yet been taken by the student.

%failedCoursesStudentShouldTake(?Std, ?FC): Finds list of failed courses that the student should take again.

%moveToFrontLowCreditCourses(?S, ?SList): Gives higher priority to zero or one credit courses, but only if less than 12 courses remain. Used to enforce the selection of the first part of the graduation project in the last year.

%studentTakenAllPrerequisites(?Std, ?Crs): Checks that the student has taken with a passing grade all the prerequisites of a given course.

%takenCourseWithPassingGrade(?Std, ?Crs): Checks that the student has taken a course with a passing grade.

%taken_SE_with_passing_grade(?Std, ?SE): Checks that the student has passed a course belonging to a specific category in the curriculum (for example, a university elective).

%taken_SE_with_failing_grade(?Std, ?SE): Checks that the student has not passed any course belonging to a specific category in the curriculum (for example, a university elective).

%find_course_opening(?Std, ?AS, ?Chck, ?CO): Finds a course opening for a reference code which can be associated with more than one course (for example, area elective). ?AS contains information about which courses can be taken for a specific reference code.

%mostRecentGradeRealSem(?Std, ?Crs, ?G): Finds the most recent grade a student obtained in a course, taking into account summer semesters as well.

%findMostRecentGradeRealSem(?T, ?Term, ?Crs, ?G): Utility predicate for finding the most recent grade a student obtained in a course, taking into account summer semesters as well. Called by %mostRecentGradeRealSem.

%mostRecentGradeSERealSem(?Std, ?SE, ?G): Finds the most recent grade a student obtained in some course with a reference code which can be associated with more than one course (for example, area elective), taking into account summer semesters as well.

4.5.4 Predicates For Optimizing the Number of Clashes

%how_many_clashes_all(?Std, ?Year, ?Sem, ?CO): Finds a registration suggestion for a given student, year and semester such that the number of clashes is minimized. Prints a list of suggestions in increasing order of the number of clashes.

%how_many_clashes(?Std, ?Year, ?Seme, ?CP, ?HM): Returns how many clashes a specific registration suggestion has.

%howManyClashesTwoCO(?CO1, ?CO2, ?HM): Returns how many clashes two specific course openings have.

%courses_assigned_to_student(?Std,?Year, ?Sem,?CrsL): Returns the list of courses that the student should take in the given year and semester.

%courseOpeningsForStudent(?Std, ?Year, ?Sem, ?COL): Returns the list of course openings suggested by the registration advisor component.

%courseOpeningsForStudent0(?_Year, ?_Sem, ?Crslist, COP2): Helper predicate called by the %courseOpeningsForStudent predicate.

%insertCourseOpeningsForStudent(?_Std,?_Year, ?_Sem, ?COP2): Utility predicate called by %how_many_clashes. Inserts temporarily into the logic base information about which courses should be taken by the student.

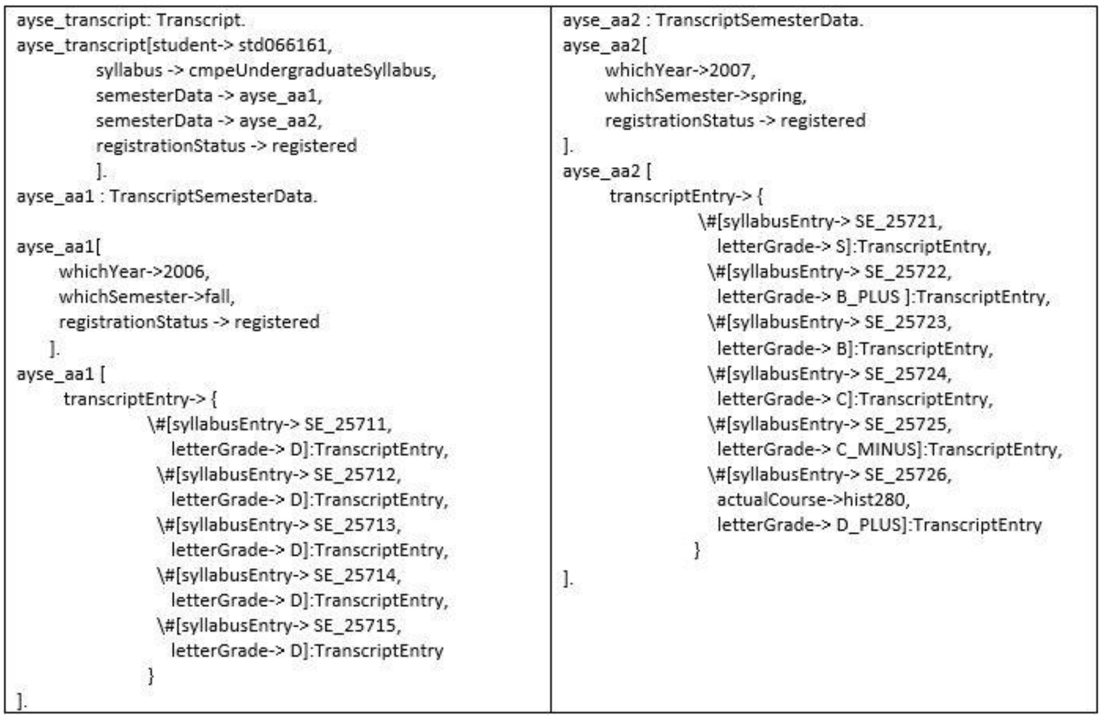
Chapter 5

SAMPLE RUNS

In this chapter, we present sample runs of our agent. Firstly, we run the module of *transcript generator*. Second, we run the module of *course finder*. Third, we demonstrate screen shot of results while running *optimization* module.

5.1 Sample Runs for Transcript Generator

In this section we show sample runs of *transcript generator* module. First, we give sample of a transcript. We run transcript generator module according to the given transcript. Figure 5.1 show us transcript information for first and second semester. *ayse_aa1* represents the first semester transcript information. *ayse_aa2* represents the second semester transcript information.

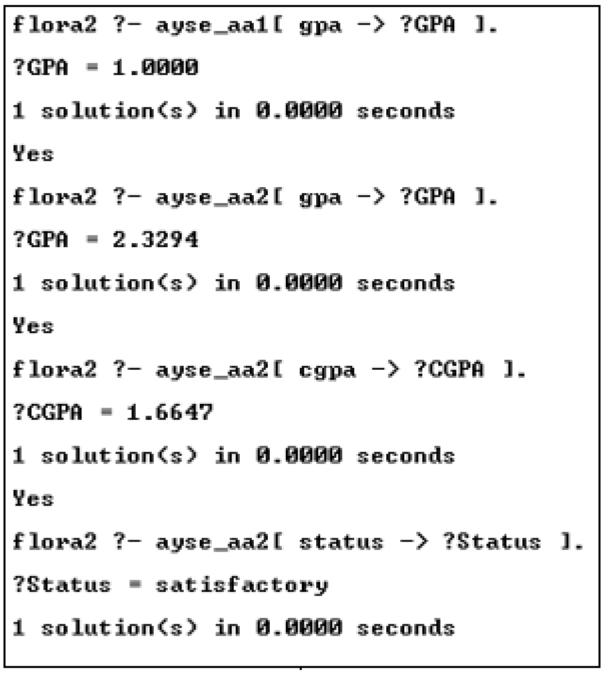


```
ayse_transcript: Transcript.
ayse_transcript[student-> std066161,
          syllabus -> cmpeUndergraduateSyllabus,
          semesterData -> ayse_aa1,
          semesterData -> ayse_aa2,
          registrationStatus -> registered
          ].
ayse_aa1 : TranscriptSemesterData.

ayse_aa1[
     whichYear->2006,
     whichSemester->fall,
     registrationStatus -> registered
    ].
ayse_aa1 [
      transcriptEntry-> {
                \#[syllabusEntry-> SE_25711,
                  letterGrade-> D]:TranscriptEntry,
                 \#[syllabusEntry-> SE_25712,
                  letterGrade-> D]:TranscriptEntry,
                 \#[syllabusEntry-> SE_25713,
                  letterGrade-> D]:TranscriptEntry,
                 \#[syllabusEntry-> SE_25714,
                  letterGrade-> D]:TranscriptEntry,
                 \#[syllabusEntry-> SE_25715,
                  letterGrade-> D]:TranscriptEntry
                }
].
```

```
ayse_aa2 : TranscriptSemesterData.
ayse_aa2[
     whichYear->2007,
     whichSemester->spring,
     registrationStatus -> registered
].
ayse_aa2 [
      transcriptEntry-> {
                \#[syllabusEntry-> SE_25721,
                  letterGrade-> S]:TranscriptEntry,
                 \#[syllabusEntry-> SE_25722,
                  letterGrade-> B_PLUS ]:TranscriptEntry,
                 \#[syllabusEntry-> SE_25723,
                  letterGrade-> B]:TranscriptEntry,
                 \#[syllabusEntry-> SE_25724,
                  letterGrade-> C]:TranscriptEntry,
                 \#[syllabusEntry-> SE_25725,
                  letterGrade-> C_MINUS]:TranscriptEntry,
                 \#[syllabusEntry-> SE_25726,
                  actualCourse->hist280,
                  letterGrade-> D_PLUS]:TranscriptEntry
                }
].
```

Figure 5.1: Transcript information for the first and second semester

Transcript generator module computes the GPA, CGPA and status according to these two semester transcript information. Figure 5.2 shows the result of transcript generator for its queries. First query is ayse_aa1[gpa -> ?GPA] that gives the GPA for first semester. Second query is ayse_aa2[gpa -> ?GPA] thar gives the GPA for second semester. Third query is ayse_aa2[cgpa -> ?CGPA] that gives the CGPA for two semester. The last query is ayse_aa2[status -> ?Status] that gives status of the student.



```
flora2 ?- ayse_aa1[ gpa -> ?GPA ].
?GPA = 1.0000
1 solution(s) in 0.0000 seconds
Yes
flora2 ?- ayse_aa2[ gpa -> ?GPA ].
?GPA = 2.3294
1 solution(s) in 0.0000 seconds
Yes
flora2 ?- ayse_aa2[ cgpa -> ?CGPA ].
?CGPA = 1.6647
1 solution(s) in 0.0000 seconds
Yes
flora2 ?- ayse_aa2[ status -> ?Status ].
?Status = satisfactory
1 solution(s) in 0.0000 seconds
```

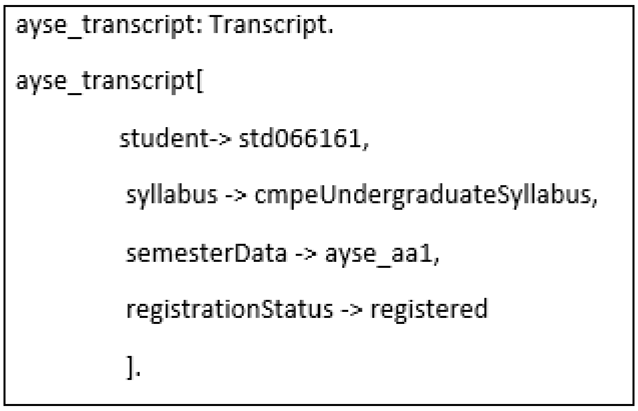
Figure 5.2: GPA, CGPA and status for a given transcript information

5.2 Sample Runs for Course Finder – Regular Student

In this section we show sample runs of *course finder* which is a one of the modules of REG-EXPERT to introduce this module of our agent to the user.

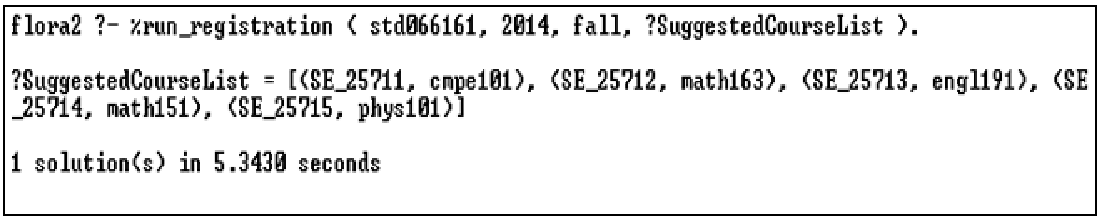
Normally, a student who is regular should finish the department in eight semesters. As we described in chapter 2, there are forty courses that are four or three credited, one course that is one credited and two courses that are non-credited. There are sample runs of module of *course finder* for each semester of a regular student. We consider student has satisfactory status for each semester.

In the first semester, student should be registered to five courses that are specified in the department curriculum. Since this is first semester for a student there is no any transcript yet. Figure 5.3 shows the semester data for a first semester. *ayse_aa1* represents the first semester transcript. In figure 5.4 we can see suggestion of courses that should be taken in the given year and semester. We give student number (*std066161*), year (*2014*), semester (*fall*) to which we want to register the student and query result (*?SuggestedCourseList*) as a input. Our agent checks the all constraints and gives a list of courses according to department curriculum as a result.



```
ayse_transcript: Transcript.

ayse_transcript[

          student-> std066161,

           syllabus -> cmpeUndergraduateSyllabus,

           semesterData -> ayse_aa1,

           registrationStatus -> registered

           ].
```

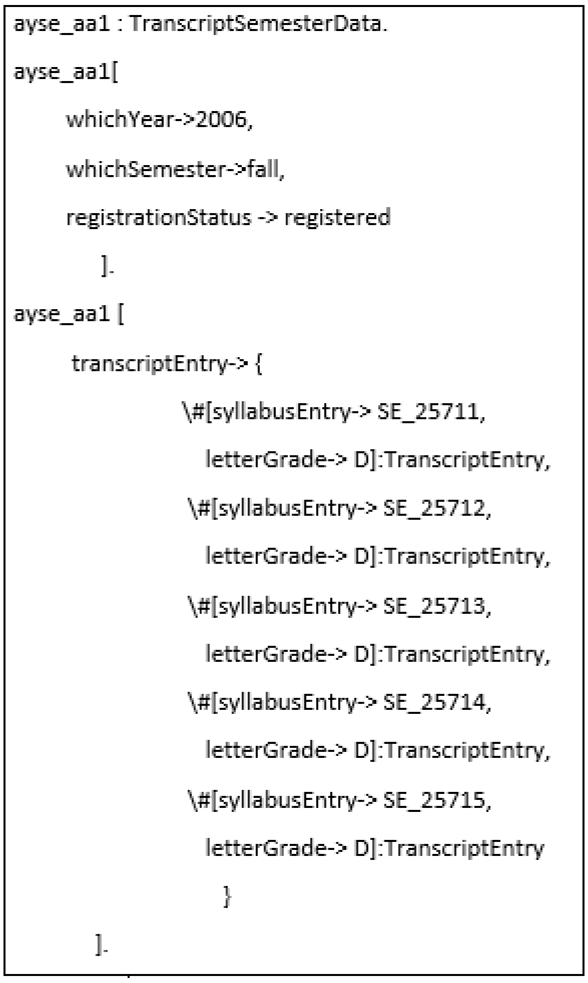
Figure 5.3: Semester data for the first semester transcript



```
flora2 ?- %run_registration ( std066161, 2014, fall, ?SuggestedCourseList ).

?SuggestedCourseList = [(SE_25711, cmpe101), (SE_25712, math163), (SE_25713, engl191), (SE
_25714, math151), (SE_25715, phys101)]

1 solution(s) in 5.3430 seconds
```

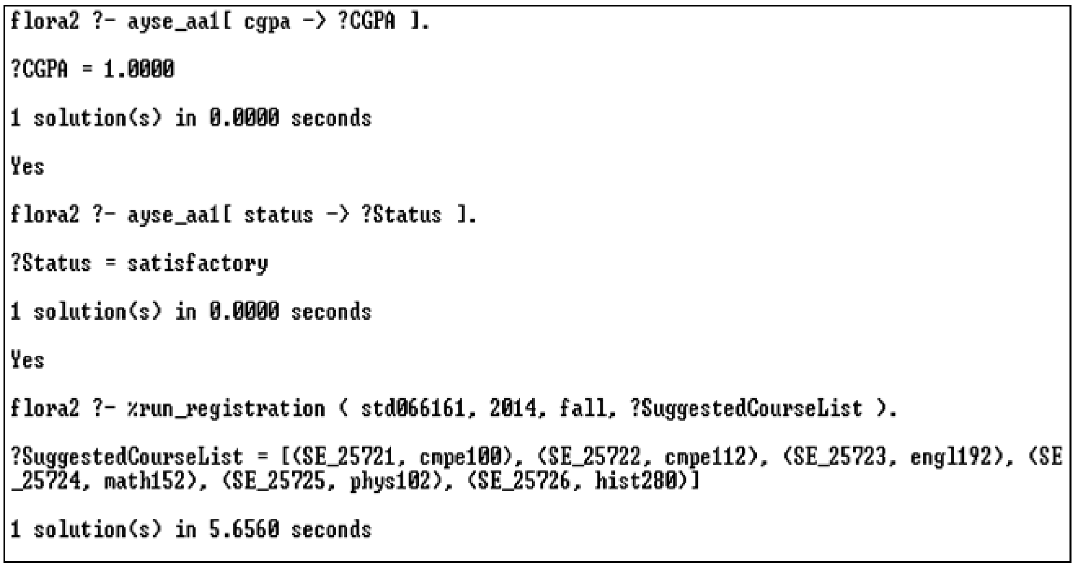
Figure 5.4: Suggested course list for the first semester

Figure 5.5 shows the first semester transcript information of the student. Figure 5.6 shows the CGPA and status of the student for the first semester. Also, it gives the list of courses suggested according to the current transcript information. It is like a run of first semester, we give same parameters as a input. Our agent gives courses of second semester as a result.



```
ayse_aa1 : TranscriptSemesterData.
ayse_aa1[
        whichYear->2006,
        whichSemester->fall,
        registrationStatus -> registered
             ].
ayse_aa1 [
         transcriptEntry-> {
                          \#[syllabusEntry-> SE_25711,
                             letterGrade-> D]:TranscriptEntry,
                           \#[syllabusEntry-> SE_25712,
                             letterGrade-> D]:TranscriptEntry,
                           \#[syllabusEntry-> SE_25713,
                             letterGrade-> D]:TranscriptEntry,
                           \#[syllabusEntry-> SE_25714,
                             letterGrade-> D]:TranscriptEntry,
                           \#[syllabusEntry-> SE_25715,
                             letterGrade-> D]:TranscriptEntry
                               }
            ].
```

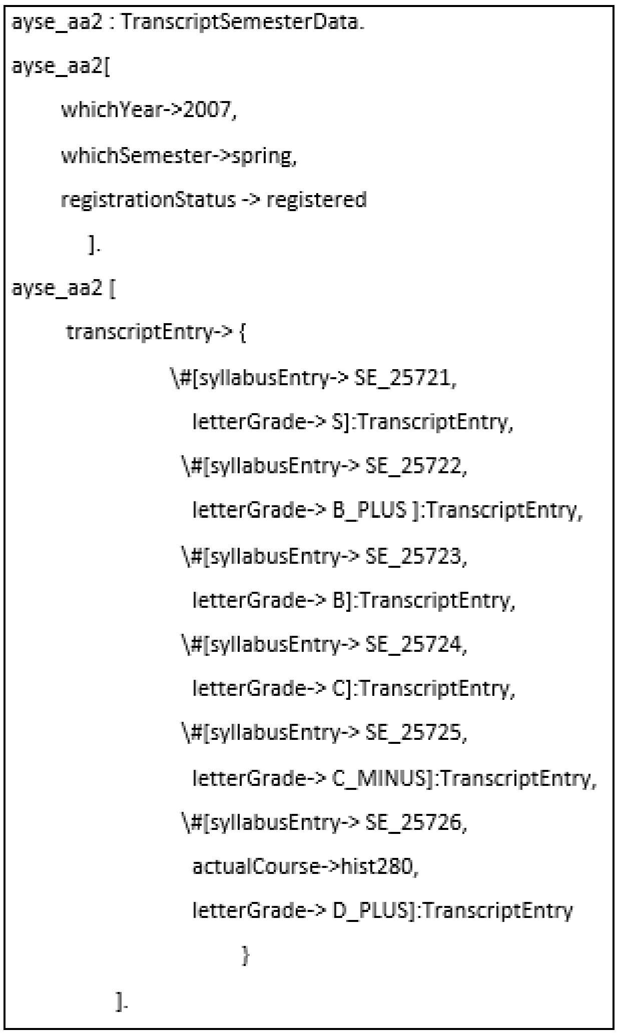
Figure 5.5: Transcript information for the first semester



```
flora2 ?- ayse_aa1[ cgpa -> ?CGPA ].

?CGPA = 1.0000

1 solution(s) in 0.0000 seconds

Yes

flora2 ?- ayse_aa1[ status -> ?Status ].

?Status = satisfactory

1 solution(s) in 0.0000 seconds

Yes

flora2 ?- %run_registration ( std066161, 2014, fall, ?SuggestedCourseList ).

?SuggestedCourseList = [(SE_25721, cmpe100), (SE_25722, cmpe112), (SE_25723, engl192), (SE
_25724, math152), (SE_25725, phys102), (SE_25726, hist280)]

1 solution(s) in 5.6560 seconds
```

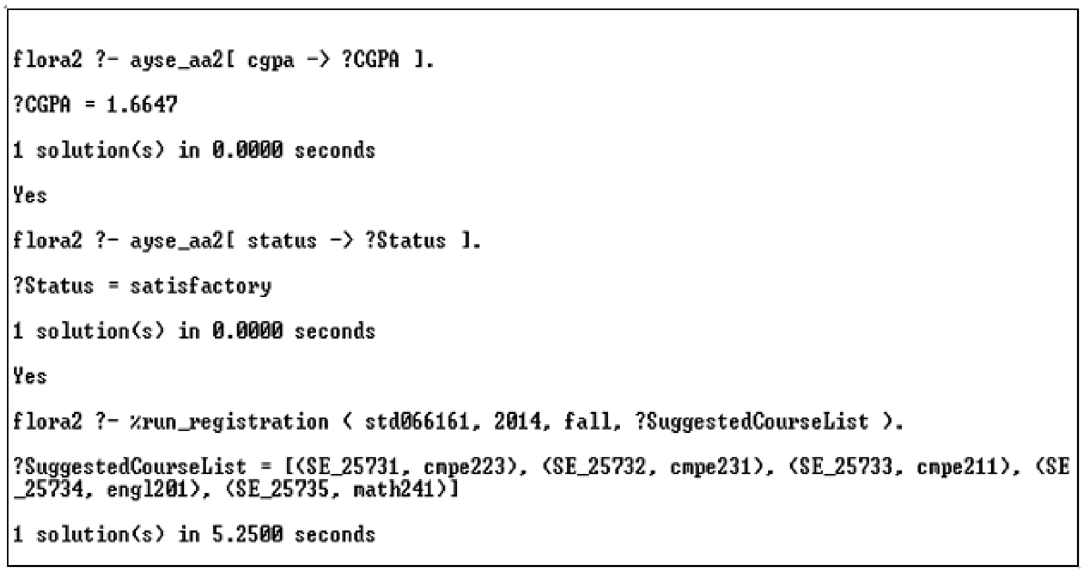
Figure 5.6: Suggested course list for the second semester

In addition to the first semester transcript, figure 5.7 shows the second semester transcript information of a student. Figure 5.8 shows the CGPA and status of the student for the second semester and gives the list of courses suggested according to the current transcript information. It is like a run of first semester, we give same parameters as a input. Our agent gives courses of third semester as a result.



```
ayse_aa2 : TranscriptSemesterData.
ayse_aa2[
       whichYear->2007,
       whichSemester->spring,
       registrationStatus -> registered
          ].
ayse_aa2 [
        transcriptEntry-> {
                     \#[syllabusEntry-> SE_25721,
                       letterGrade-> S]:TranscriptEntry,
                      \#[syllabusEntry-> SE_25722,
                       letterGrade-> B_PLUS ]:TranscriptEntry,
                      \#[syllabusEntry-> SE_25723,
                       letterGrade-> B]:TranscriptEntry,
                      \#[syllabusEntry-> SE_25724,
                       letterGrade-> C]:TranscriptEntry,
                      \#[syllabusEntry-> SE_25725,
                       letterGrade-> C_MINUS]:TranscriptEntry,
                      \#[syllabusEntry-> SE_25726,
                       actualCourse->hist280,
                       letterGrade-> D_PLUS]:TranscriptEntry
                              }
              ].
```

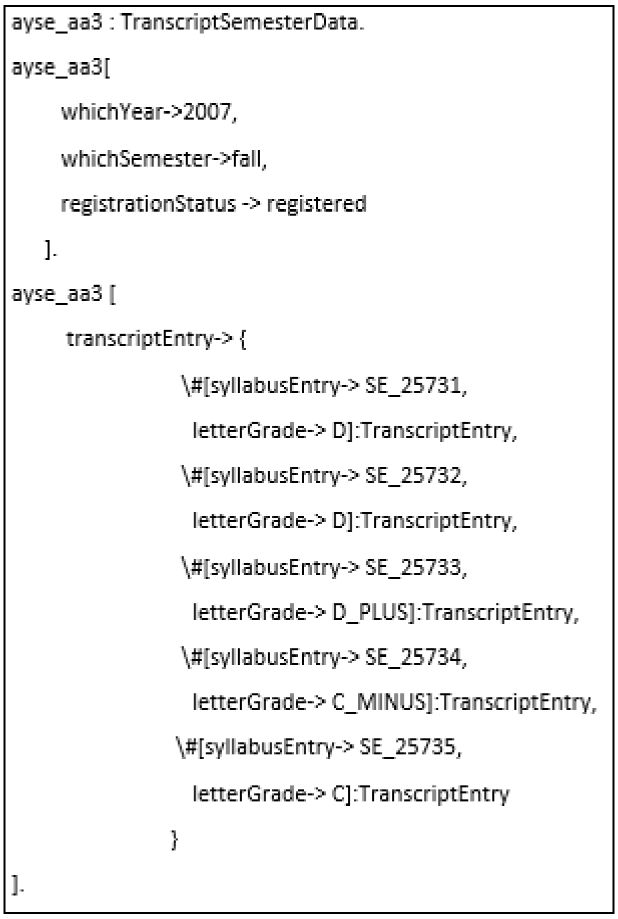
Figure 5.7: Transcript information for the second



```
flora2 ?- ayse_aa2[ cgpa -> ?CGPA ].

?CGPA = 1.6647

1 solution(s) in 0.0000 seconds

Yes

flora2 ?- ayse_aa2[ status -> ?Status ].

?Status = satisfactory

1 solution(s) in 0.0000 seconds

Yes

flora2 ?- %run_registration ( std066161, 2014, fall, ?SuggestedCourseList ).

?SuggestedCourseList = [(SE_25731, cmpe223), (SE_25732, cmpe231), (SE_25733, cmpe211), (SE
_25734, engl201), (SE_25735, math241)]

1 solution(s) in 5.2500 seconds
```

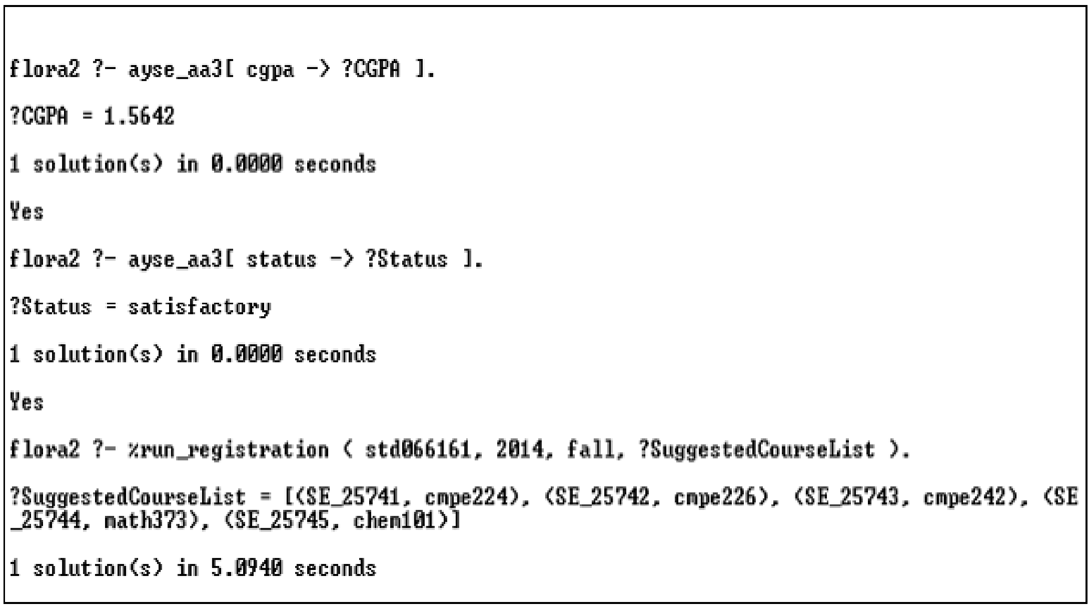
Figure 5.8: Suggested course list for the third semester

In addition to the first and second semester transcript, figure 5.9 shows the third semester transcript information of a student. Figure 5.10 shows the CGPA and status of the student for third semester and gives the list of courses suggested according to the current transcript information. Input is like a run of first semester, we give same parameters as a input. Our agent gives courses of fourth semester as a result.



```
ayse_aa3 : TranscriptSemesterData.
ayse_aa3[
      whichYear->2007,
      whichSemester->fall,
      registrationStatus -> registered
    ].
ayse_aa3 [
       transcriptEntry-> {
                   \#[syllabusEntry-> SE_25731,
                     letterGrade-> D]:TranscriptEntry,
                   \#[syllabusEntry-> SE_25732,
                     letterGrade-> D]:TranscriptEntry,
                   \#[syllabusEntry-> SE_25733,
                     letterGrade-> D_PLUS]:TranscriptEntry,
                   \#[syllabusEntry-> SE_25734,
                     letterGrade-> C_MINUS]:TranscriptEntry,
                  \#[syllabusEntry-> SE_25735,
                     letterGrade-> C]:TranscriptEntry
                  }
].
```

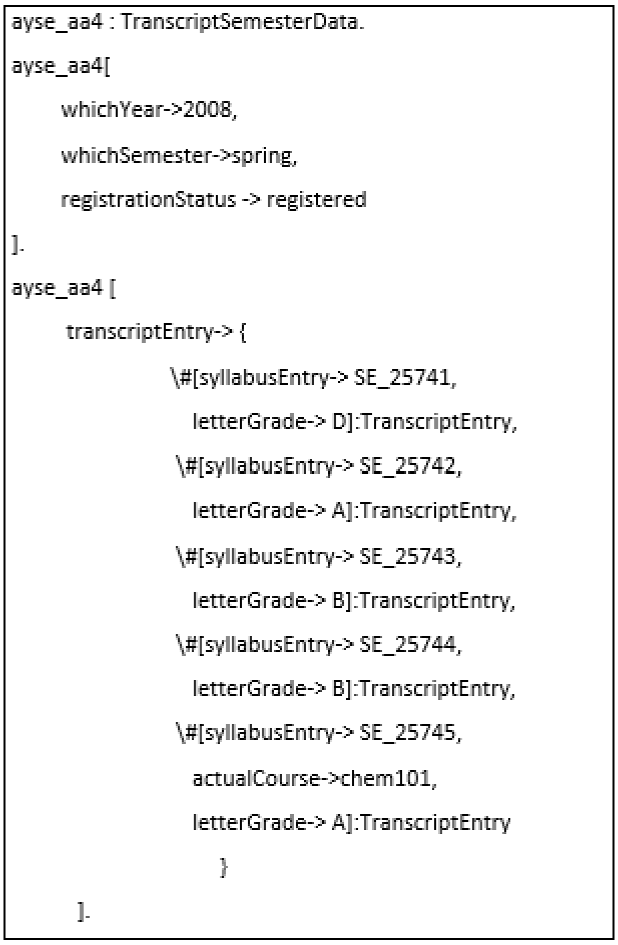
Figure 5.9: Transcript information for the third semester



```
flora2 ?- ayse_aa3[ cgpa -> ?CGPA ].

?CGPA = 1.5642

1 solution(s) in 0.0000 seconds

Yes

flora2 ?- ayse_aa3[ status -> ?Status ].

?Status = satisfactory

1 solution(s) in 0.0000 seconds

Yes

flora2 ?- %run_registration ( std066161, 2014, fall, ?SuggestedCourseList ).

?SuggestedCourseList = [(SE_25741, cmpe224), (SE_25742, cmpe226), (SE_25743, cmpe242), (SE
_25744, math373), (SE_25745, chem101)]

1 solution(s) in 5.0940 seconds
```

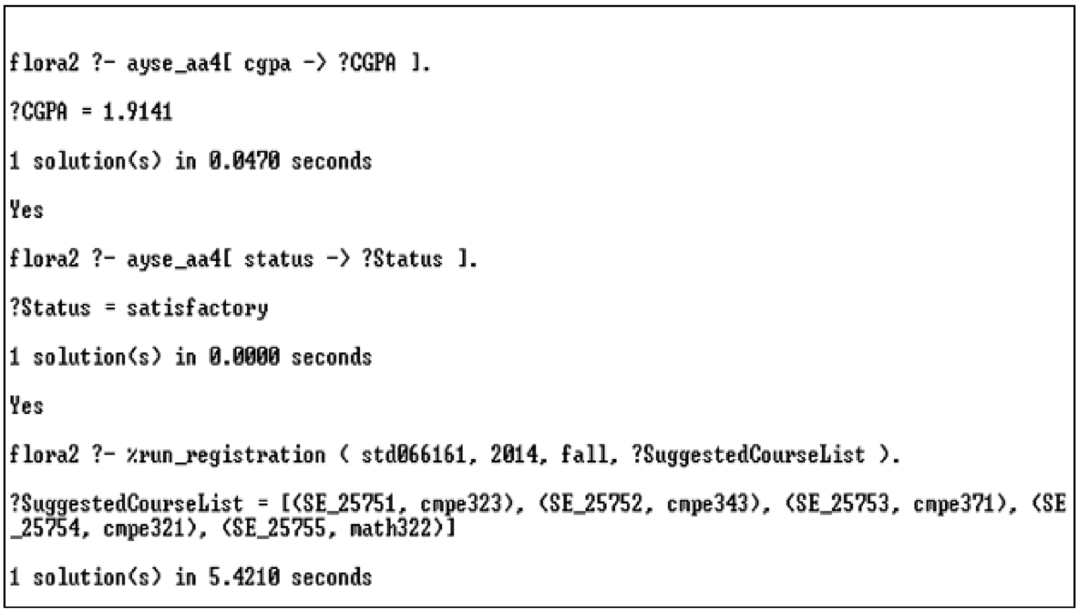
Figure 5.10: Suggested course list for the fourth semester

In addition to the first, second and third semester transcript, figure 5.11 shows the fourth semester transcript information of a student. Figure 5.12 shows the CGPA and status of the student for fourth semester and gives the list of courses suggested according to the current transcript information. Input is like a run of first semester, we give same parameters as a input. Our agent gives courses of fifth semester as a result.



```
ayse_aa4 : TranscriptSemesterData.
ayse_aa4[
      whichYear->2008,
      whichSemester->spring,
      registrationStatus -> registered
].
ayse_aa4 [
       transcriptEntry-> {
                   \#[syllabusEntry-> SE_25741,
                     letterGrade-> D]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25742,
                     letterGrade-> A]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25743,
                     letterGrade-> B]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25744,
                     letterGrade-> B]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25745,
                     actualCourse->chem101,
                     letterGrade-> A]:TranscriptEntry
                         }
        ].
```

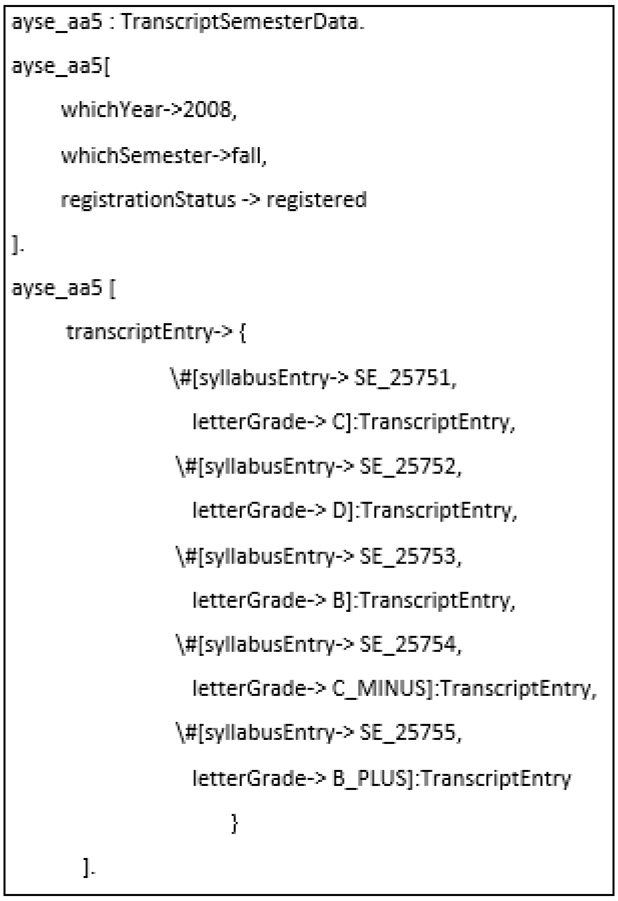
Figure 5.11: Transcript information for the fourth semester



```
flora2 ?- ayse_aa4[ cgpa -> ?CGPA ].

?CGPA = 1.9141

1 solution(s) in 0.0470 seconds

Yes

flora2 ?- ayse_aa4[ status -> ?Status ].

?Status = satisfactory

1 solution(s) in 0.0000 seconds

Yes

flora2 ?- %run_registration ( std066161, 2014, fall, ?SuggestedCourseList ).

?SuggestedCourseList = [(SE_25751, cmpe323), (SE_25752, cmpe343), (SE_25753, cmpe371), (SE
_25754, cmpe321), (SE_25755, math322)]

1 solution(s) in 5.4210 seconds
```

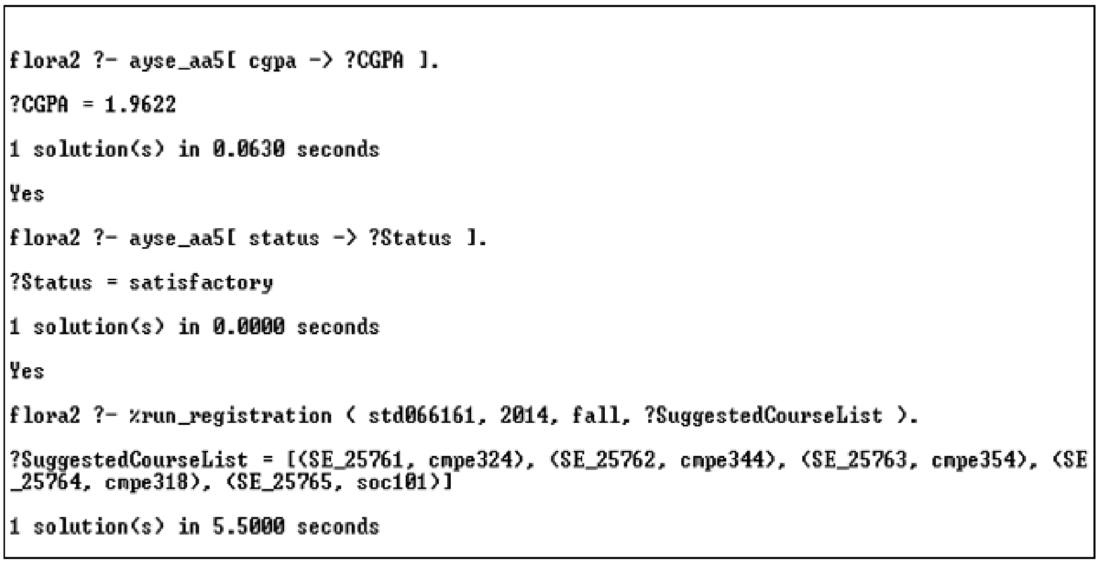
Figure 5.12: Suggested course list for the fifth semester

In addition to the first, second, third and fourth semester transcript, figure 5.13 shows the fifth semester transcript information of a student. Figure 5.14 shows the CGPA and status of the student for fifth semester and gives the list of courses suggested according to the current transcript information. Input is like a run of first semester, we give same parameters as a input. Our agent gives courses of sixth semester as a result.



```
ayse_aa5 : TranscriptSemesterData.
ayse_aa5[
        whichYear->2008,
        whichSemester->fall,
        registrationStatus -> registered
].
ayse_aa5 [
         transcriptEntry-> {
                          \#[syllabusEntry-> SE_25751,
                              letterGrade-> C]:TranscriptEntry,
                           \#[syllabusEntry-> SE_25752,
                              letterGrade-> D]:TranscriptEntry,
                           \#[syllabusEntry-> SE_25753,
                              letterGrade-> B]:TranscriptEntry,
                           \#[syllabusEntry-> SE_25754,
                              letterGrade-> C_MINUS]:TranscriptEntry,
                           \#[syllabusEntry-> SE_25755,
                              letterGrade-> B_PLUS]:TranscriptEntry
                                    }
           ].
```

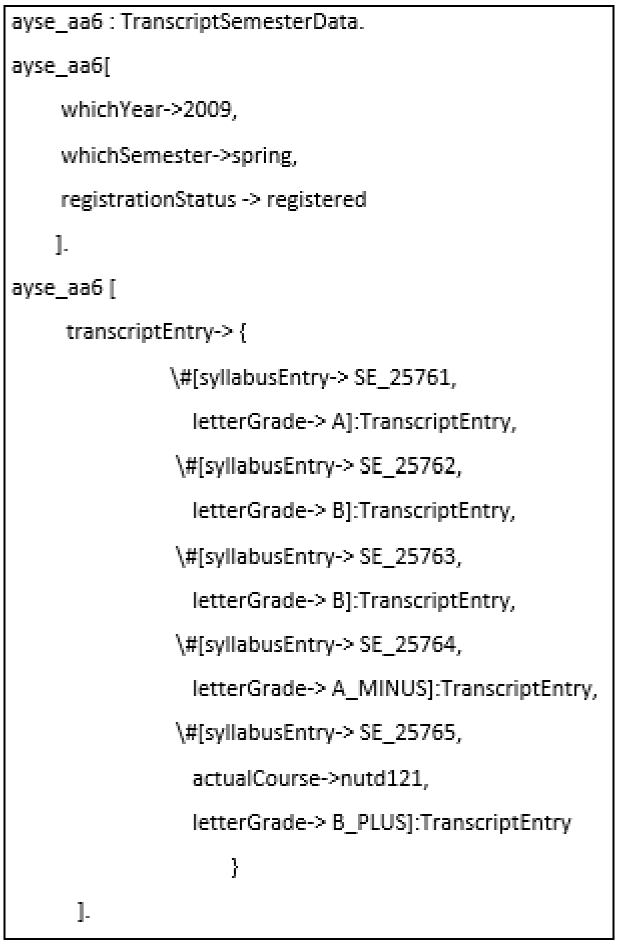
Figure 5.13: Transcript information for the fifth semester



```
flora2 ?- ayse_aa5[ cgpa -> ?CGPA ].

?CGPA = 1.9622

1 solution(s) in 0.0630 seconds

Yes

flora2 ?- ayse_aa5[ status -> ?Status ].

?Status = satisfactory

1 solution(s) in 0.0000 seconds

Yes

flora2 ?- %run_registration ( std066161, 2014, fall, ?SuggestedCourseList ).

?SuggestedCourseList = [(SE_25761, cmpe324), (SE_25762, cmpe344), (SE_25763, cmpe354), (SE
_25764, cmpe318), (SE_25765, soc101)]

1 solution(s) in 5.5000 seconds
```

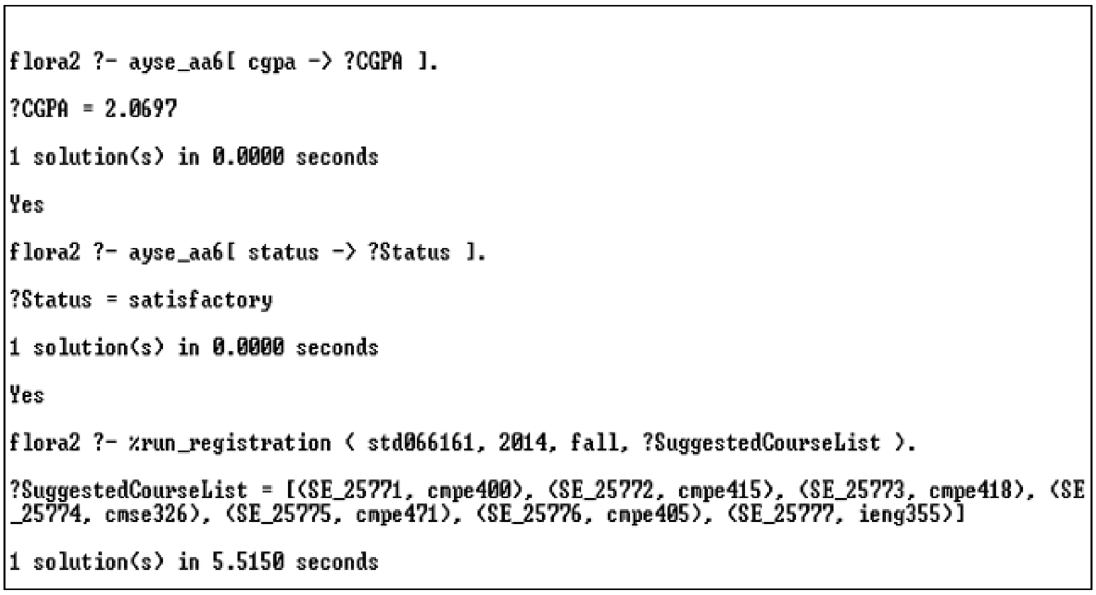
Figure 5.14: Suggested course list for the sixth semester

In addition to the first, second, third, fourth and fifth semester transcript, figure 5.15 shows the sixth semester transcript information of a student. Figure 5.16 shows the CGPA and status of the student for sixth semester and gives the list of courses suggested according to the current transcript information. Input is like a run of first semester, we give same parameters as a input. Our agent gives courses of seventh semester as a result.



```
ayse_aa6 : TranscriptSemesterData.
ayse_aa6[
        whichYear->2009,
        whichSemester->spring,
        registrationStatus -> registered
       ].
ayse_aa6 [
         transcriptEntry-> {
                    \#[syllabusEntry-> SE_25761,
                      letterGrade-> A]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25762,
                      letterGrade-> B]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25763,
                      letterGrade-> B]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25764,
                      letterGrade-> A_MINUS]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25765,
                      actualCourse->nutd121,
                      letterGrade-> B_PLUS]:TranscriptEntry
                          }
          ].
```

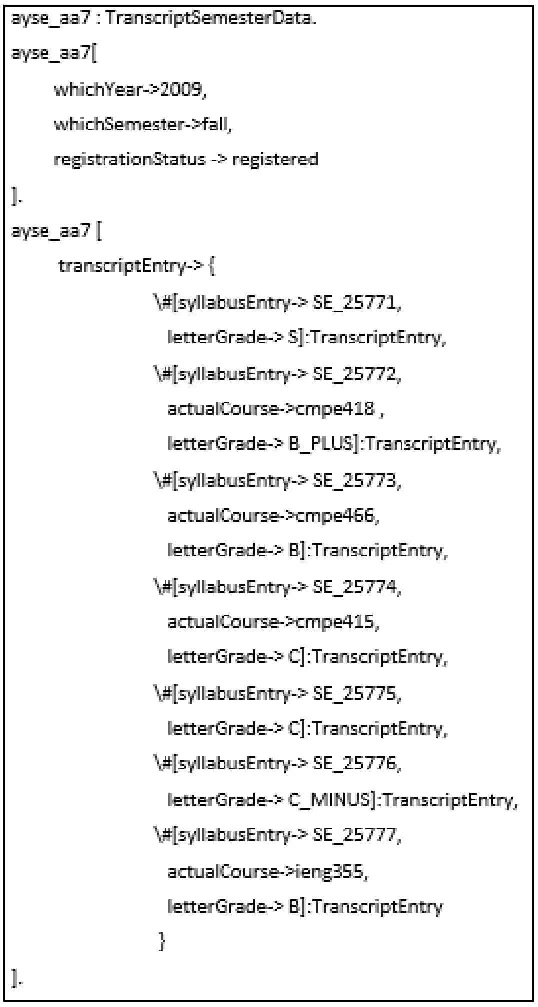
Figure 5.15: Transcript information for the sixth semester



```
flora2 ?- ayse_aa6[ cgpa -> ?CGPA ].
?CGPA = 2.0697
1 solution(s) in 0.0000 seconds
Yes
flora2 ?- ayse_aa6[ status -> ?Status ].
?Status = satisfactory
1 solution(s) in 0.0000 seconds
Yes
flora2 ?- %run_registration ( std066161, 2014, fall, ?SuggestedCourseList ).
?SuggestedCourseList = [(SE_25771, cmpe400), (SE_25772, cmpe415), (SE_25773, cmpe418), (SE
_25774, cmse326), (SE_25775, cmpe471), (SE_25776, cmpe405), (SE_25777, ieng355)]
1 solution(s) in 5.5150 seconds
```

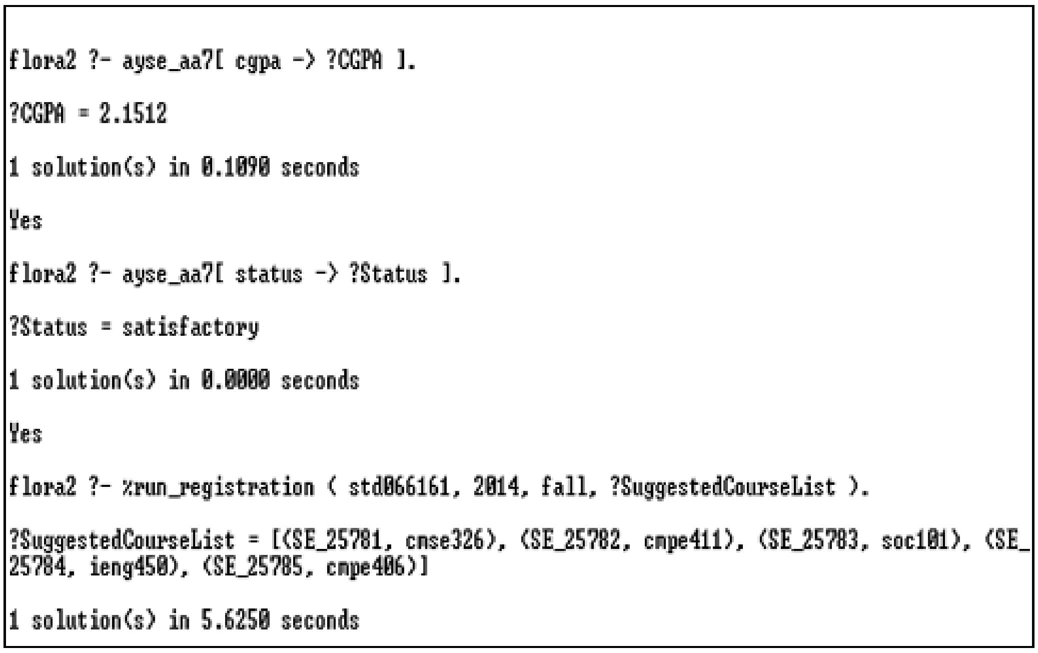
Figure 5.16: Suggested course list for the seventh semester

In addition to the first, second, third, fourth, fifth and sixth semester transcript, Figure 5.17 shows the seventh semester transcript information of a student. Figure 5.18 shows the CGPA and status of the student for seventh semester and gives the list of courses suggested according to the current transcript information. Input is like a run of first semester, we give same parameters as a input. Our agent gives courses of eighth semester as a result.



```
ayse_aa7 : TranscriptSemesterData.
ayse_aa7[
      whichYear->2009,
      whichSemester->fall,
      registrationStatus -> registered
].
ayse_aa7 [
       transcriptEntry-> {
                    \#[syllabusEntry-> SE_25771,
                      letterGrade-> S]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25772,
                      actualCourse->cmpe418 ,
                      letterGrade-> B_PLUS]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25773,
                      actualCourse->cmpe466,
                      letterGrade-> B]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25774,
                      actualCourse->cmpe415,
                      letterGrade-> C]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25775,
                      letterGrade-> C]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25776,
                      letterGrade-> C_MINUS]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25777,
                      actualCourse->ieng355,
                      letterGrade-> B]:TranscriptEntry
                     }
].
```

Figure 5.17: Transcript information for the seventh semester



```
flora2 ?- ayse_aa7[ cgpa -> ?CGPA ].

?CGPA = 2.1512

1 solution(s) in 0.1090 seconds

Yes

flora2 ?- ayse_aa7[ status -> ?Status ].

?Status = satisfactory

1 solution(s) in 0.0000 seconds

Yes

flora2 ?- %run_registration ( std066161, 2014, fall, ?SuggestedCourseList ).

?SuggestedCourseList = [(SE_25781, cmse326), (SE_25782, cmpe411), (SE_25783, soc101), (SE_
25784, ieng450), (SE_25785, cmpe406)]

1 solution(s) in 5.6250 seconds
```

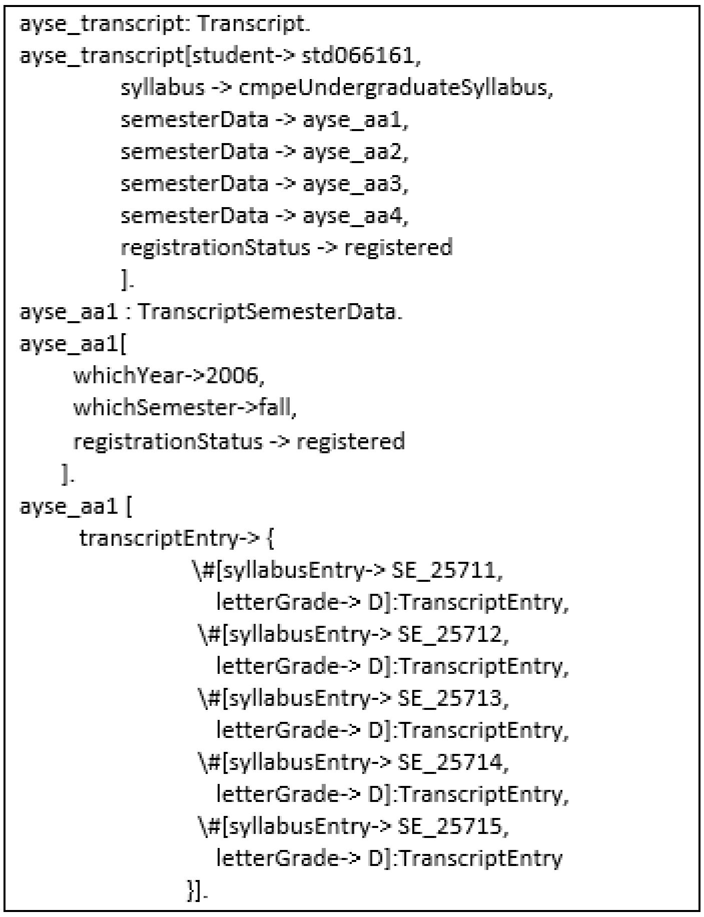
Figure 5.18: Suggested course list for the eighth semester

5.3 Registering the Student According to His/Her Status

The student's success status are evaluated every semester. After second semester the student's CGPA should be in some range as explained in chapter 2. If student's CGPA is not in the given range, then his/her success status will change accordingly.

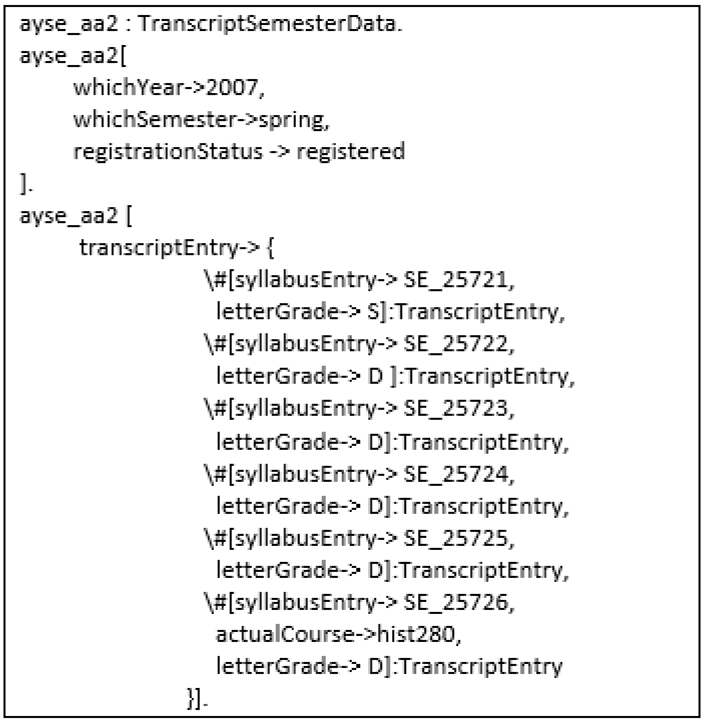
5.3.1 Registering a Student Who is On Probation Status

In this section, we will show sample runs for different types of status. Firstly, if a student's status is on probation, then he/she can register only two new courses. According to status table of Eastern Mediterranean University, if student's CGPA is between 1.00 and 1.50 at the end of fourth semester then student will be in status of on probation. Figure 5.19, 5.20, 5.21, and 5.22 contains the transcript of a student who is on probation for the first, second, third and fourth semester respectively. In figure 5.13 student's CGPA is 1.40 at the end of fourth semester. Therefore, her status is on probation. When we run our agent, it will give two new courses and three already taken courses and passed with low grade such as D as an output. In our example, it gives cmpe323 and cmpe343 whose reference codes are 25751 and 25752 respectively as new courses from fifth semester. Also, it gives cmpe101, math163 and engl191 whose reference codes are 25711, 25712 and 25713 respectively as an already taken courses. If we check her transcript, we will see that she passed these courses with D grade.



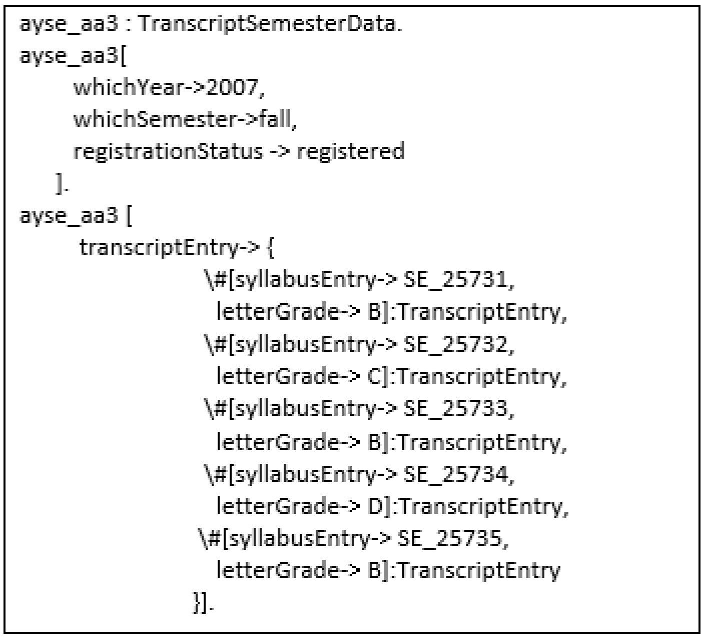
```
ayse_transcript: Transcript.
ayse_transcript[student-> std066161,
            syllabus -> cmpeUndergraduateSyllabus,
            semesterData -> ayse_aa1,
            semesterData -> ayse_aa2,
            semesterData -> ayse_aa3,
            semesterData -> ayse_aa4,
            registrationStatus -> registered
            ].
ayse_aa1 : TranscriptSemesterData.
ayse_aa1[
       whichYear->2006,
       whichSemester->fall,
       registrationStatus -> registered
      ].
ayse_aa1 [
        transcriptEntry-> {
                      \#[syllabusEntry-> SE_25711,
                         letterGrade-> D]:TranscriptEntry,
                      \#[syllabusEntry-> SE_25712,
                         letterGrade-> D]:TranscriptEntry,
                      \#[syllabusEntry-> SE_25713,
                         letterGrade-> D]:TranscriptEntry,
                      \#[syllabusEntry-> SE_25714,
                         letterGrade-> D]:TranscriptEntry,
                      \#[syllabusEntry-> SE_25715,
                         letterGrade-> D]:TranscriptEntry
                     }].
```

Figure 5.19: Transcript information for the first semester to test on probation status



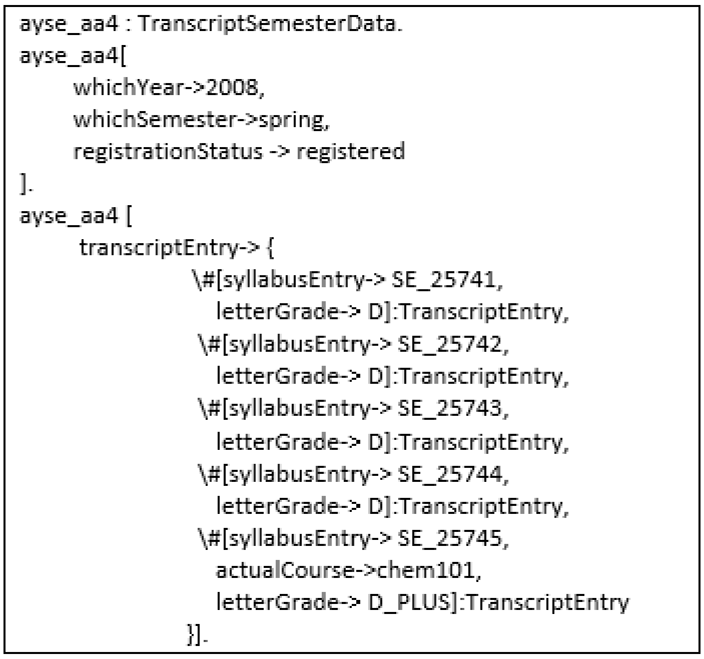
```
ayse_aa2 : TranscriptSemesterData.
ayse_aa2[
       whichYear->2007,
       whichSemester->spring,
       registrationStatus -> registered
].
ayse_aa2 [
        transcriptEntry-> {
                        \#[syllabusEntry-> SE_25721,
                          letterGrade-> S]:TranscriptEntry,
                        \#[syllabusEntry-> SE_25722,
                          letterGrade-> D ]:TranscriptEntry,
                        \#[syllabusEntry-> SE_25723,
                          letterGrade-> D]:TranscriptEntry,
                        \#[syllabusEntry-> SE_25724,
                          letterGrade-> D]:TranscriptEntry,
                        \#[syllabusEntry-> SE_25725,
                          letterGrade-> D]:TranscriptEntry,
                        \#[syllabusEntry-> SE_25726,
                          actualCourse->hist280,
                          letterGrade-> D]:TranscriptEntry
                      }].
```

Figure 5.20: Transcript information for the second semester to test on probation status



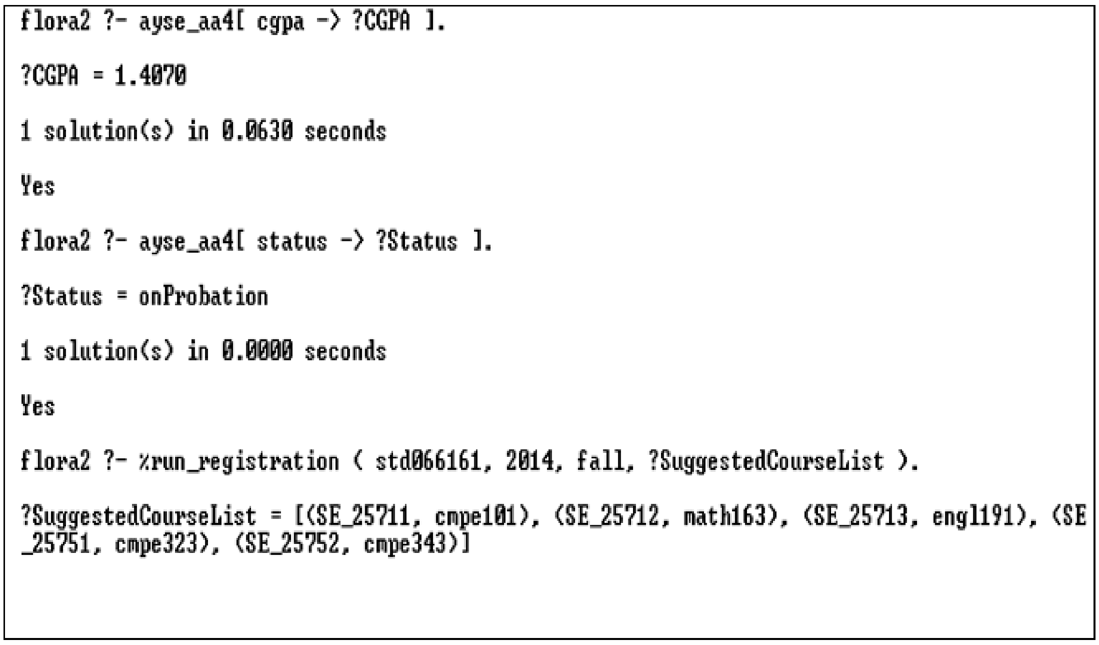
```
ayse_aa3 : TranscriptSemesterData.
ayse_aa3[
       whichYear->2007,
       whichSemester->fall,
       registrationStatus -> registered
     ].
ayse_aa3 [
        transcriptEntry-> {
                        \#[syllabusEntry-> SE_25731,
                          letterGrade-> B]:TranscriptEntry,
                        \#[syllabusEntry-> SE_25732,
                          letterGrade-> C]:TranscriptEntry,
                        \#[syllabusEntry-> SE_25733,
                          letterGrade-> B]:TranscriptEntry,
                        \#[syllabusEntry-> SE_25734,
                          letterGrade-> D]:TranscriptEntry,
                       \#[syllabusEntry-> SE_25735,
                          letterGrade-> B]:TranscriptEntry
                      }].
```

Figure 5.21: Transcript information for the third semester to test on probation status



```
ayse_aa4 : TranscriptSemesterData.
ayse_aa4[
      whichYear->2008,
      whichSemester->spring,
      registrationStatus -> registered
].
ayse_aa4 [
       transcriptEntry-> {
                   \#[syllabusEntry-> SE_25741,
                     letterGrade-> D]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25742,
                     letterGrade-> D]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25743,
                     letterGrade-> D]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25744,
                     letterGrade-> D]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25745,
                     actualCourse->chem101,
                     letterGrade-> D_PLUS]:TranscriptEntry
                  }].
```

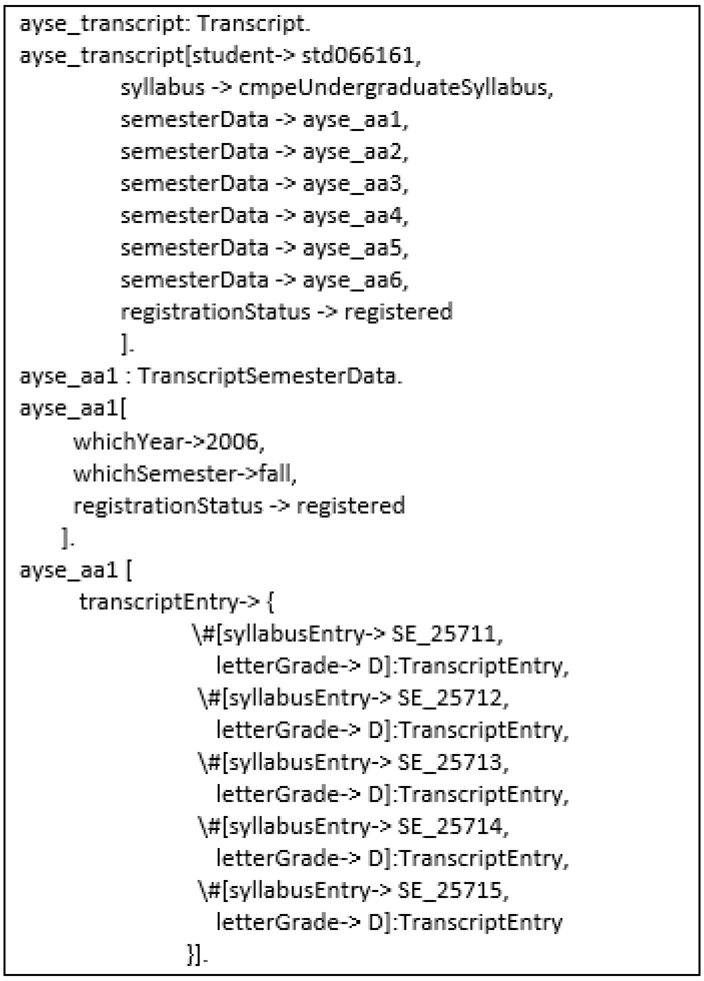
Figure 5.22: Transcript information for the fourth semester to test on probation status



```
flora2 ?- ayse_aa4[ cgpa -> ?CGPA ].

?CGPA = 1.4070

1 solution(s) in 0.0630 seconds

Yes

flora2 ?- ayse_aa4[ status -> ?Status ].

?Status = onProbation

1 solution(s) in 0.0000 seconds

Yes

flora2 ?- %run_registration ( std066161, 2014, fall, ?SuggestedCourseList ).

?SuggestedCourseList = [(SE_25711, cmpe101), (SE_25712, math163), (SE_25713, engl191), (SE
_25751, cmpe323), (SE_25752, cmpe343)]
```

Figure 5.23: Status of the student is on probation

Sometimes student may be unsuccessful in some semesters. Then he/she may be successful in other semester but this success is not affect his/her CGPA enough. Suppose there is a student that her status is satisfactory for the first semester, on probation for the second, third and fourth semester and unsatisfactory for the fifth semester. Figure 5.24, 5.25, 5.26, 5.27, 5.28, and 5.29 contains the transcript information of a student who is in such situation for the first, second, third, fourth, fifth and sixth semester respectively. In the sixth semester, even if her CGPA is 3.84, her status will not be satisfactory. It will arise from unsatisfactory to on probation. Therefore, even if her GPA is greater than 3.00, she is not able to take an extra course in the seventh semester. On the contrary, she will take only two new courses as well as summer training course from seventh semester.



```
ayse_transcript: Transcript.
ayse_transcript[student-> std066161,
            syllabus -> cmpeUndergraduateSyllabus,
            semesterData -> ayse_aa1,
            semesterData -> ayse_aa2,
            semesterData -> ayse_aa3,
            semesterData -> ayse_aa4,
            semesterData -> ayse_aa5,
            semesterData -> ayse_aa6,
            registrationStatus -> registered
            ].
ayse_aa1 : TranscriptSemesterData.
ayse_aa1[
      whichYear->2006,
      whichSemester->fall,
      registrationStatus -> registered
     ].
ayse_aa1 [
       transcriptEntry-> {
                    \#[syllabusEntry-> SE_25711,
                       letterGrade-> D]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25712,
                       letterGrade-> D]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25713,
                       letterGrade-> D]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25714,
                       letterGrade-> D]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25715,
                       letterGrade-> D]:TranscriptEntry
                    }].
```

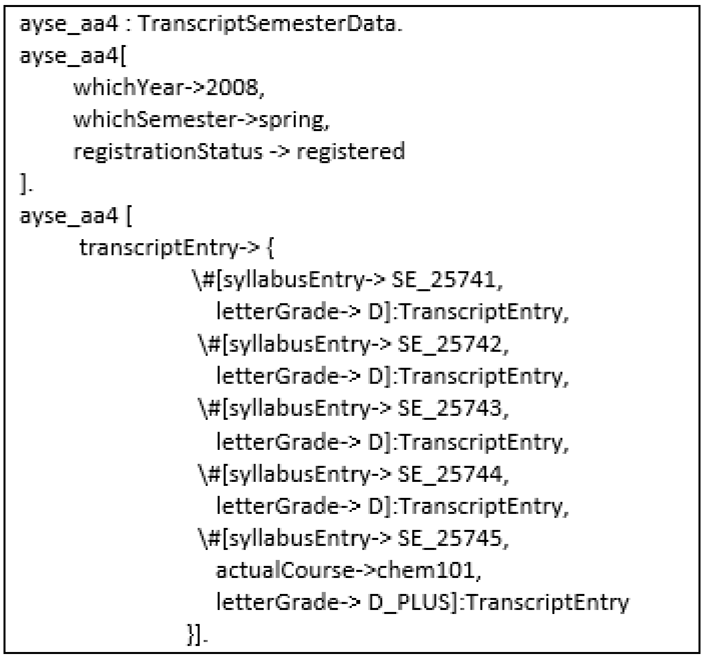
Figure 5.24: Transcript information for the first semester (honor but status is on probation)

```
ayse_aa2 : TranscriptSemesterData.
ayse_aa2[
        whichYear->2007,
        whichSemester->spring,
        registrationStatus -> registered
].
ayse_aa2 [
         transcriptEntry-> {
                         \#[syllabusEntry-> SE_25721,
                           letterGrade-> S]:TranscriptEntry,
                         \#[syllabusEntry-> SE_25722,
                           letterGrade-> D ]:TranscriptEntry,
                         \#[syllabusEntry-> SE_25723,
                           letterGrade-> D]:TranscriptEntry,
                         \#[syllabusEntry-> SE_25724,
                           letterGrade-> D]:TranscriptEntry,
                         \#[syllabusEntry-> SE_25725,
                           letterGrade-> D]:TranscriptEntry,
                         \#[syllabusEntry-> SE_25726,
                           actualCourse->hist280,
                           letterGrade-> D]:TranscriptEntry
                        }].
```

Figure 5.25: Transcript information for the second semester (honor but status is on probation)

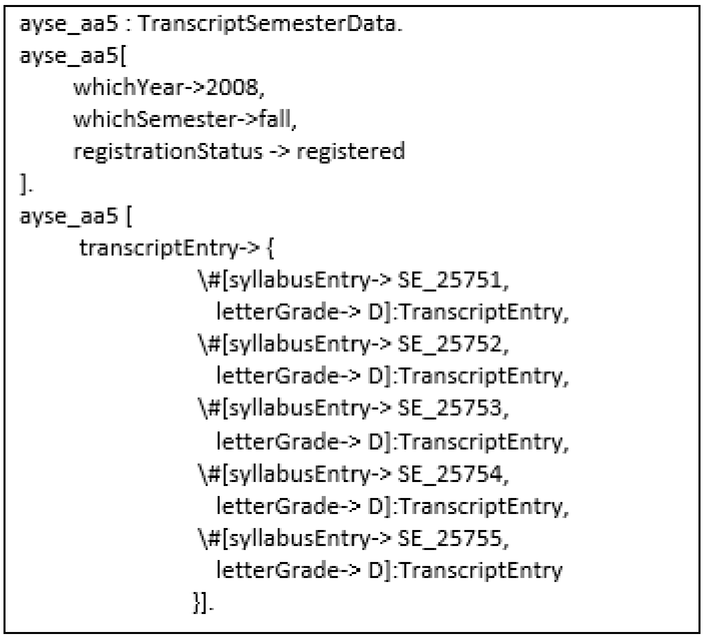
```
ayse_aa3 : TranscriptSemesterData.
ayse_aa3[
        whichYear->2007,
        whichSemester->fall,
        registrationStatus -> registered
     ].
ayse_aa3 [
         transcriptEntry-> {
                         \#[syllabusEntry-> SE_25731,
                           letterGrade-> D]:TranscriptEntry,
                         \#[syllabusEntry-> SE_25732,
                           letterGrade-> D]:TranscriptEntry,
                         \#[syllabusEntry-> SE_25733,
                           letterGrade-> D]:TranscriptEntry,
                         \#[syllabusEntry-> SE_25734,
                           letterGrade-> D]:TranscriptEntry,
                        \#[syllabusEntry-> SE_25735,
                           letterGrade-> D]:TranscriptEntry
                         }].
```

Figure 5.26: Transcript information for the third semester (honor but status is on probation)



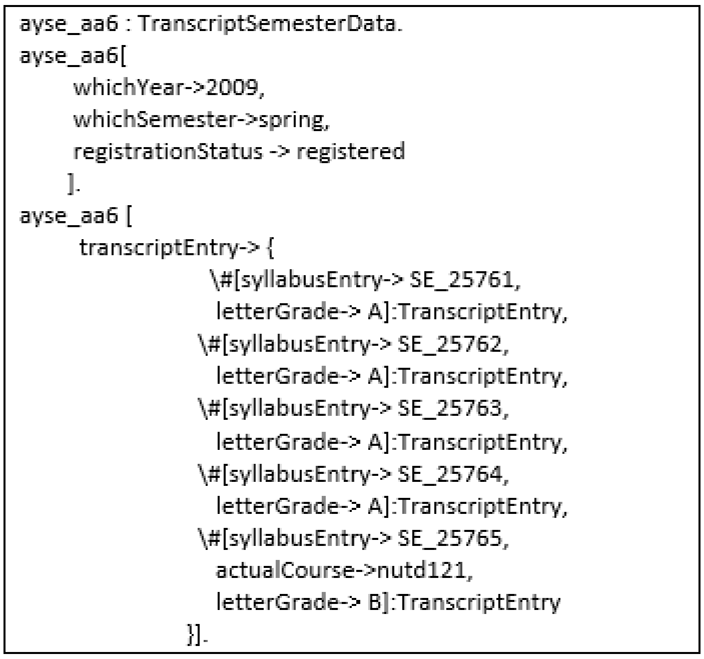
```
ayse_aa4 : TranscriptSemesterData.
ayse_aa4[
        whichYear->2008,
        whichSemester->spring,
        registrationStatus -> registered
].
ayse_aa4 [
         transcriptEntry-> {
                          \#[syllabusEntry-> SE_25741,
                            letterGrade-> D]:TranscriptEntry,
                          \#[syllabusEntry-> SE_25742,
                            letterGrade-> D]:TranscriptEntry,
                          \#[syllabusEntry-> SE_25743,
                            letterGrade-> D]:TranscriptEntry,
                          \#[syllabusEntry-> SE_25744,
                            letterGrade-> D]:TranscriptEntry,
                          \#[syllabusEntry-> SE_25745,
                            actualCourse->chem101,
                            letterGrade-> D_PLUS]:TranscriptEntry
                         }].
```

Figure 5.27: Transcript information for the fourth semester (honor but status is on probation)



```
ayse_aa5 : TranscriptSemesterData.
ayse_aa5[
        whichYear->2008,
        whichSemester->fall,
        registrationStatus -> registered
].
ayse_aa5 [
         transcriptEntry-> {
                          \#[syllabusEntry-> SE_25751,
                            letterGrade-> D]:TranscriptEntry,
                          \#[syllabusEntry-> SE_25752,
                            letterGrade-> D]:TranscriptEntry,
                          \#[syllabusEntry-> SE_25753,
                            letterGrade-> D]:TranscriptEntry,
                          \#[syllabusEntry-> SE_25754,
                            letterGrade-> D]:TranscriptEntry,
                          \#[syllabusEntry-> SE_25755,
                            letterGrade-> D]:TranscriptEntry
                         }].
```

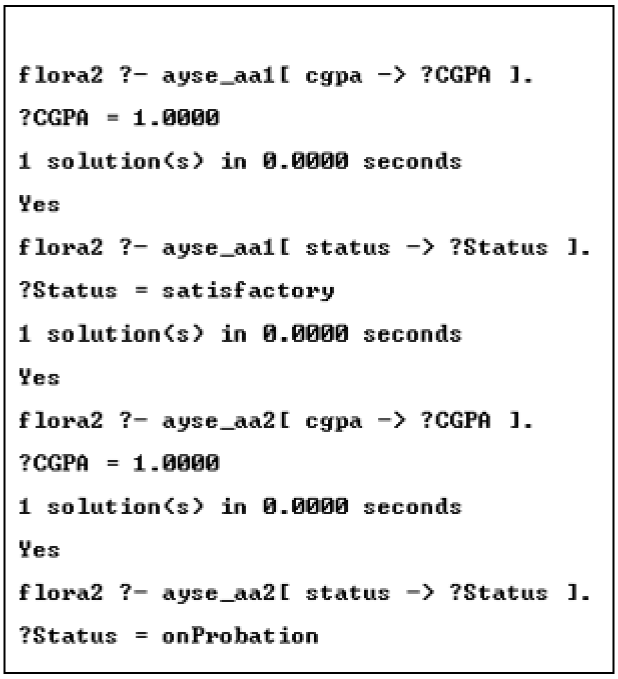
Figure 5.28: Transcript information for the fifth semester (honor but status is on probation)



```
ayse_aa6 : TranscriptSemesterData.
ayse_aa6[
      whichYear->2009,
      whichSemester->spring,
      registrationStatus -> registered
      ].
ayse_aa6 [
       transcriptEntry-> {
                      \#[syllabusEntry-> SE_25761,
                        letterGrade-> A]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25762,
                        letterGrade-> A]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25763,
                        letterGrade-> A]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25764,
                        letterGrade-> A]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25765,
                        actualCourse->nutd121,
                        letterGrade-> B]:TranscriptEntry
                    }].
```

Figure 5.29: Transcript information for the sixth semester (honor but status is on probation)

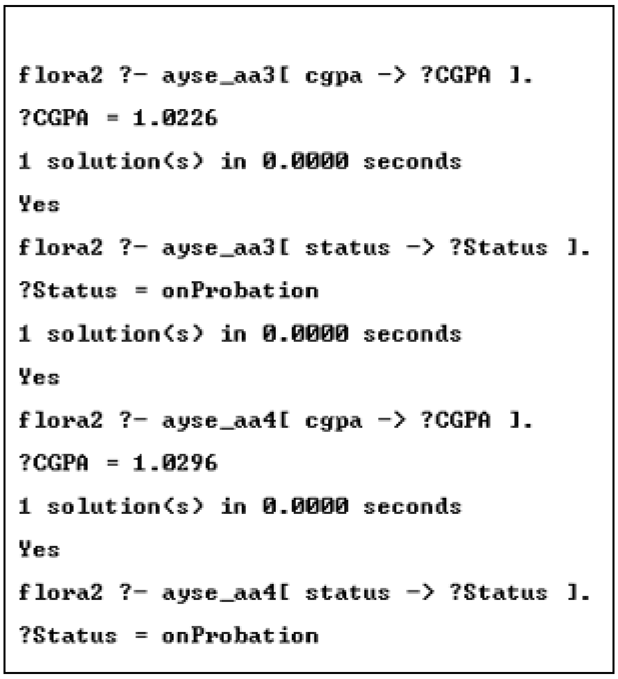
Figure 5.30 shows the CGPA and status of the first and second semester according to the transcript given in the table 4. Her status is satisfactory for the first semester and on probation for the second semester.



```
flora2 ?- ayse_aa1[ cgpa -> ?CGPA ].
?CGPA = 1.0000
1 solution(s) in 0.0000 seconds
Yes
flora2 ?- ayse_aa1[ status -> ?Status ].
?Status = satisfactory
1 solution(s) in 0.0000 seconds
Yes
flora2 ?- ayse_aa2[ cgpa -> ?CGPA ].
?CGPA = 1.0000
1 solution(s) in 0.0000 seconds
Yes
flora2 ?- ayse_aa2[ status -> ?Status ].
?Status = onProbation
```

Figure 5.30: Cgpa and status for the first and second semesters

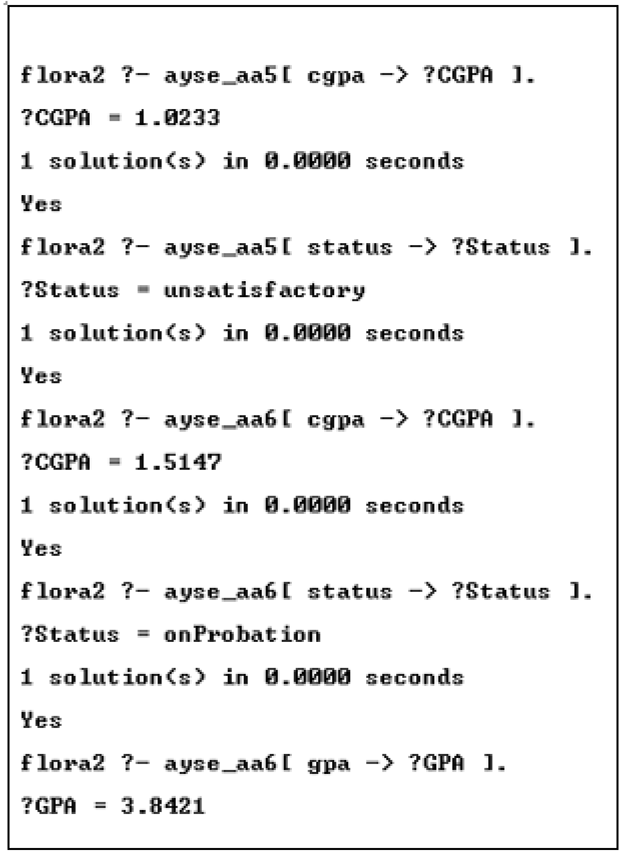
Figure 5.31 shows the CGPA and status of the third and fourth semester according to the transcript given in the table 4. Her status is on probation for the both third and fourth semester.



```
flora2 ?- ayse_aa3[ cgpa -> ?CGPA ].
?CGPA = 1.0226
1 solution(s) in 0.0000 seconds
Yes
flora2 ?- ayse_aa3[ status -> ?Status ].
?Status = onProbation
1 solution(s) in 0.0000 seconds
Yes
flora2 ?- ayse_aa4[ cgpa -> ?CGPA ].
?CGPA = 1.0296
1 solution(s) in 0.0000 seconds
Yes
flora2 ?- ayse_aa4[ status -> ?Status ].
?Status = onProbation
```

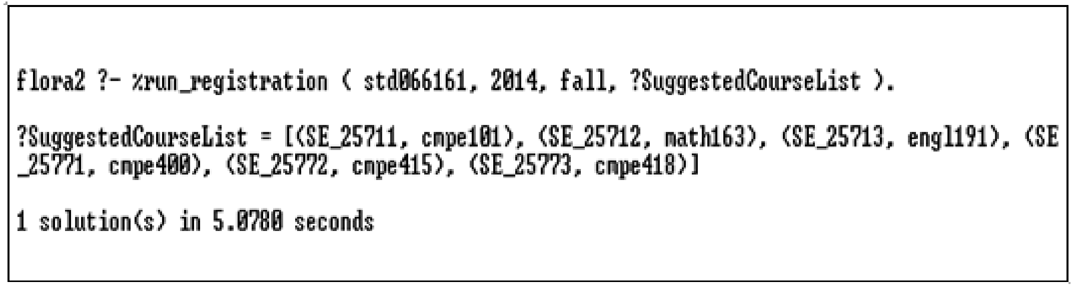
Figure 5.31: Cgpa and status for the third and fourth semesters

Figure 5.32 shows the CGPA and status of the fifth and sixth semester according to the transcript given in table 4. Her status is unsatisfactory for the fifth semester and on probation for the sixth semester. Also, it shows that the student's GPA is 3.84. This means that the student can take one extra course. However, her status is onprobation so, she cannot take an extra course. Figure 5.33 shows the list of courses suggested for this situation.



```
flora2 ?- ayse_aa5[ cgpa -> ?CGPA ].
?CGPA = 1.0233
1 solution(s) in 0.0000 seconds
Yes
flora2 ?- ayse_aa5[ status -> ?Status ].
?Status = unsatisfactory
1 solution(s) in 0.0000 seconds
Yes
flora2 ?- ayse_aa6[ cgpa -> ?CGPA ].
?CGPA = 1.5147
1 solution(s) in 0.0000 seconds
Yes
flora2 ?- ayse_aa6[ status -> ?Status ].
?Status = onProbation
1 solution(s) in 0.0000 seconds
Yes
flora2 ?- ayse_aa6[ gpa -> ?GPA ].
?GPA = 3.8421
```

Figure 5.32: Cgpa and status for the fifth and sixth semesters

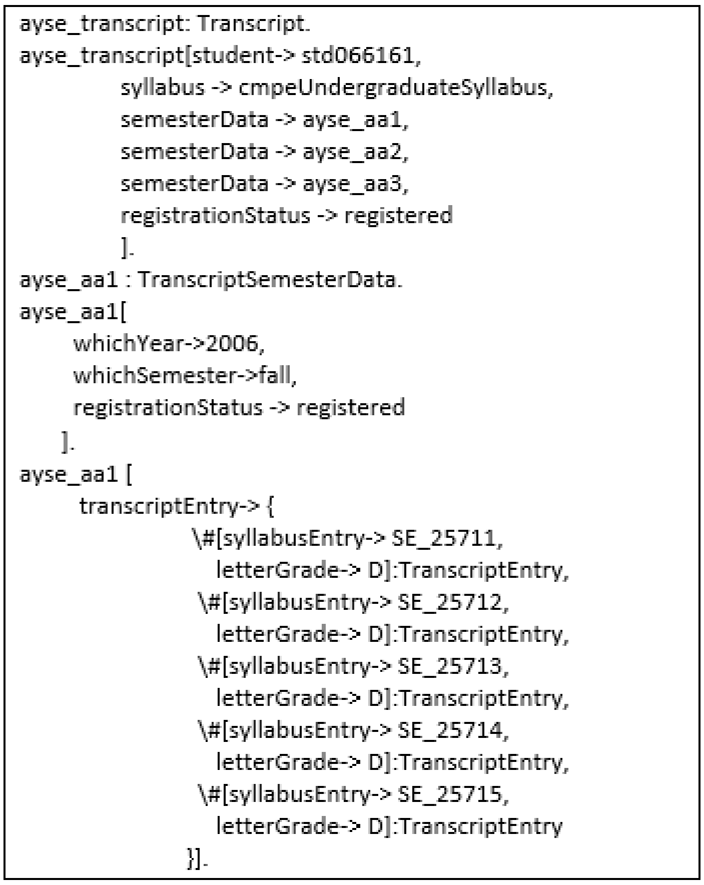


```
flora2 ?- %run_registration ( std066161, 2014, fall, ?SuggestedCourseList ).

?SuggestedCourseList = [(SE_25711, cmpe101), (SE_25712, math163), (SE_25713, engl191), (SE
_25771, cmpe400), (SE_25772, cmpe415), (SE_25773, cmpe418)]

1 solution(s) in 5.0780 seconds
```

Figure 5.33: High honor but status of the student is on probation

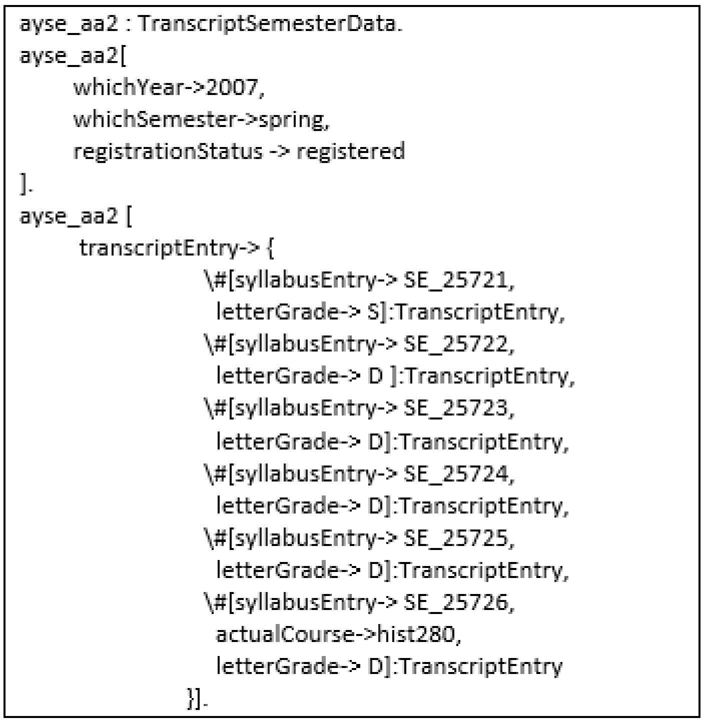
5.3.2 Registering a Student Who is in Unsatisfactory Status

Another student status is unsatisfactory. In this status student cannot take any new courses. Figure 5.34, 5.35, and 5.36 contains the transcript information of a student who is in unsatisfactory status for the first, second and three semester respectively. Figure 5.37 shows the student's CGPA and status for the third semester. She has 0.92 CGPA at the end of the third semester. So, her status is unsatisfactory. Our agent gives her five already passed courses with low grade in the last semesters.



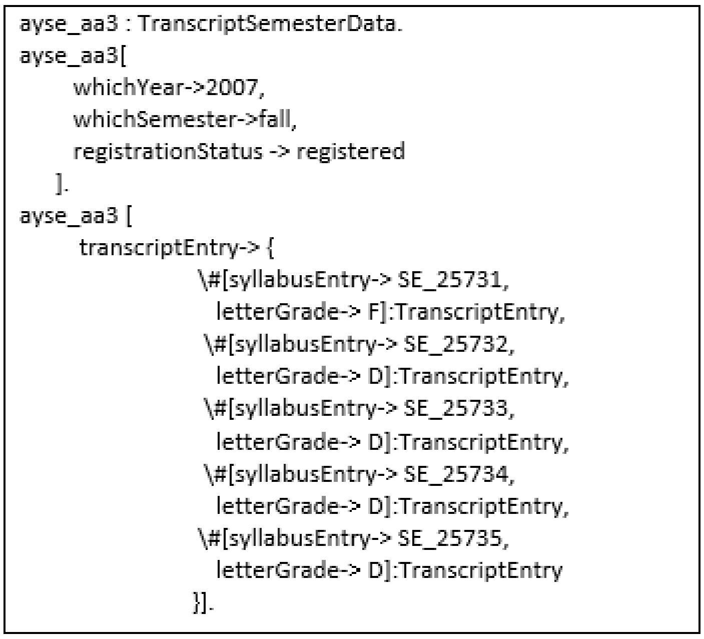
```
ayse_transcript: Transcript.
ayse_transcript[student-> std066161,
            syllabus -> cmpeUndergraduateSyllabus,
            semesterData -> ayse_aa1,
            semesterData -> ayse_aa2,
            semesterData -> ayse_aa3,
            registrationStatus -> registered
            ].
ayse_aa1 : TranscriptSemesterData.
ayse_aa1[
      whichYear->2006,
      whichSemester->fall,
      registrationStatus -> registered
     ].
ayse_aa1 [
       transcriptEntry-> {
                    \#[syllabusEntry-> SE_25711,
                       letterGrade-> D]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25712,
                       letterGrade-> D]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25713,
                       letterGrade-> D]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25714,
                       letterGrade-> D]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25715,
                       letterGrade-> D]:TranscriptEntry
                    }].
```

Figure 5.34: Transcript information for first semester to test unsatisfactory status



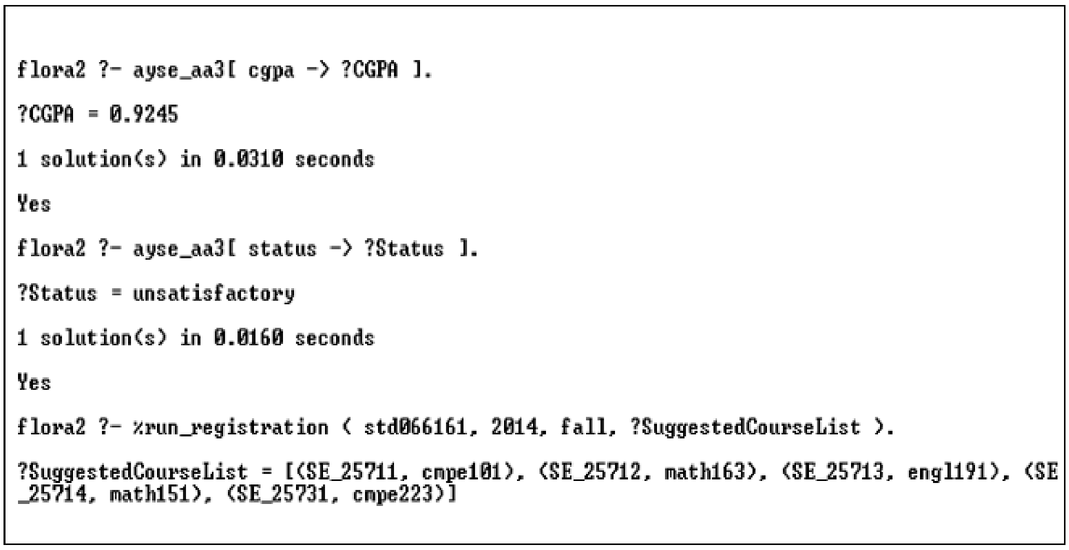
```
ayse_aa2 : TranscriptSemesterData.
ayse_aa2[
      whichYear->2007,
      whichSemester->spring,
      registrationStatus -> registered
].
ayse_aa2 [
       transcriptEntry-> {
                     \#[syllabusEntry-> SE_25721,
                       letterGrade-> S]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25722,
                       letterGrade-> D ]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25723,
                       letterGrade-> D]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25724,
                       letterGrade-> D]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25725,
                       letterGrade-> D]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25726,
                       actualCourse->hist280,
                       letterGrade-> D]:TranscriptEntry
                    }].
```

Figure 5.35: Transcript information for second semester to test unsatisfactory status



```
ayse_aa3 : TranscriptSemesterData.
ayse_aa3[
      whichYear->2007,
      whichSemester->fall,
      registrationStatus -> registered
    ].
ayse_aa3 [
       transcriptEntry-> {
                   \#[syllabusEntry-> SE_25731,
                     letterGrade-> F]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25732,
                     letterGrade-> D]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25733,
                     letterGrade-> D]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25734,
                     letterGrade-> D]:TranscriptEntry,
                   \#[syllabusEntry-> SE_25735,
                     letterGrade-> D]:TranscriptEntry
                   }].
```

Figure 5.36 Transcript information for third semester to test unsatisfactory status

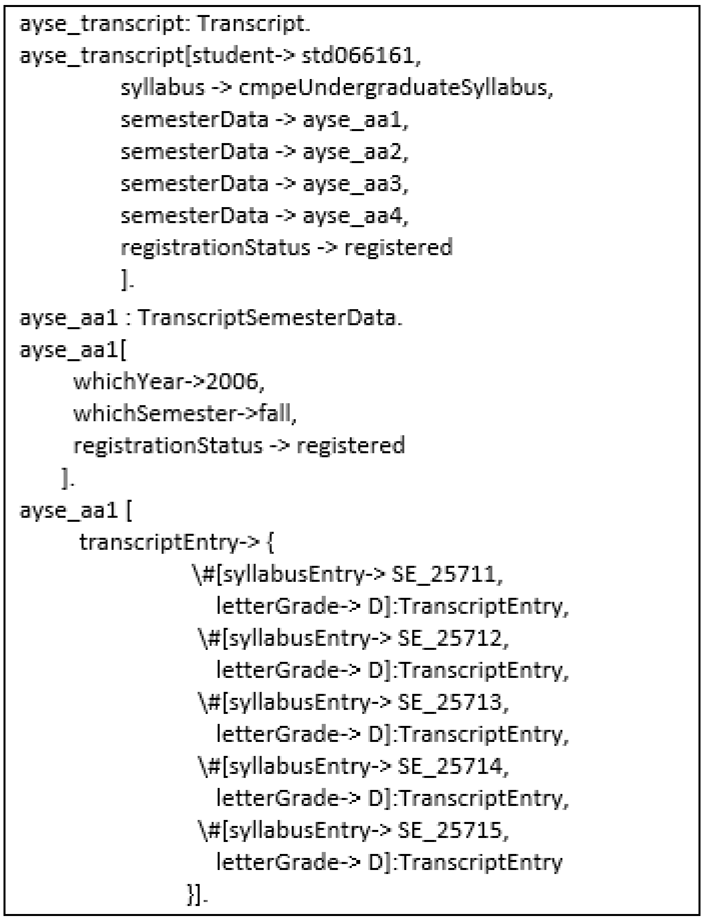


```
flora2 ?- ayse_aa3[ cgpa -> ?CGPA ].

?CGPA = 0.9245

1 solution(s) in 0.0310 seconds

Yes

flora2 ?- ayse_aa3[ status -> ?Status ].

?Status = unsatisfactory

1 solution(s) in 0.0160 seconds

Yes

flora2 ?- %run_registration ( std066161, 2014, fall, ?SuggestedCourseList ).

?SuggestedCourseList = [(SE_25711, cmpe101), (SE_25712, math163), (SE_25713, engl191), (SE
_25714, math151), (SE_25731, cmpe223)]
```

Figure 5.37: Status of the student is unsatisfactory

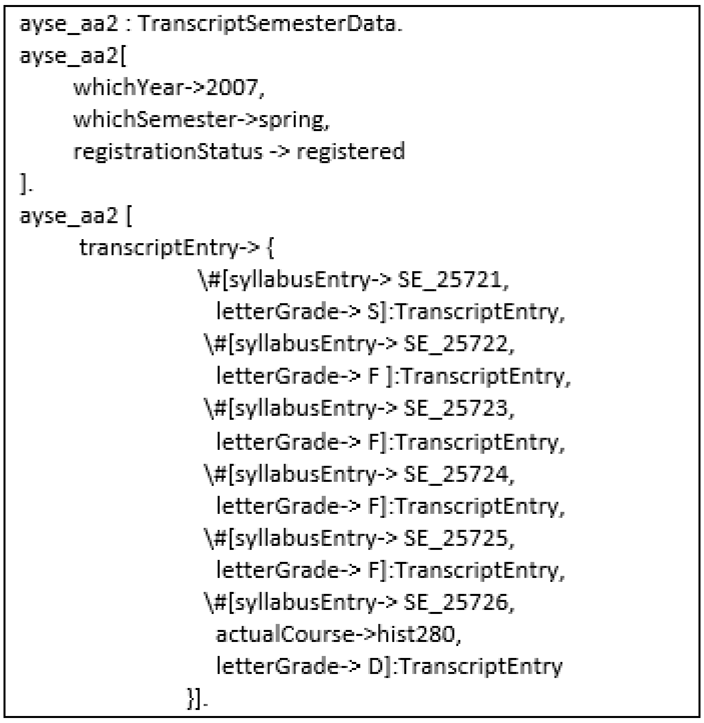
5.3.3 Case of a Dismissed Student

Lastly, there is a status of dismissed. If a student has status of unsatisfactory in three semesters consecutively, department administrator has the right to dismiss the student from department. Detail of this rule is given in chapter 2. Figure 5.38, 5.39, 5.40, and 5.41 contains the transcript information of a student who is in dismissed status of the first, second, third and fourth semester respectively. Figure 5.42 shows the CGPA and student's status that is unsatisfactory for the second and third semester. As shown in figure 5.43 the student is at the end of fourth semester and her CGPA is 0.76 so, her status is dismissed and our agent says no to her registration request because of her status.



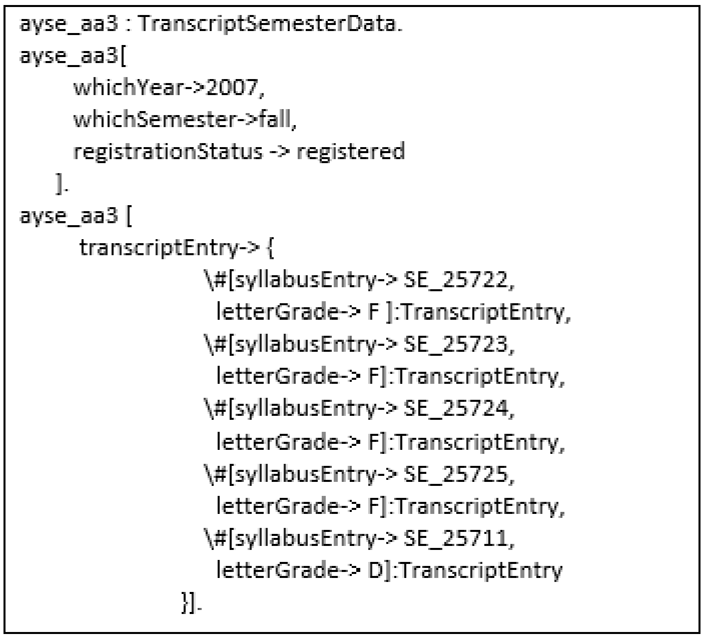
```
ayse_transcript: Transcript.
ayse_transcript[student-> std066161,
            syllabus -> cmpeUndergraduateSyllabus,
            semesterData -> ayse_aa1,
            semesterData -> ayse_aa2,
            semesterData -> ayse_aa3,
            semesterData -> ayse_aa4,
            registrationStatus -> registered
            ].
ayse_aa1 : TranscriptSemesterData.
ayse_aa1[
      whichYear->2006,
      whichSemester->fall,
      registrationStatus -> registered
     ].
ayse_aa1 [
       transcriptEntry-> {
                    \#[syllabusEntry-> SE_25711,
                       letterGrade-> D]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25712,
                       letterGrade-> D]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25713,
                       letterGrade-> D]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25714,
                       letterGrade-> D]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25715,
                       letterGrade-> D]:TranscriptEntry
                    }].
```

Figure 5.38: Transcript information for first semester to test dismissed status



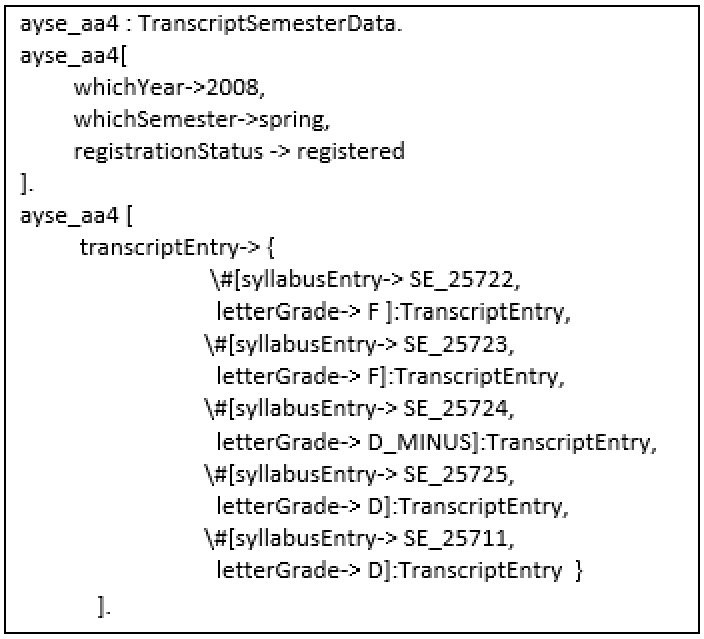
```
ayse_aa2 : TranscriptSemesterData.
ayse_aa2[
        whichYear->2007,
        whichSemester->spring,
        registrationStatus -> registered
].
ayse_aa2 [
         transcriptEntry-> {
                         \#[syllabusEntry-> SE_25721,
                           letterGrade-> S]:TranscriptEntry,
                          \#[syllabusEntry-> SE_25722,
                           letterGrade-> F ]:TranscriptEntry,
                          \#[syllabusEntry-> SE_25723,
                           letterGrade-> F]:TranscriptEntry,
                          \#[syllabusEntry-> SE_25724,
                           letterGrade-> F]:TranscriptEntry,
                          \#[syllabusEntry-> SE_25725,
                           letterGrade-> F]:TranscriptEntry,
                          \#[syllabusEntry-> SE_25726,
                           actualCourse->hist280,
                           letterGrade-> D]:TranscriptEntry
                       }].
```

Figure 5.39: Transcript information for second semester to test dismissed status



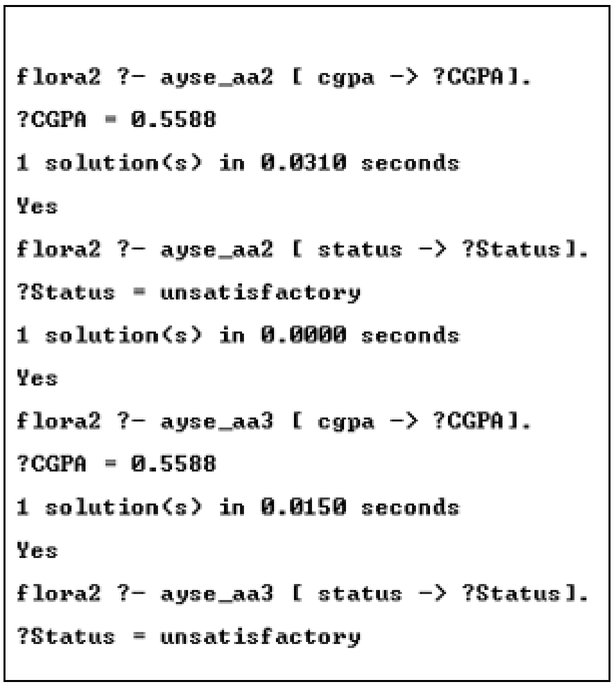
```
ayse_aa3 : TranscriptSemesterData.
ayse_aa3[
        whichYear->2007,
        whichSemester->fall,
        registrationStatus -> registered
     ].
ayse_aa3 [
         transcriptEntry-> {
                          \#[syllabusEntry-> SE_25722,
                           letterGrade-> F ]:TranscriptEntry,
                          \#[syllabusEntry-> SE_25723,
                           letterGrade-> F]:TranscriptEntry,
                          \#[syllabusEntry-> SE_25724,
                           letterGrade-> F]:TranscriptEntry,
                          \#[syllabusEntry-> SE_25725,
                           letterGrade-> F]:TranscriptEntry,
                          \#[syllabusEntry-> SE_25711,
                           letterGrade-> D]:TranscriptEntry
                      }].
```

Figure 5.40: Transcript information for third semester to test dismissed status



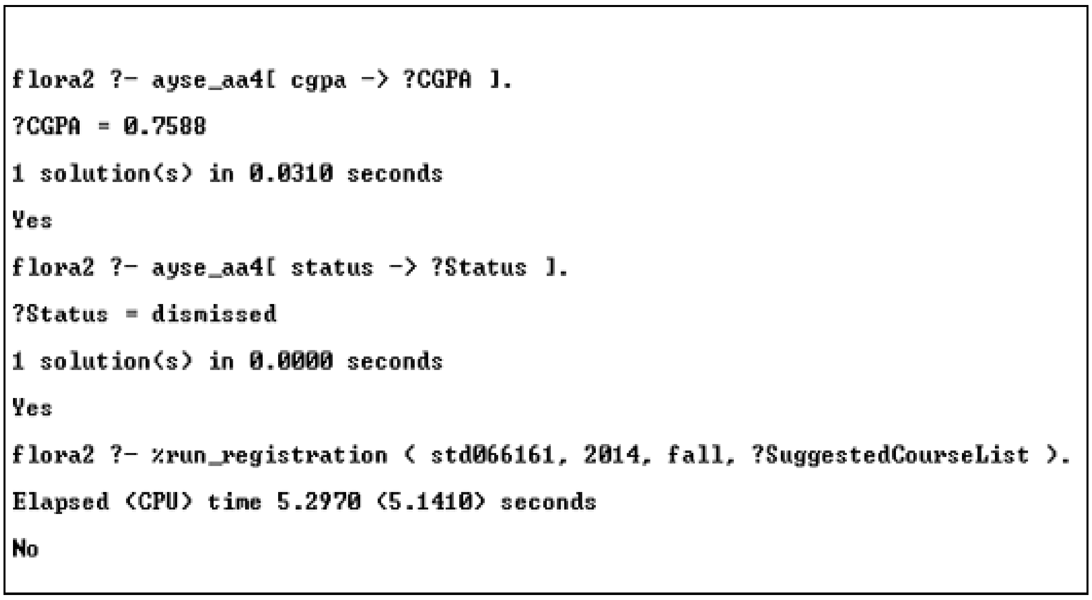
```
ayse_aa4 : TranscriptSemesterData.
ayse_aa4[
      whichYear->2008,
      whichSemester->spring,
      registrationStatus -> registered
].
ayse_aa4 [
       transcriptEntry-> {
                    \#[syllabusEntry-> SE_25722,
                     letterGrade-> F ]:TranscriptEntry,
                   \#[syllabusEntry-> SE_25723,
                     letterGrade-> F]:TranscriptEntry,
                   \#[syllabusEntry-> SE_25724,
                     letterGrade-> D_MINUS]:TranscriptEntry,
                   \#[syllabusEntry-> SE_25725,
                     letterGrade-> D]:TranscriptEntry,
                   \#[syllabusEntry-> SE_25711,
                     letterGrade-> D]:TranscriptEntry  }
         ].
```

Figure 5.41: Transcript information for fourth semester to test dismissed status



```
flora2 ?- ayse_aa2 [ cgpa -> ?CGPA].
?CGPA = 0.5588
1 solution(s) in 0.0310 seconds
Yes
flora2 ?- ayse_aa2 [ status -> ?Status].
?Status = unsatisfactory
1 solution(s) in 0.0000 seconds
Yes
flora2 ?- ayse_aa3 [ cgpa -> ?CGPA].
?CGPA = 0.5588
1 solution(s) in 0.0150 seconds
Yes
flora2 ?- ayse_aa3 [ status -> ?Status].
?Status = unsatisfactory
```

Figure 5.42: CGP and status of the second and third semesters



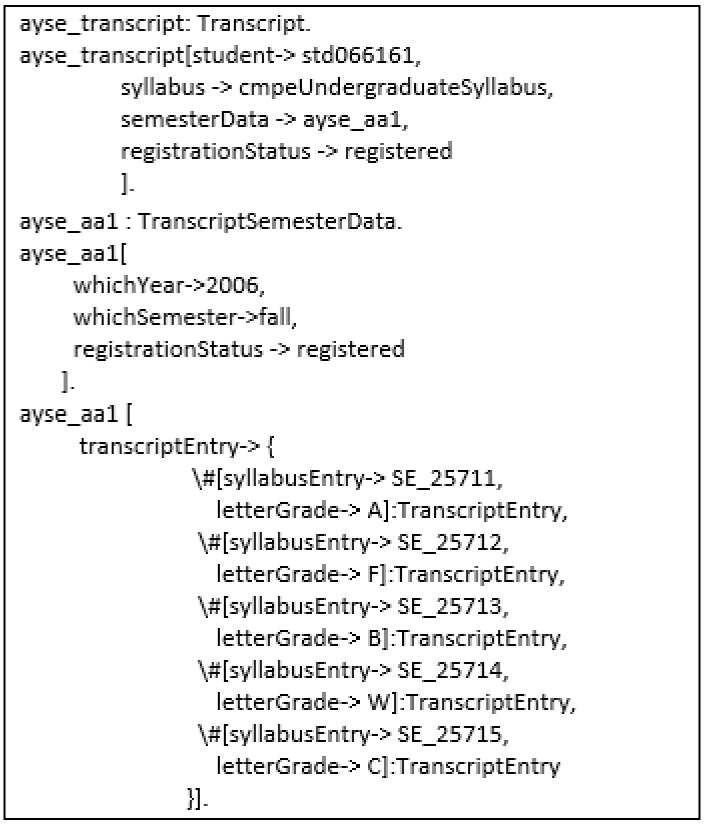
```
flora2 ?- ayse_aa4[ cgpa -> ?CGPA ].
?CGPA = 0.7588
1 solution(s) in 0.0310 seconds
Yes
flora2 ?- ayse_aa4[ status -> ?Status ].
?Status = dismissed
1 solution(s) in 0.0000 seconds
Yes
flora2 ?- %run_registration ( std066161, 2014, fall, ?SuggestedCourseList ).
Elapsed (CPU) time 5.2970 (5.1410) seconds
No
```

Figure 5.43: Status of the student is dismissed

5.4 Special Cases

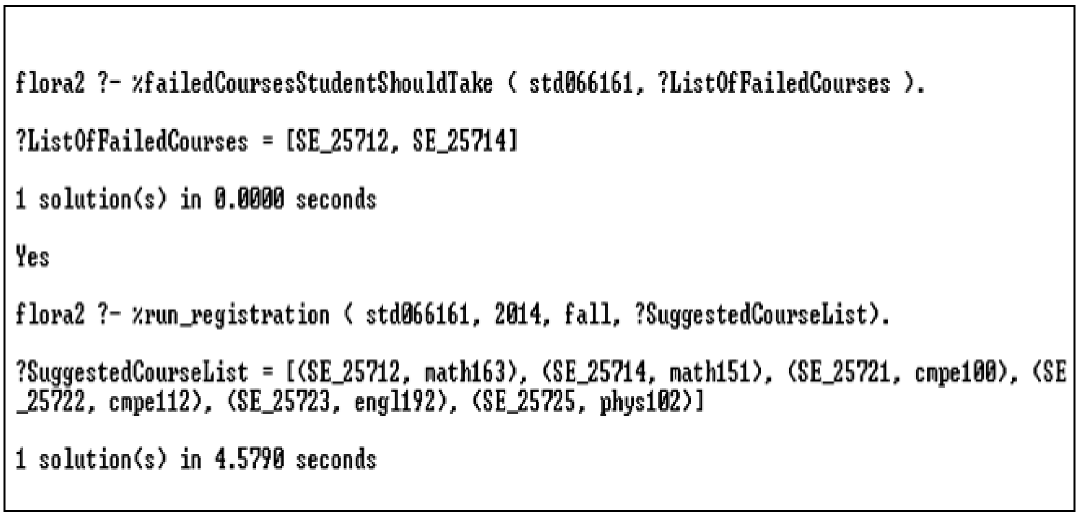
There are rules and regulations that must be obeyed while registering the student to his/her courses and some special cases. These are explained in chapter 2.

Firstly, if a student failed from course(s) or withdraw course(s), he/she must take this failed or withdrawn course(s) with a preference. Figure 5.44 contains the first semester transcript information. There are two courses with failing grade. Therefore, figure 5.45 shows the list of the suggested courses. There is a repetition of courses. Since one of them is failed and one of them is withdrawn in previous semester. For this purpose, first we run predicate of *%failedCoursesStudentShouldTake* with parameters of student number (*std066161*) and result (*ListOfFailedCourses*). This predicate gives the list of failed and withdrawn courses in the previous semester. The student failed from a course (*SE_25712)* and withdraw a course (*SE_25714*). Moreover, the course that has reference code of 25714 is prerequisite of course that has reference code of 25724. Because of this, agent did not suggest the course that has reference code of 25724. As a result, it suggests one failed course, one withdrawn course and three new courses which are suitable from next semester.



```
ayse_transcript: Transcript.
ayse_transcript[student-> std066161,
            syllabus -> cmpeUndergraduateSyllabus,
            semesterData -> ayse_aa1,
            registrationStatus -> registered
            ].
ayse_aa1 : TranscriptSemesterData.
ayse_aa1[
       whichYear->2006,
       whichSemester->fall,
       registrationStatus -> registered
      ].
ayse_aa1 [
       transcriptEntry-> {
                     \#[syllabusEntry-> SE_25711,
                        letterGrade-> A]:TranscriptEntry,
                      \#[syllabusEntry-> SE_25712,
                        letterGrade-> F]:TranscriptEntry,
                      \#[syllabusEntry-> SE_25713,
                        letterGrade-> B]:TranscriptEntry,
                      \#[syllabusEntry-> SE_25714,
                        letterGrade-> W]:TranscriptEntry,
                      \#[syllabusEntry-> SE_25715,
                        letterGrade-> C]:TranscriptEntry
                    }].
```

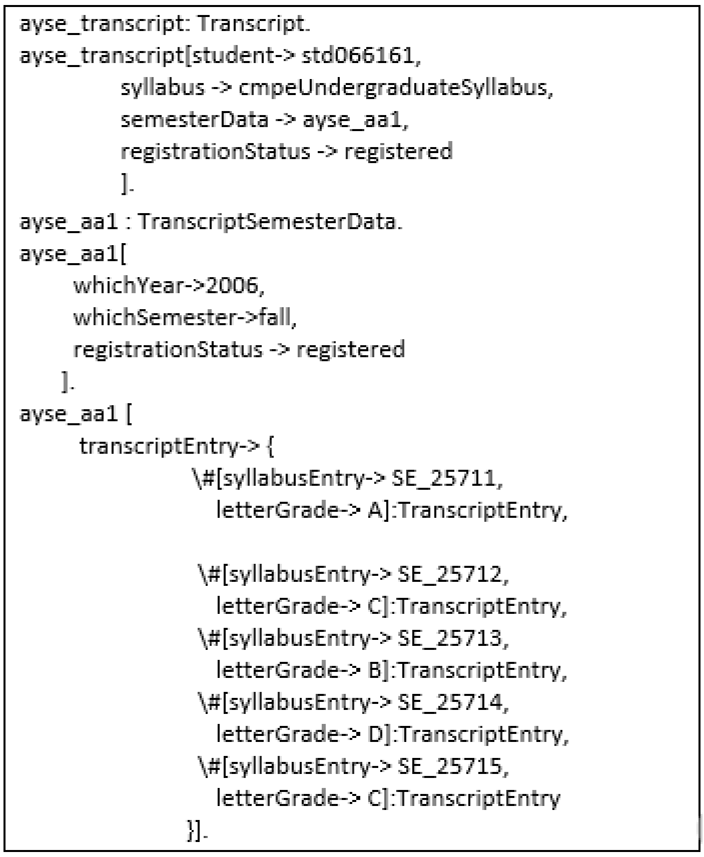
Figure 5.44: Transcript information to show failed courses for the first semester



```
flora2 ?- %failedCoursesStudentShouldTake ( std066161, ?ListOfFailedCourses ).

?ListOfFailedCourses = [SE_25712, SE_25714]

1 solution(s) in 0.0000 seconds

Yes

flora2 ?- %run_registration ( std066161, 2014, fall, ?SuggestedCourseList).

?SuggestedCourseList = [(SE_25712, math163), (SE_25714, math151), (SE_25721, cmpe100), (SE
_25722, cmpe112), (SE_25723, engl192), (SE_25725, phys102)]

1 solution(s) in 4.5790 seconds
```

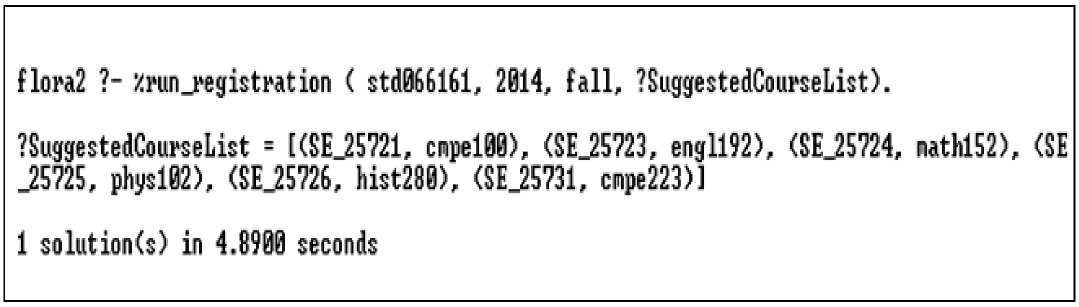
Figure 5.45: Repetition of the course(s) that failed or withdrawn

Secondly, if any course is not offered in the current semester, student cannot take that course. Consider a course cmpe112 which has reference code of 25722 is not offered in the current semester. We extract cmpe112 from course opening list for this sample. Figure 5.46 contains the transcript information for the first semester. We consider that the student passed all courses in the first semester. Figure 5.47 shows the suggested course list for this situation. For a regular student, there are six courses, one of is non credited in the second semester. However, the course cmpe112 is not offered in this semester. Therefore, our agent could not suggest the course cmpe112 to the advisor. Instead of cmpe112, it will give another suitable course from the next semester. In this example, it suggests the course cmpe223 whose reference code is 25731.



```
ayse_transcript: Transcript.
ayse_transcript[student-> std066161,
            syllabus -> cmpeUndergraduateSyllabus,
            semesterData -> ayse_aa1,
            registrationStatus -> registered
            ].
ayse_aa1 : TranscriptSemesterData.
ayse_aa1[
      whichYear->2006,
      whichSemester->fall,
      registrationStatus -> registered
     ].
ayse_aa1 [
       transcriptEntry-> {
                   \#[syllabusEntry-> SE_25711,
                      letterGrade-> A]:TranscriptEntry,

                    \#[syllabusEntry-> SE_25712,
                      letterGrade-> C]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25713,
                      letterGrade-> B]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25714,
                      letterGrade-> D]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25715,
                      letterGrade-> C]:TranscriptEntry
                  }].
```

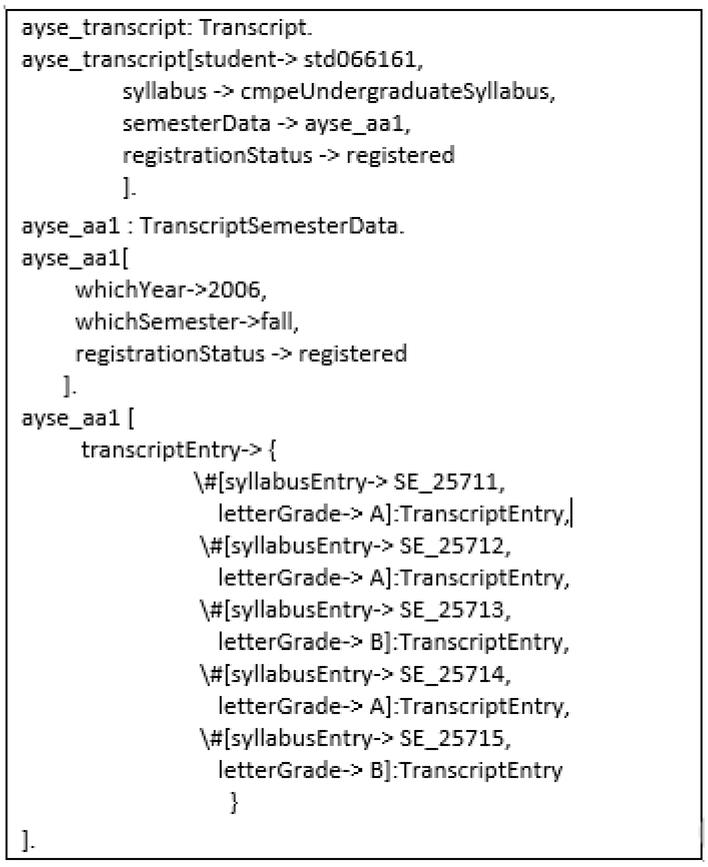
Figure 5.46: Transcript information for first semester (if course is not offered)



```
flora2 ?- %run_registration ( std066161, 2014, fall, ?SuggestedCourseList).

?SuggestedCourseList = [(SE_25721, cmpe100), (SE_25723, engl192), (SE_25724, math152), (SE
_25725, phys102), (SE_25726, hist280), (SE_25731, cmpe223)]

1 solution(s) in 4.8900 seconds
```

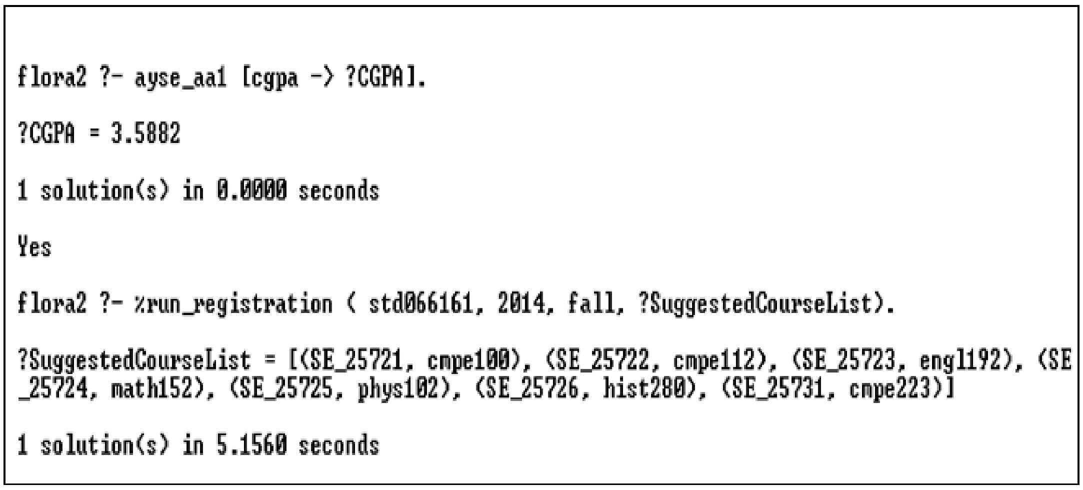
Figure 5.47: List of the suggested courses in case of a course not offered in that semester

Figure 5.48 contains the transcript information for the first semester. The student's GPA is greater than 3.00 in this semester. Figure 5.49 shows that our agent suggests courses of the second semester to the student. Furthermore, the student's GPA is greater than 3.00. Because of this student can take one extra suitable course from next semester. In this run, besides five credited courses and a non-credited course of the seond semester, our agent suggests the course cmpe223 whose reference code is 25731 as an extra course.



```
ayse_transcript: Transcript.
ayse_transcript[student-> std066161,
            syllabus -> cmpeUndergraduateSyllabus,
            semesterData -> ayse_aa1,
            registrationStatus -> registered
            ].
ayse_aa1 : TranscriptSemesterData.
ayse_aa1[
       whichYear->2006,
       whichSemester->fall,
       registrationStatus -> registered
     ].
ayse_aa1 [
       transcriptEntry-> {
                     \#[syllabusEntry-> SE_25711,
                       letterGrade-> A]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25712,
                       letterGrade-> A]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25713,
                       letterGrade-> B]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25714,
                       letterGrade-> A]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25715,
                       letterGrade-> B]:TranscriptEntry
                         }
].
```

Figure 5.48: Transcript information for first semester-Honor



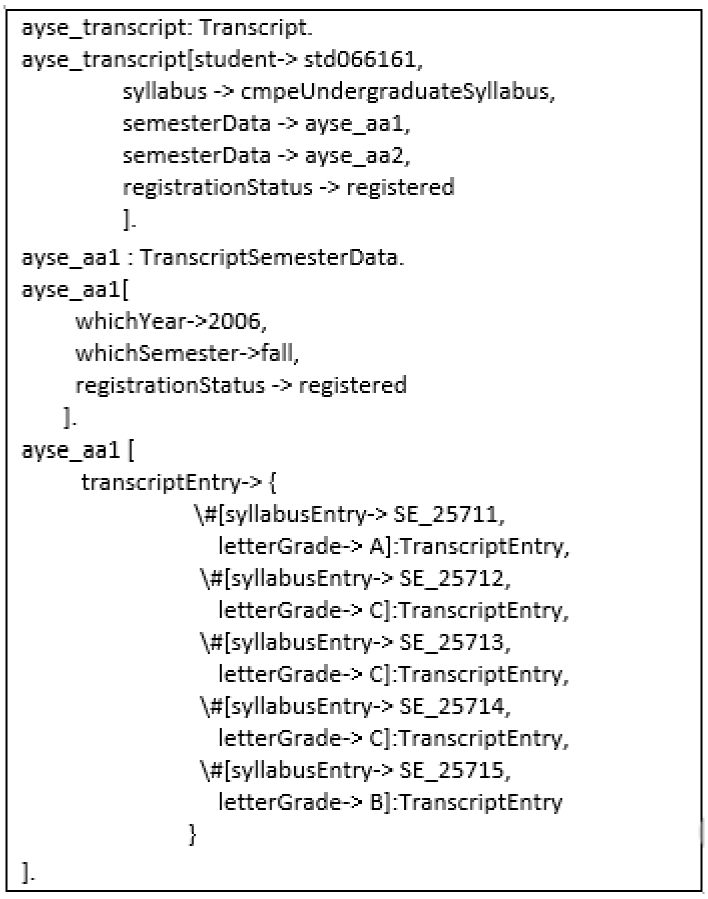
```
flora2 ?- ayse_aa1 [cgpa -> ?CGPA].

?CGPA = 3.5882

1 solution(s) in 0.0000 seconds

Yes

flora2 ?- %run_registration ( std066161, 2014, fall, ?SuggestedCourseList).

?SuggestedCourseList = [(SE_25721, cmpe100), (SE_25722, cmpe112), (SE_25723, engl192), (SE
_25724, math152), (SE_25725, phys102), (SE_25726, hist280), (SE_25731, cmpe223)]

1 solution(s) in 5.1560 seconds
```

Figure 5.49: List of the suggested courses with one extra course

5.6 Sample Run for Optimization

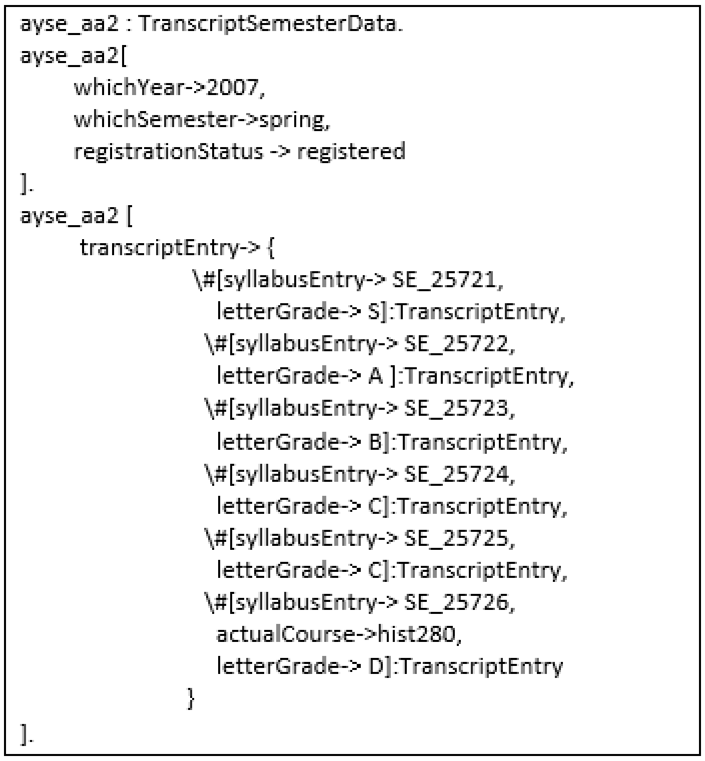
In this section we show sample runs of the *optimization* module which is another module of REG-EXPERT to introduce our agent to the user.

Figure 5.50 and 5.51 contains the transcript information for the first and second semester respectively. Figure 5.52 shows the CGPA and status for the transcript information below. Figure 5.53 shows that we first run our *course finder* module and our agent suggests the course list that student (*std066161*) should take in *fall* semester of *2014*. Secondly, we run our optimization module by giving query of *%how_many_clashes_all (std066161, 2014, fall, ?TimeTable).* As shown in figure 5.31 our agent gives a time table which is course based. Our agent finds the optimal solution which has 0 hour clash. There are two time table because cmpe211 has two groups. Therefore, cmpe211 has two different period in both time table.



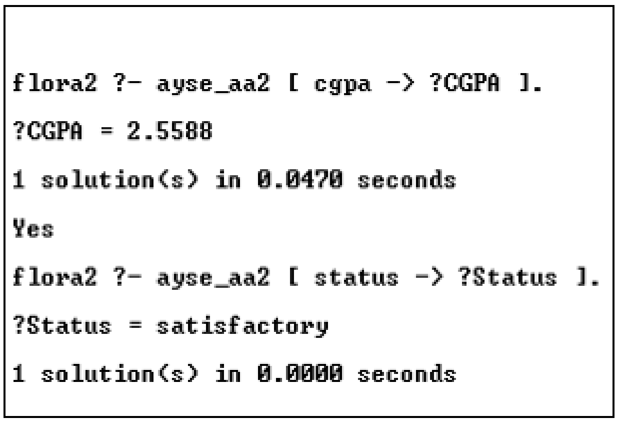
```
ayse_transcript: Transcript.
ayse_transcript[student-> std066161,
            syllabus -> cmpeUndergraduateSyllabus,
            semesterData -> ayse_aa1,
            semesterData -> ayse_aa2,
            registrationStatus -> registered
            ].
ayse_aa1 : TranscriptSemesterData.
ayse_aa1[
       whichYear->2006,
       whichSemester->fall,
       registrationStatus -> registered
      ].
ayse_aa1 [
        transcriptEntry-> {
                     \#[syllabusEntry-> SE_25711,
                        letterGrade-> A]:TranscriptEntry,
                      \#[syllabusEntry-> SE_25712,
                        letterGrade-> C]:TranscriptEntry,
                      \#[syllabusEntry-> SE_25713,
                        letterGrade-> C]:TranscriptEntry,
                      \#[syllabusEntry-> SE_25714,
                        letterGrade-> C]:TranscriptEntry,
                      \#[syllabusEntry-> SE_25715,
                        letterGrade-> B]:TranscriptEntry
                     }
].
```

Figure 5.50: Transcript information for first semester-optimization module



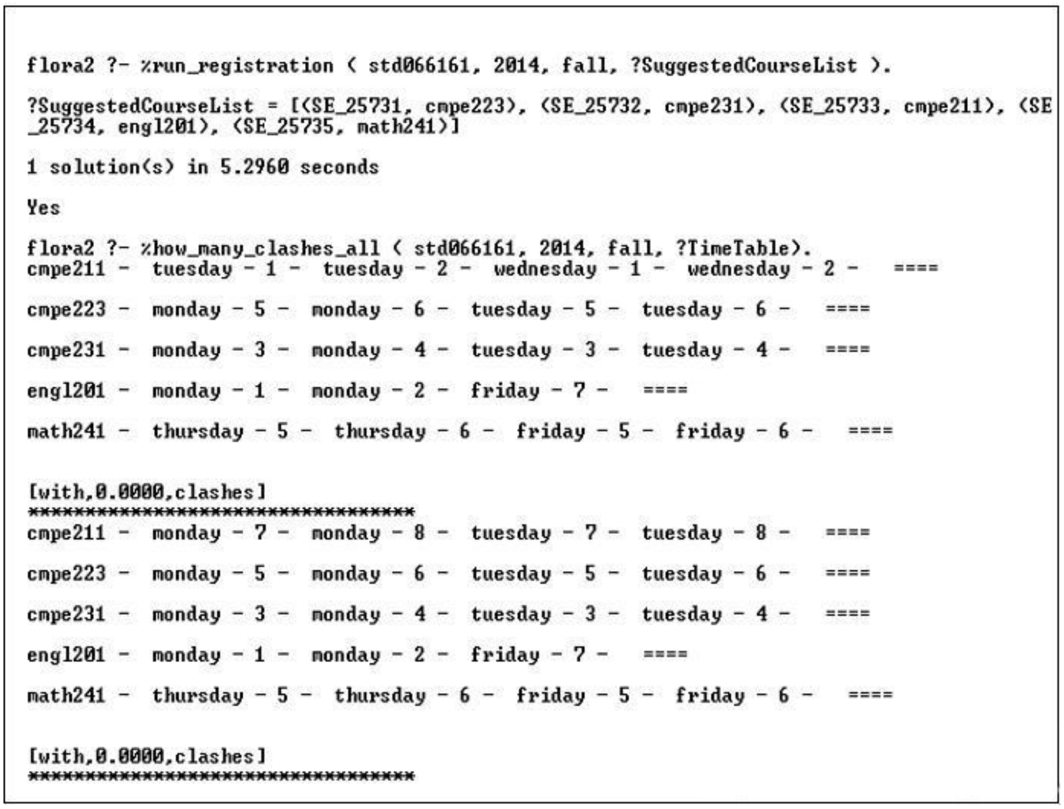
```
ayse_aa2 : TranscriptSemesterData.
ayse_aa2[
       whichYear->2007,
       whichSemester->spring,
       registrationStatus -> registered
].
ayse_aa2 [
        transcriptEntry-> {
                     \#[syllabusEntry-> SE_25721,
                        letterGrade-> S]:TranscriptEntry,
                       \#[syllabusEntry-> SE_25722,
                        letterGrade-> A ]:TranscriptEntry,
                       \#[syllabusEntry-> SE_25723,
                        letterGrade-> B]:TranscriptEntry,
                       \#[syllabusEntry-> SE_25724,
                        letterGrade-> C]:TranscriptEntry,
                       \#[syllabusEntry-> SE_25725,
                        letterGrade-> C]:TranscriptEntry,
                       \#[syllabusEntry-> SE_25726,
                        actualCourse->hist280,
                        letterGrade-> D]:TranscriptEntry
                     }
].
```

Figure 5.51: Transcript information for second semester-optimization module



```
flora2 ?- ayse_aa2 [ cgpa -> ?CGPA ].

?CGPA = 2.5588

1 solution(s) in 0.0470 seconds

Yes

flora2 ?- ayse_aa2 [ status -> ?Status ].

?Status = satisfactory

1 solution(s) in 0.0000 seconds
```

Figure 5.52: CGPA and status for the Table 8



```
flora2 ?- %run_registration ( std066161, 2014, fall, ?SuggestedCourseList ).

?SuggestedCourseList = [(SE_25731, cmpe223), (SE_25732, cmpe231), (SE_25733, cmpe211), (SE
_25734, engl201), (SE_25735, math241)]

1 solution(s) in 5.2960 seconds

Yes

flora2 ?- %how_many_clashes_all ( std066161, 2014, fall, ?TimeTable).
cmpe211 -  tuesday - 1 -  tuesday - 2 -  wednesday - 1 -  wednesday - 2 -   ====

cmpe223 -  monday - 5 -  monday - 6 -  tuesday - 5 -  tuesday - 6 -   ====

cmpe231 -  monday - 3 -  monday - 4 -  tuesday - 3 -  tuesday - 4 -   ====

engl201 -  monday - 1 -  monday - 2 -  friday - 7 -   ====

math241 -  thursday - 5 -  thursday - 6 -  friday - 5 -  friday - 6 -   ====


[with,0.0000,clashes]
************************************
cmpe211 -  monday - 7 -  monday - 8 -  tuesday - 7 -  tuesday - 8 -   ====

cmpe223 -  monday - 5 -  monday - 6 -  tuesday - 5 -  tuesday - 6 -   ====

cmpe231 -  monday - 3 -  monday - 4 -  tuesday - 3 -  tuesday - 4 -   ====

engl201 -  monday - 1 -  monday - 2 -  friday - 7 -   ====

math241 -  thursday - 5 -  thursday - 6 -  friday - 5 -  friday - 6 -   ====


[with,0.0000,clashes]
************************************
```

Figure 5.53: Optimized time table

Chapter 6

RELATED WORK

In this chapter we focus on related works with our work. Our work is intelligent agent system that has three modules: *transcript computation*, *course finder* and *optimization*. There are other systems which are focused on course registration. Each has its own strategy for implementing the course registration system and approach to course registration problem in the universities. Some of these systems are “ Semantic Web Services for University Course Registration” [16], “ A Framework for a WAP-Based Course Registration System” [17] and “ Using Bayesian Network for Planning Course Registration Model for Undergraduate Students” [18]. Firstly, we give information about “ Semantic Web Services for University Course Registration [16]”. In this system, they focused on semantic web services and give a semantic specification of a course registration system with no actual implementation [16].

Second related work is “ A Framework for a WAP-Based Course Registration System” [17]. They implement a system that is WAP-based course registration. They designed this system to provide some facilities to students, while they are registering to their courses. “ WAP-based system is an emerging technology aims to enable clients to facilitate their daily business in more efficient way and in less consuming time” [17]. Also, this system gives an opportunity to students to register their course by themselves [17].

Another related work to course registration is “ Using Bayesian Network for Planning Course Registration Model for Undergraduate Students” [18]. In this work they use Bayesian Network which is data mining technique. They try to help to students to plan their enrollments. Bayesian Network is used to predict undergraduate student's accomplishment. In this work they focused on helping the students according to their skills and so that they will be successful in their department [18].

Furthermore, there is one more related work which is “ Optimization Algorithms for Student Scheduling via Constraint Satisfiability” [19]. In this work they focused on student scheduling problem which is a kind of constraint satisfiability problem. They tried different type of optimization algorithms to solve this problem and they found that best algorithm is offering ordering algorithm.

Also, there is a related work that is “ Assuring Quality Service in Higher Education: Registration and Advising Attitudes in a Private University Lebanon” [11]. In this work they focused on advising the students for their academic education life. “ This study attempts to measure student attitudes of registration and academic advising across different faculties to assure positive quality service complementing that of the academic” [11].

Our work is different from all the above cited works because of its knowledge-based nature and specificity in its aim, i.e. helping advisors make correct choice in the registiration process in an intelligent manner.

Chapter 7

CONCLUSION AND FUTURE WORK

In this thesis we focused on the design and implementation of a REG-EXPERT, which is an intelligent software agent implemented in the Flora-2 language. The main function of this agent is to help advisors to register the courses of students according to the rules of university. Specifically, REG-EXPERT obeys the rules and regulations of the Eastern Mediterranean University and its curriculi in making its recommendation. Also, it considers the status of the student because status of the student affect how many courses will be loaded to student in the related semester. REG-EXPERT has three modules. One of them is transcript generation. This module contains information about transcript of the student. Also, it computes the GPA and CGPA of the student. Secondly, REG-EXPERT has a module that makes course registration suggestions, which is the heart of the agent. Finally, there is the module of optimization. This module finds a timetable for the student which has as few clashes as possible, given the currently opened courses in the semester.

In the future, we are planning to design graphical user interface that is user friendly and functional of our intelligent agent. Also, we have some ideas on implementing new module(s) to improve our agent's functionality. For example, currently there is no interface to relational data about students, courses etc. So, data should be converted manually into the knowledge base.

REFERENCES

[1] *CMPE - EMU Department* [Online]. Available: http://cmpe.emu.edu.tr/ [Accessed November 2014].

[2] Kifer, M., Yang, G., Wan, H., & Zhao, C. (2013). Flora-2: User's Manual.

[3] *EMU Regulations* [Online]. Available: http://mevzuat.emu.edu.tr/content.htm [Accessed November 2014].

[4] Padgham, L., & Winikoff, M. (2004). *Developing Intelligent Agent Systems*, John Wiley and Sons, (pp. 1-4).

[5] Wikimedia Foundation, Inc., Software agent, http://en.wikipedia.org/wiki/Software_agent (Accessed on December 2014).

[6] Croft, D., W. (1997). Intelligent Software Agents: Definitions and Applications, *Analytic Services, Inc. (ANSER).*

[7] Wikimedia Foundation, Inc., Intelligent agent, http://en.wikipedia.org/wiki/Intelligent_agent (Accessed on December 2014).

[8] Mouta, F., & Oliveira E. (1997). How an Agent Makes Decisions while Keeping Responsiveness. *IEEE International Conference on Intelligent Processing Systems*, 154-158.

[9] Akerkar, R., A., & Sajja, P., S. (2010). *Knowledge-Based Systems*, David Pallai, (pp. 1-19).

[10] Wikimedia Foundation, Inc., Knowledge-based systems, http://en.wikipedia.org/wiki/Knowledge-based_systems (Accessed on January 2015).

[11] Abouchedid, K., & Nasser, R. (2002). Assuirng Quality Service in Higher Education: Registration and Advising Attitudes in a Private University Lebanon, *Quality Assurance in Education*, Vol. 10, No. 4, 198-203.

[12] Kifer, M. (2005). Nonmonotonic Reasoning in FLORA-2. *Springer-Verlag Berlin Heidelberg*, 1-12.

[13] Wikimedia Foundation, Inc., Flora-2, http://en.wikipedia.org/wiki/Flora-2 (Accessed on December 2014).

[14] Wikimedia Foundation, Inc., HiLog, http://en.wikipedia.org/wiki/HiLog (Accessed on December 2014).

[15] Knowledge Representation & Reasoning with Flora-2, Transaction Logic http://flora.sourceforge.net/aboutTR.html (Accessed December 2014).

[16] Cobanoğlu, Ş., & Bayram, Z. (2014). Semantic Web Services for University Course Registration. *Lecture Notes in Computer Science*, 3-16.

[17] Al-Bastaki, Y., & Al-Ajeeli, A.(2004). A Framework for a WAP-Based Course Registration System. *Computers & Education*, Vol. 44, No. 3, 327-342.

[18] Pumpuang, P., Sirivihok, A., Praneetpolgrang, P., & Numprasertchai, S. (2008). Using Bayesian Network for Planning Course Registration Model for Undergraduate Students, *Digital Ecosystems and Technologies*, 492-496.

[19] Feldman, R., & Golumbic, M. C. (1990). Optimization Algorithms for Student Scheduling via Constraint Satisfiability, *The Computer Journal*, Vol. 33, No. 4 356-364.

APPENDICES

Appendix A: Source Code for Concepts of REG-EXPERT Agent

A1: Concepts for the Transcript Generator Module

```
Transcript[| student=>Student,
           syllabus => Syllabus,
           semesterData => TranscriptSemesterData,
           registrationStatus => RegistrationStatus|].
TranscriptEntry[| syllabusEntry=>SyllabusEntry,
               actualCourse=> Course,
               courseTaken=>Course,
               letterGrade=>Grade,
               numeric_contribution => \float,
               credit_counted => \integer,
               computed_credit_counted(\integer) => \integer,
               computed_numeric_contribution(\integer) => \integer |].
TranscriptSemesterData[| whichYear=> \integer,
                     whichSemester=>Semester,
                     transcriptEntry => TranscriptEntry,
                     gpa=> \float,
                     cgpa=> \float,
                     status => Status,
                    registrationStatus => RegistrationStatus,
                    actualAcademicTerm => \integer,
                    realAcademicTerm => \integer,
                    numeric_contribution => \float,
                    computed_credit_counted(\integer)=> \integer,
                    computed_numeric_contribution(\integer) => \float,
                    cumulative_credits => \float,
                    cumulative_contribution => \float,
                    how_many_courses_remaining => \integer,
                    how_many_credited_courses_remaining => \integer|].
?X[ mostRecentRealAcademicSemester ->?MRRAS]:-
   ?X: Student,
   ?Transcript[student->?X],
   ?MRRAS = max {?RAT |
                    ?Transcript[semesterData-> ?TSD],
                    ?TSD[realAcademicTerm->?RAT],
                    ?TSD[registrationStatus->registered]},
   !.
?X[ mostRecentRealAcademicSemester ->?MRRAS]:-
   ?X: Student,
   ?MRRAS=0.
?X[ currentAcademicSemester->?CAS]:-
   ?X: Student,
   ?Transcript[student->?X],
   ?LastActualSemester = max {?AAT |
                              ?Transcript[semesterData-> ?TSD],
                              ?TSD[actualAcademicTerm->?AAT],
                              ?TSD[registrationStatus->registered],
                              ?TSD[whichSemester->?Semester],
                              (?Semester = spring ; ?Semester = fall)},
   ?CAS \is ?LastActualSemester + 1,
   !.
?X[ currentAcademicSemester->?CAS]:-
   ?X: Student,
   ?CAS=1.
?X[ turkishOrForeign->turkish]:-
   ?X: Student,
   ( ?X[nationality->turkish] ; ?X[nationality->turkish_cypriot]),
   !.
?X[ turkishOrForeign->foreign]:-
   ?X: Student.
```

A2: Concepts for the Course Finder Module

```
RegistrationRequest[|student => Student,
                     year => \integer,
                     semester => Semester|].
SingleRegistrationResult[|student=> Student,
                          year=> \integer,
                          semester => Semester,
                          single_reg=> CourseOpening|].
RegistrationResult [|student=> Student,
                     year=> \integer,
                     semester => Semester,
                     syllabusEntry => SyllabusEntry,
                     courseOpening=> CourseOpening|].
Student[|yearEnrolled=>\integer,
         semesterEnrolled=>Semester,
         inProgram=>AcademicProgram,
         currentAcademicSemester => \integer,
         mostRecentRealAcademicSemester => \integer,
         turkishOrForeign => TurkishOrForeign|].
TurkishOrForeign[].
Day[].
Language[].
Course[|courseCode => string,
       courseName => string,
       hasPrerequisite  => Course,
       lecture_hours => \integer,
       lab_hours => \integer,
       credits => \integer,
       instructionLanguage => Language,
       ects => \integer |].
CourseType[].
Syllabus[| forProgram=>AcademicProgram,
            syllabusEntry => SyllabusEntry|].
SyllabusEntry[| whichYear=>YearType,
                whichSemester=>Semester,
                referenceCode=> \string,
                course=>Course,
                courseType => CourseType|].
Person[| id => \string,
         gender => Gender,
         date_of_birth => \date,
         name => \string,
         lastName => \string,
         address => Address,
         nationality => Nationality|].
Nationality[].
Gender[].
Grade[||].
RegistrationStatus[].
AcademicProgram [| programName=> \string,
                   programID=> \string,
                   syllabus=> Syllabus,
                   belongsTo=> Department|].
CourseOpening [| groupNo => \integer,
                 ofCourse => Course,
                 year => \integer,
                 semester => Semester,
                 teachingTimes => RoomDayPeriodDuration|].
```

```
Curriculum [| academicProgram=> AcademicProgram,
            refCode=> \string,
            courseName=> Course|].
TakesCourse [| courseOpening=> CourseOpening,
             student=> Student,
             grade=> \string|].
RoomDayPeriodDuration [| room => Classroom,
                      day => Day,
                      period => \integer,
                      duration => \integer|].
Building[].
Semester[].
Student::Person.
UndergraduateProgram::AcademicProgram.
GraduateProgram:: AcademicProgram.
TurkishProgram:: AcademicProgram.
EnglishProgram:: AcademicProgram.
EnglishUndergraduateProgram:: EnglishProgram.
EnglishUndergraduateProgram::UndergraduateProgram.
LectureRoomDayPeriodDuration::RoomDayPeriodDuration.
LabRoomDayPeriodDuration::RoomDayPeriodDuration.
AreaElective :: Course.
CmpeAreaElective :: AreaElective.
CmseAreaElective :: AreaElective.
BlgmAreaElective :: AreaElective.
UniversityElective :: Course.
CmpeUniversityElective :: UniversityElective.
CmseUniversityElective :: UniversityElective.
BlgmUniversityElective :: UniversityElective.
```

```
ScienceCourse :: Course.
CmpeScienceCourse :: ScienceCourse.
CmseScienceCourse :: ScienceCourse.
BlgmScienceCourse :: ScienceCourse.
FinanceCourse :: Course.
CmpeFinanceCourse :: FinanceCourse.
CmseFinanceCourse :: FinanceCourse.
BlgmFinanceCourse :: FinanceCourse.
EthicsCourse :: Course.
CmpeEthicsCourse :: EthicsCourse.
CmseEthicsCourse :: EthicsCourse.
BlgmEthicsCourse :: EthicsCourse.
TurkishCourse: CourseType.
HistoryCourse: CourseType.
TurkishOrHistoryCourse: CourseType.
CmpeSummerTraining: CourseType.
NormalCourse: CourseType.
CmpeEthicsCourse: CourseType.
CmpeFinanceCourse: CourseType.
CmpeScienceCourse: CourseType.
CmpeUniversityElective: CourseType.
CmpeAreaElective: CourseType.
CmpeGraduationProject1 : CourseType.
CmpeGraduationProject2 : CourseType.
TurkishCourse:: TurkishOrHistoryCourse.
HistoryCourse:: TurkishOrHistoryCourse.
NormalSyllabusEntry::SyllabusEntry.
RestricredElectiveScienceSyllabusEntry:: SyllabusEntry.
RestricredElectiveFinanceSyllabusEntry:: SyllabusEntry.
```

AreaElectiveSyllabusEntry:: SyllabusEntry.

UniversityElectiveSyllabusEntry:: SyllabusEntry.

EthicsElectiveSyllabusEntry:: SyllabusEntry.

TurkishOrHistorySyllabusEntry:: SyllabusEntry.

TurkishUndergraduateProgram:: TurkishProgram.

TurkishUndergraduateProgram:: UndergraduateProgram.

RealGrade :: Grade.

PassingGrade :: Grade.

FailingGrade :: Grade.

A3: Concepts for the Optimization Module

```
CourseOpening [| groupNo => \integer,
            ofCourse => Course,
            year => \integer,
            semester => Semester,
            teachingTimes => RoomDayPeriodDuration|].
RoomDayPeriodDuration [| room => Classroom,
                  day => Day,
                  period => \integer,
                  duration => \integer|].
```

Appendix B: Source Code for Modules of REG-EXPERT Agent

B1: Source Code for the Transcript Component

```
?X[how_many_credited_courses_remaining->?HM]:-
    ?X:TranscriptSemesterData,
    ?X[ realAcademicTerm -> ?RAT],
    ?Transcript[ semesterData -> ?X],
    ?Transcript[syllabus-> ?Syllabus],
    ?ListOfCreditedCoursesAlreadyPassed =
      setof { ?SE |  ?Transcript[ semesterData ->?TSD],
                     ?TSD[ transcriptEntry -> ?TE],
                     ?TE[ syllabusEntry -> ?SE,
                          courseTaken -> ?Course
                        ],
                     ?Course[credits -> ?Credits],
                     ?Credits>0,
                     %findMostRecentGradeSERealSem(?Transcript, ?RAT,  ?SE, ?LG),
                     ?LG: PassingGrade},
   ?ListOfCreditedCoursesInTheSyllabus =
     setof { ?SE1 | ?Syllabus[ syllabusEntry -> ?SE1],
                    ?SE1[course->?Course1],
                    ?Course1[credits -> ?Credits1],
                    ?Credits1>0 },
   %setSubtract( ?ListOfCreditedCoursesInTheSyllabus,
                 ?ListOfCreditedCoursesAlreadyPassed,
                 ?ListOfCreditedCoursesRemaining,
   ?ListOfCreditedCoursesRemaining[length->?HM]@\btp.
?X[how_many_courses_remaining->?HM]:-
    ?X:TranscriptSemesterData,
    ?X[ realAcademicTerm -> ?RAT],
    ?Transcript[ semesterData -> ?X],
    ?Transcript[syllabus-> ?Syllabus],
    ?ListOfCoursesAlreadyPassed =
      setof { ?SE |  ?Transcript[ semesterData ->?TSD],
                     ?TSD[ transcriptEntry -> ?TE],
                     ?TE[ syllabusEntry -> ?SE ],
                     %findMostRecentGradeSERealSem(?Transcript, ?RAT,  ?SE, ?LG),
                     ?LG: PassingGrade},
    ?ListOfCoursesInTheSyllabus =
      setof { ?SE1 | ?Syllabus[ syllabusEntry -> ?SE1]},
    %setSubtract( ?ListOfCoursesInTheSyllabus,
    ?ListOfCoursesAlreadyPassed,
    ?ListOfCoursesRemaining),
    ?ListOfCoursesRemaining[length->?HM]@\btp.
?X[status->satisfactory]:-
    ?X:TranscriptSemesterData,
    ?X[actualAcademicTerm->?Z],
    (?Z=0;?Z=1),
    !.
?X[status->satisfactory]:-
    ?X:TranscriptSemesterData,
    ?X[actualAcademicTerm->?Z],
    (?Z=2;?Z=3;?Z=4),
    ?X[cgpa->?Cgpa],
    ?Cgpa>=1.50 ,!.
?X[status->satisfactory]:-
    ?X:TranscriptSemesterData,
    ?X[actualAcademicTerm->?Z],
    (?Z=5;?Z=6;?Z=7),
    ?X[cgpa->?Cgpa],
    ?Cgpa>=1.80 ,!.
?X[status->satisfactory]:-
    ?X:TranscriptSemesterData,
    ?X[actualAcademicTerm->?AAT],
    ?AAT>=8,
    ?X[cgpa->?Cgpa],
    ?Cgpa>=2.00 ,!.
?X[status->onProbation]:-
    ?X:TranscriptSemesterData,
    ?X[actualAcademicTerm->?Z],
    (?Z=2;?Z=3;?Z=4),
    ?X[cgpa->?Cgpa],
    ?Cgpa<1.50 ,
    ?Cgpa>=1.00,!.
?X[status->onProbation]:-
    ?X:TranscriptSemesterData,
    ?X[actualAcademicTerm->?Z],
    (?Z=5;?Z=6;?Z=7),
    ?X[cgpa->?Cgpa],
    ?Cgpa<1.80 ,
    ?Cgpa>=1.50,!.
?X[status->onProbation]:-
    ?X:TranscriptSemesterData,
    ?X[actualAcademicTerm->?AAT],
    ?AAT>=8,
    ?X[cgpa->?Cgpa],
    ?Cgpa<2.00 ,
    ?Cgpa>=1.80,!.
```

```
?X[status->unsatisfactory]:-
   ?X:TranscriptSemesterData,
   ?X[actualAcademicTerm->?Z],
   (?Z=2;?Z=3),
   ?X[cgpa->?Cgpa],
   ?Cgpa<1.00, !.
?X[status->dismissed]:-
   ?X:TranscriptSemesterData,
   ?X[actualAcademicTerm->?Z],
   ?Z>=4,
   ?X[cgpa->?Cgpa],
   ?Cgpa<1.00, !.
?X[status->unsatisfactory]:-
   ?X:TranscriptSemesterData,
   ?X[actualAcademicTerm->?Z] ,
   (?Z=5;?Z=6;?Z=7),
   ?X[cgpa->?Cgpa],
   ?Cgpa<1.50, !.
?X[status->unsatisfactory]:-
   ?X:TranscriptSemesterData,
   ?X[actualAcademicTerm->?AAT],
   ?AAT>=8,
   ?X[cgpa->?Cgpa],
   ?Cgpa<1.80, !.
\udf num(A) := 4.
\udf num(A_MINUS) := 3.7.
\udf num(B_PLUS) := 3.3.
\udf num(B) := 3.
\udf num(B_MINUS) := 2.7.
```

```
\udf num(C_PLUS) := 2.3.
\udf num(C) := 2.
\udf num(C_MINUS) := 1.7.
\udf num(D_PLUS) := 1.3.
\udf num(D) := 1.
\udf num(D_MINUS) := 0.7.
\udf num(F) := 0.
\udf num(NG) := 0.
?X[courseTaken->?Course]:-
   ?X:TranscriptEntry,
   ?X[actualCourse->?Course],!.
?X[courseTaken->?Course]:-
   ?X:TranscriptEntry,
   ?X[syllabusEntry->?SE],
   ?SE[course->?Course].
?X[computed_credit_counted(?LastRealSemester)->0]:-
   ?X:TranscriptEntry,
   ?X[syllabusEntry->?SE],
   ?TSD[transcriptEntry->?X],
   ?TRANSCRIPT[semesterData->?TSD],
   ?TSD[realAcademicTerm -> ?RATprev],
   ?TRANSCRIPT[semesterData->?OtherTSD],
   ?OtherTSD[realAcademicTerm -> ?OtherRAT],
   ?OtherRAT > ?RATprev,
   ?OtherRAT =< ?LastRealSemester,
   ?OtherTSD[transcriptEntry -> ?OtherTranscriptEntry],
   ?OtherTranscriptEntry[syllabusEntry -> ?SE],
   !.
?X[computed_credit_counted(?_LastSemester)->?CCC]:-
```

```
   ?X:TranscriptEntry,
   ?X[credit_counted->?CCC].
?X[credit_counted->?CC]:-
   ?X:TranscriptEntry,
   ?X[letterGrade->?LG],
   ?LG:RealGrade,
   ?X[syllabusEntry->?SE],
   ?SE[course->?CRSE],
   ?CRSE[credits->?CC].
?X[credit_counted->0]:-
   ?X:TranscriptEntry,
   ?X[letterGrade->?LG],
   \+ ?LG:RealGrade.
?X[numeric_contribution->?NGrade]:-
    ?X:TranscriptEntry,
    ?X[letterGrade->?LG],
    ?LG:RealGrade,
    ?NumericGrade = num(?LG),
    ?X[syllabusEntry->?SE],
    ?SE[course->?CRSE],
    ?CRSE[credits->?CC],
    ?NGrade \is ?CC*?NumericGrade.
?X[numeric_contribution->0]:-
   ?X:TranscriptEntry,
   ?X[letterGrade->?LG],
   \+ ?LG:RealGrade.
?X[computed_numeric_contribution(?LastSemester)->?CNC]:-
   ?X:TranscriptEntry,
   ?X[letterGrade->?LG],
   ?LG:RealGrade,
   ?NumericGrade = num(?LG),
   ?X[computed_credit_counted(?LastSemester)->?CCC],
   ?CNC \is ?CCC*?NumericGrade.
?X[computed_numeric_contribution(?_LastSemester)->0]:-
    ?X:TranscriptEntry,
    ?X[letterGrade->?LG],
    \+ ?LG:RealGrade.
?X[numeric_contribution->?NCSemester]:-
   ?X:TranscriptSemesterData,
   ?NCSemester = sum{?Y| ?X[transcriptEntry->?TE],
                      ?TE[numeric_contribution->?Y]}.
?X[computed_numeric_contribution(?LastSemester)->?NCSemester]:-
    ?X:TranscriptSemesterData,
    ?NCSemester = sum{?Y| ?X[transcriptEntry->?TE],
                       ?TE[computed_numeric_contribution(?LastSemester)->?Y]}.
?X[credit_counted->?CCSemester]:-
   ?X:TranscriptSemesterData,
   ?CCSemester = sum{?Y| ?X[transcriptEntry->?TE],
                      ?TE[credit_counted->?Y]}.
?X[computed_credit_counted(?LastSemester)->?CCSemester]:-
    ?X:TranscriptSemesterData,
    ?CCSemester = sum{?Y| ?X[transcriptEntry->?TE],
                       ?TE[computed_credit_counted(?LastSemester)->?Y]}.
?X[gpa->?GPA]:-
   ?X:TranscriptSemesterData,
   ?X[numeric_contribution->?NC],
   ?X[credit_counted->?CC],
   ?GPA \is ?NC/?CC.
```

```
?X[cumulative_credits->?CumC]:-
   ?X:TranscriptSemesterData,
   ?X[actualAcademicTerm ->?AAT],
   ?AAT=1,
   !,
   ?X[credit_counted->?CC],
   ?CumC = ?CC.
?X[cumulative_credits->?CumC]:-
   ?X:TranscriptSemesterData,
   ?X[realAcademicTerm ->?RAT],
   ?Transcript[semesterData->?X],
   ?RAT>1,
   ?PrevSemesterCredits =
     sum{?CC| ?Transcript[semesterData->?Y],
            \+ ?X=?Y,
            ?Y[realAcademicTerm->?RAT2],
            ?RAT2 < ?RAT,
            ?Y[computed_credit_counted(?RAT)->?CC]},
   ?X[credit_counted->?CC2],
   ?CumC \is ?CC2+?PrevSemesterCredits.
?X[cumulative_contribution->?CumC]:-
   ?X:TranscriptSemesterData,
   ?X[actualAcademicTerm ->?AAT],
   ?AAT=1,
   !,
   ?X[numeric_contribution->?CC],
   ?CumC = ?CC.
?X[cumulative_contribution->?CumC]:-
   ?X:TranscriptSemesterData,
   ?X[realAcademicTerm ->?RAT],
   ?Transcript[semesterData->?X],
   ?RAT>1,
   ?PrevSemesterContribution =
       sum{?CC| ?Transcript[semesterData->?Y],
               \+ ?X=?Y,
               ?Y[realAcademicTerm->?RAT2],
               ?RAT2 < ?RAT,
               ?Y[computed_numeric_contribution(?RAT)->?CC]},
               ?X[numeric_contribution->?CC2],
               ?CumC \is ?CC2+?PrevSemesterContribution.
?X[cgpa->?CGPA]:-
   ?X:TranscriptSemesterData,
   ?X[cumulative_contribution->?CumC],
   ?X[cumulative_credits->?CC],
   ?CGPA \is ?CumC/?CC.


?X[actualAcademicTerm->?AAT]:-
   ?X:TranscriptSemesterData,
   ?X[whichYear-> ?Year, whichSemester -> ?Semester],
   (?Semester=fall; ?Semester=spring),
   ?TRANSCRIPT[semesterData->?X],
   ?AAT0 = count{ ?TSD |
                   ?TRANSCRIPT[ semesterData -> ?TSD],
                   ?TSD[registrationStatus -> registered],
                   ?TSD[whichYear-> ?Year1],
                   ?TSD[whichSemester-> ?Semester1],
                   (?Semester1 = fall ; ?Semester1 = spring),
                   %semester_is_before(?Year1, ?Semester1, ?Year, ?Semester)
               },
   ?AAT \is ?AAT0 + 1.
 ?X[actualAcademicTerm->?AAT]:-
    ?X:TranscriptSemesterData,
    ?X[whichYear-> ?Year, whichSemester -> ?Semester],
    ?Semester=summer,
    ?TRANSCRIPT[semesterData->?X],
    ?AAT = count{ ?TSD |
                       ?TRANSCRIPT[ semesterData -> ?TSD],
                       ?TSD[registrationStatus -> registered],
                       ?TSD[whichYear-> ?Year1],
                       ?TSD[whichSemester-> ?Semester1],
                       (?Semester1 = fall ; ?Semester1 = spring),
                       %semester_is_before(?Year1, ?Semester1, ?Year, ?Semester)
                }.
?X[realAcademicTerm->?AAT]:-
   ?X:TranscriptSemesterData,
   ?X[whichYear-> ?Year, whichSemester -> ?Semester],
   ?TRANSCRIPT[semesterData->?X],
   ?AAT0 = count{ ?TSD |
                       ?TRANSCRIPT[ semesterData -> ?TSD],
                       ?TSD[registrationStatus -> registered],
                       ?TSD[whichYear-> ?Year1],
                       ?TSD[whichSemester-> ?Semester1],
                       %semester_is_before(?Year1, ?Semester1, ?Year, ?Semester)
                },
   ?AAT \is ?AAT0 + 1.
```

B2: Source Code for the Registration Component

```
%r2:- [ +'D://Flora//transcript'],
    [ +'D://Flora//student_instances'],
    [ +'D://Flora//regular_transcript_instances'],
    [ +'D://Flora//syllabus_instances'],
    [ +'D://Flora//classroom_instances'],
    [ +'D://Flora//Concepts'],
    [ +'D://Flora//address_instances'],
    [ +'D://Flora//course_instances'],
    [ +'D://Flora//room_day_period_instances'],
    [ +'D://Flora//instances'],
    [ +'D://Flora//CourseCodes'],
    [ +'D://Flora//utilities'],
    [ +'D://Flora//regular_transcript_instances'],
    [ +'D://Flora//constraints'],
    [ 'D://Flora//empty'>>m1],
    [ +'D://Flora//course_opening_instances'>>m1],
    [ 'D://Flora//empty'>>m2].
%run(?RegistrationResult):-
  ?_RegReq[student->?Student,
       year->?Year,
       semester->?Semester]:RegistrationRequest,
  %run_registration(?Student, ?Year, ?Semester, ?RegistrationResult).
%run_registration(?Student, ?Year, ?Semester, ?RegistrationResult):-
  %t_del(m2, ?_),
  t_deleteall{course_assigned(?,?,?,?,?)@m2},
  %find_student_status(?Student, ?StudentStatus),
  %make_registration(?Student, ?StudentStatus, ?Year, ?Semester),
  !,
  ?RegistrationResult =
  setof{?SE_Course | ?RR[?_->?_]:RegistrationResult@m2,
                 ?RR[courseOpening->?CO]@m2,
                 ?CO[ofCourse->?Course]@m1,
                 ?RR[syllabusEntry->?SE]@m2,
                 ?SE_Course = (?SE,?Course)}.
%find_student_status(?Student, ?StudentStatus):-
   ?Student[currentAcademicSemester-> ?CAS],
    \if (?CAS=1)
    \then
       (?StudentStatus=satisfactory)
    \else (
         ?Student[mostRecentRealAcademicSemester -> ?MRRAS],
         ?_Transcript[  student -> ?Student,
                     semesterData -> ?_TSD[ realAcademicTerm -> ?MRRAS,
                                        status -> ?StudentStatus
                                       ]  ]  ).
%make_registration(?Student, unsatisfactory,?Year,?Semester):-
  %make_registration_repeat_courses(5, ?Student,?Year,?Semester ).
%make_registration(?Student, satisfactory,?Year,?Semester):-
  ?Student[currentAcademicSemester->?CAS],
  ?CAS=1,
  !,
  ?HowManyNewCourses = 5,
  %make_registration_new_courses(?HowManyNewCourses,
?Student,?Year,?Semester ).
%make_registration(?Student, satisfactory,?Year,?Semester):-
  %failedCoursesStudentShouldTake(?Student, ?ListOfFailedCourses),
  ?ListOfFailedCourses[length->?HowManyFailedCourses]@\btp,
  %maxNumberOfCoursesStudentCanTake(?Student,?MaxCourses),
  ?HowManyNewCourses \is ?MaxCourses - ?HowManyFailedCourses,

    %make_registration_repeat_courses_list(?ListOfFailedCourses,
?Student,?Year,?Semester ),

    %make_registration_new_courses(?HowManyNewCourses,
?Student,?Year,?Semester ).

%make_registration(?Student, onProbation,?Year,?Semester):-

  %failedCoursesStudentShouldTake(?Student, ?ListOfFailedCourses),

  ?ListOfFailedCourses[length->?HowManyFailedCourses0]@\btp,

  %how_many_zero_one_credit(?ListOfFailedCourses,
?HowManyZeroOneCreditCourses),

  ?HowManyFailedCourses \is ?HowManyFailedCourses0 -
?HowManyZeroOneCreditCourses,

  %maxNumberOfCoursesStudentCanTake(?Student,?MaxCourses),

  ?HowManyNewCourses \is ?MaxCourses - ?HowManyFailedCourses,

  \if (?HowManyNewCourses>2)

  \then (?HowManyNewCoursesX = 2, ?HowManyOtherRepeatedCourses \is
?HowManyNewCourses-2)

  \else (?HowManyNewCoursesX = ?HowManyNewCourses,
?HowManyOtherRepeatedCourses = 0),

  %make_registration_repeat_courses_list(?ListOfFailedCourses,
?Student,?Year,?Semester ),

  %make_registration_repeat_courses(?HowManyOtherRepeatedCourses,
?Student,?Year,?Semester ),

  %make_registration_new_courses(?HowManyNewCoursesX,
?Student,?Year,?Semester ).

%how_many_zero_one_credit(?ListOfFailedCourses,
?HowManyZeroOneCreditCourses):-

  ?ListOfFailedCourses=[],

  !,

  ?HowManyZeroOneCreditCourses = 0.

%how_many_zero_one_credit( ?ListOfFailedCourses,
?HowManyZeroOneCreditCourses):-

  ?ListOfFailedCourses= [?SE|?RestSE],

  ?SE[course->?Course],
  ?Course[credits->?CR],
  (?CR=0 ; ?CR=1),
  !,
  %how_many_zero_one_credit(?RestSE, ?Temp),
  ?HowManyZeroOneCreditCourses \is ?Temp+1.
%how_many_zero_one_credit( ?ListOfFailedCourses,
?HowManyZeroOneCreditCourses):-
  ?ListOfFailedCourses = [?SE|?RestSE],
  ?SE[course->?Course],
  ?Course[credits->?CR],
  ?CR>1,
  %how_many_zero_one_credit(?RestSE,  ?HowManyZeroOneCreditCourses).
%maxNumberOfCoursesStudentCanTake(?Student,?M):-
  ?Student[currentAcademicSemester -> ?CAS],
  ?CAS = 1,
  !,
  ?Student[inProgram -> ?Program:AcademicProgram],
  ?Program[syllabus -> ?Syllabus: Syllabus],
  ?FirstYearFallSemesterSEs = setof { ?SE |
                              ?Syllabus[syllabusEntry->?SE:SyllabusEntry],
                              ?SE[whichYear -> freshman],
                              ?SE[whichSemester -> fall] },
 ?FirstYearFallSemesterSEs[length->?M]@\btp.
%maxNumberOfCoursesStudentCanTake(?Student,?M):-
  %find_student_status(?Student, ?StudentStatus),
  ?Student[currentAcademicSemester -> ?CAS],
  ?PrevAcademicSemester \is ?CAS-1,
  ?Transcript[ student -> ?Student],
  ?Transcript[ semesterData ->?_TSD1[ actualAcademicTerm ->
?PrevAcademicSemester,
                              how_many_credited_courses_remaining ->
?HowManyCreditedCoursesRemaining,

                              gpa -> ?GPA,

                              cgpa -> ?CGPA

                             ]: TranscriptSemesterData

          ],

  \if (?StudentStatus=satisfactory, ?HowManyCreditedCoursesRemaining =<7 )

  \then (?M=?HowManyCreditedCoursesRemaining)

  \else (

       \if (?StudentStatus=satisfactory,(?CGPA>=3 ; ?GPA>=3))

       \then (?M=6)

       \else (

            ?M=5

           ) ).

%make_registration_repeat_courses_list(?List, ?_Student,?_Year,?_Semester ):-
?List=[],!.

%make_registration_repeat_courses_list([?SE_for_failed_course|
?RemainingFailedCourses], ?Student,?Year,?Semester ):-

  %make_single_registration_specific_course(?SE_for_failed_course,?Student,
?Year, ?Semester),

  %make_registration_repeat_courses_list(?RemainingFailedCourses,
?Student,?Year,?Semester).

%make_registration_repeat_courses(?N, ?_Student,?_Year,?_Semester ):-

  ?N=0.

%make_registration_repeat_courses(?N, ?Student,?Year,?Semester ):-

  ?N=1,

  %make_single_registration_repeat_courses(?Student, ?Year, ?Semester).


%make_registration_repeat_courses(?N, ?Student,?Year,?Semester ):-

  ?N>1,

  %make_single_registration_repeat_courses(?Student, ?Year, ?Semester),
  ?N1 \is ?N-1,

  %make_registration_repeat_courses(?N1, ?Student,?Year,?Semester).

%make_registration_new_courses(?N, ?_Student,?_Year,?_Semester ):-

  ?N=0.

%make_registration_new_courses(?N, ?Student,?Year,?Semester ):-

  ?N=1,

  %make_single_registration_new_courses(?Student, ?Year, ?Semester).

%make_registration_new_courses(?N, ?Student,?Year,?Semester ):-

  ?N>1,

  %make_single_registration_new_courses(?Student, ?Year, ?Semester),

  ?N1 \is ?N-1,

  %make_registration_new_courses(?N1, ?Student,?Year,?Semester).

%sorted_grades([NG,F,D_MINUS,D,D_PLUS,C_MINUS,C,C_PLUS,B_MINUS,B,
B_PLUS,A_MINUS,A]).

%make_single_registration_specific_course(?SE_for_failed_course,?Student, ?Year,
?Semester):-

   %find_course_opening(?Student,  ?SE_for_failed_course, no_check, ?CO1),

   ?CO1[ ofCourse -> ?Course1]@m1,

   \+ course_assigned(?Student, ?Year, ?Semester, ?SE_for_failed_course, ?_Course
)@m2,

   \+ course_assigned(?Student, ?Year, ?Semester, ?_SE_for_failed_course,
?Course1 )@m2,

   newoid{?RegResultID},

   t_insert{?RegResultID[ student->?Student:Student,

                    year-> ?Year,

                    semester -> ?Semester:Semester,

                    syllabusEntry -> ?SE_for_failed_course,

                    courseOpening->?CO1:CourseOpening]:RegistrationResult}@m2,

   t_insert{course_assigned(?Student, ?Year, ?Semester, ?SE_for_failed_course,
?Course1)}@m2.
```

```
%make_single_registration_repeat_courses(?Student, ?Year, ?Semester):-
  %sorted_grades(?SG),
  %member3(?PrevGrade, ?SG),
  ?_Transcript[ student -> ?Student,
            semesterData -> ?_TSD1[ transcriptEntry -> ?_TE1[ syllabusEntry ->
?SE1,
                                               letterGrade -> ?PrevGrade
                                              ]: TranscriptEntry
                          ]
           ],
  %find_course_opening(?Student, ?SE1, no_check,  ?CO1),
  ?CO1[ ofCourse-> ?Course1]:CourseOpening@m1,
  \+ course_assigned(?Student, ?Year, ?Semester,?SE1, ?_Course1 )@m2,
  \+ course_assigned(?Student, ?Year, ?Semester,?_SE1, ?Course1 )@m2,
  newoid{?RegResultID},
  t_insert{?RegResultID[ student->?Student:Student,
                year-> ?Year,
                semester -> ?Semester:Semester,
                syllabusEntry -> ?SE1,
                courseOpening->?CO1:CourseOpening]:RegistrationResult}@m2,
  t_insert{course_assigned(?Student, ?Year, ?Semester,?SE1, ?Course1)}@m2,
  ?Course1[credits -> ?Credits],
  \if (?Credits=0 ; ?Credits=1) \then
%make_single_registration_new_courses(?Student, ?Year, ?Semester).


%courses_not_taken_by_student(?Student,?CNT):-
  ?Student[mostRecentRealAcademicSemester-> ?MRRAS],
  ?MRRAS = 0,
  !,
  ?Student[inProgram -> ?Program]:Student,
  ?Program[syllabus-> ?Syllabus]:AcademicProgram,
  ?CNT = setof{ ?SE(asc) |  ?Syllabus[ syllabusEntry -> ?SE],
                       \+(
                         course_assigned(?Student, ?_Year, ?_Semester,?SE,
?_Course)@m2
                         )
           }.
%courses_not_taken_by_student(?Student,?CNT):-
  ?Student[mostRecentRealAcademicSemester-> ?MRRAS],
  ?Student[inProgram -> ?Program]:Student,
  ?Program[syllabus-> ?Syllabus]:AcademicProgram,
  ?Transcript[student -> ?Student]:Transcript,
  ?CNT = setof{ ?SE(asc) |  ?Syllabus[ syllabusEntry -> ?SE],
                         \+ (
                             ?Transcript[ semesterData -> ?TSD:TranscriptSemesterData],
                             ?TSD[ transcriptEntry -> ?TE:TranscriptEntry],
                             ?TSD[realAcademicTerm -> ?RAT],
                             ?RAT =< ?MRRAS,
                             ?TE[ syllabusEntry -> ?SE]
                            ),
                         \+(
                            course_assigned(?Student, ?_Year, ?_Semester,?SE,
?_Course)@m2
                         )
           }.
%failedCoursesStudentShouldTake(?Student, ?ListOfFailedCourses):-
  ?Transcript[student -> ?Student]:Transcript,
  ?ListOfFailedCourses = setof { ?SE |
                          ?Transcript[ semesterData -> ?_TSD1[ transcriptEntry ->
?_TE1[ syllabusEntry -> ?SE
                                                                      ]: TranscriptEntry
```

```
                                        ]: TranscriptSemesterData
                              ],
                         %taken_SE_with_failing_grade(?Student, ?SE),
                         %find_course_opening(?Student, ?SE, no_check, ?_CO)
                       }.
%make_single_registration_new_courses(?Student, ?Year, ?Semester):-
  %courses_not_taken_by_student(?Student, ?SyllabusEntryList),
  ?SyllabusEntryList=[],
  !.
%make_single_registration_new_courses(?Student, ?Year, ?Semester):-
  %courses_not_taken_by_student(?Student, ?SyllabusEntryList0),
  %moveToFrontLowCreditCourses(?SyllabusEntryList0, ?SyllabusEntryList),
  %member3(?ASyllabusEntry, ?SyllabusEntryList),
  %find_course_opening(?Student, ?ASyllabusEntry, check, ?CO1),
  ?CO1[ ofCourse-> ?Course1]:CourseOpening@m1,
  \+ course_assigned(?Student, ?Year, ?Semester,?_SE1, ?Course1 )@m2,
  %studentTakenAllPrerequisites(?Student, ?Course1),
  newoid{?RegResultID},
  t_insert{?RegResultID[ student->?Student:Student,
                    year-> ?Year,
                    semester -> ?Semester:Semester,
                    syllabusEntry -> ?ASyllabusEntry,
                    courseOpening->?CO1:CourseOpening]:RegistrationResult}@m2,
  t_insert{course_assigned(?Student, ?Year, ?Semester,?ASyllabusEntry,
?Course1)}@m2,
  ?Course1[credits -> ?Credits],
  \if (?Credits=0 ; ?Credits=1) \then
%make_single_registration_new_courses(?Student, ?Year, ?Semester).
%moveToFrontLowCreditCourses(?SyllabusEntryList0, ?SyllabusEntryList):-
  ?SyllabusEntryList0[length->?HowManyCoursesRemaining]@\btp,
  ?HowManyCoursesTakenThisSemester = count{?RR |
?RR:RegistrationResult@m2},

  (?HowManyCoursesRemaining+?HowManyCoursesTakenThisSemester)>12,

  !,

  ?SyllabusEntryList = ?SyllabusEntryList0.

%moveToFrontLowCreditCourses(?SyllabusEntryList0, ?SyllabusEntryList):-

  ?LowCreditCourseSEs = setof{ ?SE | %member3(?SE,?SyllabusEntryList0),

                         ?SE[course->?Course],

                         ?Course[credits->?Credits],

                         ?Credits <2},

  %setSubtract(?SyllabusEntryList0, ?LowCreditCourseSEs, ?Temp),

  %appX(?LowCreditCourseSEs,  ?Temp, ?SyllabusEntryList).

%studentTakenAllPrerequisites(?Student, ?Course):-

  ?Prerequisites = setof { ?Pre | ?Course[hasPrerequisite -> ?Pre]},

  %takenCoursesWithPassingGrade(?Student, ?Prerequisites).


%takenCoursesWithPassingGrade(?_Student,?Prerequisites):- ?Prerequisites=[],!.

%takenCoursesWithPassingGrade(?Student,[?Pre| ?OtherPrerequisites]):-

  %takenCourseWithPassingGrade(?Student, ?Pre),

  %takenCoursesWithPassingGrade(?Student,?OtherPrerequisites).

%takenCourseWithPassingGrade(?Student, ?Course):-

  %mostRecentGradeRealSem(?Student, ?Course, ?Grade),

  ?Grade : PassingGrade.

%find_course_opening(?Student, ?ASyllabusEntry, ?CheckPrevFlag, ?CO):-

  ?ASyllabusEntry[ courseType -> TurkishOrHistoryCourse],

  \if (?Student[turkishOrForeign -> turkish])

     \then (

          ?Course : HistoryCourse,

          \if (?CheckPrevFlag = check)

             \then (
                 \+ %takenCourseWithPassingGrade(?Student, ?Course)
                 )
          )
  \else (   ?Course : TurkishCourse,
          \if (?CheckPrevFlag = check)
             \then (
                  \+ %takenCourseWithPassingGrade(?Student, ?Course)
                 )
),
  ?CO[ofCourse -> ?Course]@m1.
%find_course_opening(?Student,?ASyllabusEntry, ?CheckPrevFlag, ?CO):-
  ?ASyllabusEntry[ courseType -> CmpeAreaElective],
  !,
  ?Course : CmpeAreaElective,
  \if (?CheckPrevFlag = check)
     \then (\+ %takenCourseWithPassingGrade(?Student, ?Course)),
  \+ course_assigned(?Student, ?_Year, ?_Semester,?_ASyllabusEntry,
?Course)@m2,
  ?CO[ofCourse -> ?Course]@m1.
%find_course_opening(?Student,?ASyllabusEntry,?CheckPrevFlag, ?CO):-
   ?ASyllabusEntry[ courseType -> CmpeUniversityElective],
   !,
   ?Course : CmpeUniversityElective,
   \if (?CheckPrevFlag = check)
      \then (\+ %takenCourseWithPassingGrade(?Student, ?Course)),
   \+ course_assigned(?Student, ?_Year, ?_Semester,?_ASyllabusEntry,
?Course)@m2,
   ?CO[ofCourse -> ?Course]@m1.
%find_course_opening(?Student,?ASyllabusEntry,?CheckPrevFlag, ?CO):-
   ?ASyllabusEntry[ courseType -> ?CourseType],
   \if (?CourseType = NormalCourse)
      \then (  ?ASyllabusEntry[course -> ?Course],
           )
   \else (  ?Course : ?CourseType,
        ),
   \if (?CheckPrevFlag = check)
      \then (\+ %takenCourseWithPassingGrade(?Student, ?Course)),
   ?CO[ofCourse -> ?Course]@m1.
%taken_SE_with_failing_grade(?Student, ?SE):-
  %mostRecentGradeSERealSem(?Student, ?SE, ?Grade),
  ?Grade: FailingGrade.
%taken_SE_with_passing_grade(?Student, ?SE):-
  %mostRecentGradeSERealSem(?Student, ?SE, ?Grade),
  ?Grade: PassingGrade.
%mostRecentGradeRealSem(?Student, ?Course, ?Grade):-
  ?Student[mostRecentRealAcademicSemester -> ?MRRAS],
  ?Transcript[student -> ?Student]:Transcript,
  %findMostRecentGradeRealSem(?Transcript, ?MRRAS, ?Course, ?Grade).
%findMostRecentGradeRealSem(?Transcript, ?Term,  ?Course, ?Grade):-
  ?Transcript[ semesterData -> ?TSD:TranscriptSemesterData],
  ?TSD[ realAcademicTerm -> ?Term,
      transcriptEntry -> ?TE:TranscriptEntry],
  ?TE[ courseTaken -> ?Course1,
     letterGrade -> ?Grade1],
  ?Course1 = ?Course,
  ?Grade = ?Grade1,
  !.
%findMostRecentGradeRealSem(?Transcript, ?Term,  ?Course, ?Grade):-
  ?Term > 0,
  ?Term1 \is ?Term -1,
  %findMostRecentGradeRealSem(?Transcript, ?Term1,  ?Course, ?Grade).
%mostRecentGradeSERealSem(?Student, ?SE, ?Grade):-
  ?Student[mostRecentRealAcademicSemester -> ?MRRAS],
  ?Transcript[student -> ?Student]:Transcript,
  %findMostRecentGradeSERealSem(?Transcript, ?MRRAS, ?SE, ?Grade).
%findMostRecentGradeSERealSem(?Transcript, ?Term,  ?SE, ?Grade):-
  ?Transcript[ semesterData -> ?TSD:TranscriptSemesterData],
  ?TSD[ realAcademicTerm -> ?Term,
  transcriptEntry -> ?TE:TranscriptEntry],
  ?TE[ syllabusEntry -> ?SE,
      letterGrade -> ?Grade], !.
%findMostRecentGradeSERealSem(?Transcript, ?Term,  ?SE, ?Grade):-
  ?Term > 0,
  ?Term1 \is ?Term -1,
  %findMostRecentGradeSERealSem(?Transcript, ?Term1,  ?SE, ?Grade).
```

B3: Source Code for the Optimization Component

```
%r3:-
   [ +'D://Flora//transcript'],
   [ +'D://Flora//registration'],
   [ +'D://Flora//student_instances'],
   [ +'D://Flora//regular_transcript_instances'],
   [ +'D://Flora//syllabus_instances'],
   [ +'D://Flora//classroom_instances'],
   [ +'D://Flora//Concepts'],
   [ +'D://Flora//address_instances'],
   [ +'D://Flora//course_instances'],
   [ +'D://Flora//course_opening_instances'>>m1],
   [ +'D://Flora//room_day_period_instances'],
   [ +'D://Flora//instances'],
   [ +'D://Flora//CourseCodes'],
   [ +'D://Flora//utilities'],
   [ +'D://Flora//demet_transcript_instances'],
   [ +'D://Flora//regular_transcript_instances'],
   [ +'D://Flora//constraints'],
   [ +'D://Flora//empty'>>m1],
   [ +'D://Flora//course_opening_instances'>>m1],
   [ +'D://Flora//empty'>>m2],
   [ +'D://Flora//empty'>>m3].
%how_many_clashes_all(?Student, ?Year, ?Semester, ?CO_list_all_sorted):-
  ?CO_list_all = setof { ?P |
                        %how_many_clashes(?Student,?Year,?Semester,
?Course_CO_pair_list, ?HM),
                        ?P = (?Course_CO_pair_list, ?HM)
                    },
  %isort(?CO_list_all, ?CO_list_all_sorted),
  \if (mode(debug2)) \then ( writeln (sorted - ?CO_list_all_sorted)@\prolog),
  %showCOsList(?CO_list_all_sorted).
%how_many_clashes(?Student, ?Year, ?Semester, ?Course_CO_pair_list, ?HM):-
  %courseOpeningsForStudent(?Student,?Year, ?Semester, ?Course_CO_pair_list),
  t_deleteall{CO_assigned_to_student(?Student,?Year, ?Semester, ?_CO)@m3},
  %insertCourseOpeningsForStudent(?Student,?Year,
?Semester,?Course_CO_pair_list),
  ?HM0 = sum { ?HMClashes |
                       CO_assigned_to_student(?Student,?Year, ?Semester, ?CO1)@m3,
                       CO_assigned_to_student(?Student,?Year, ?Semester, ?CO2)@m3,
                       \+ (?CO1 = ?CO2),
                       %howManyClashesTwoCO(?CO1, ?CO2, ?HMClashes)
            },
  ?HM \is ?HM0/2.
%howManyClashesTwoCO(?CO1, ?CO2, ?HM):-
  ?HM = sum { ?Z | ?CO1:CourseOpening@m1,
                ?CO1[teachingTimes -> ?RDPD1]@m1,
                ?RDPD1:RoomDayPeriodDuration@m1,
                ?RDPD1[day -> ?Day1, period -> ?Period1]@m1,
                ?CO2:CourseOpening@m1,
                ?CO2[teachingTimes -> ?RDPD2]@m1,
                ?RDPD2:RoomDayPeriodDuration@m1,
                ?RDPD2[day -> ?Day2, period -> ?Period2]@m1,
                ?Day1 = ?Day2,
                ?Period1 = ?Period2,
                ?Z=1}.
%courses_assigned_to_student(?Student,?Year, ?Semester,?Course_list):-
  ?Course_list = setof{  ?Course |
                          ?RR:RegistrationResult@m2,
                          ?RR[student -> ?Student,
                                year -> ?Year,

                                semester -> ?Semester,

                                courseOpening -> ?CO]@m2,

                              ?CO[ofCourse->?Course]@m1  }.

%courseOpeningsForStudent(?Student, ?Year, ?Semester, ?CO_list):-

  %courses_assigned_to_student(?Student,?Year, ?Semester,?Course_list),

  %courseOpeningsForStudent0(?Year, ?Semester,?Course_list, ?CO_list).

%courseOpeningsForStudent0(?_Year,?_Semester,?Course_list,
?Course_CO_pair_list):-

  ?Course_list =[],

  ?Course_CO_pair_list=[].

%courseOpeningsForStudent0(?Year,?Semester,[?Course|?Course_list],
?Course_CO_pair_list):-

  ?CO[ofCourse -> ?Course, year->?Year, semester->?Semester]@m1,

  ?Course_CO_pair_list = [(?Course,?CO) | ?Rest_CO_List],

  %courseOpeningsForStudent0(?Year, ?Semester, ?Course_list, ?Rest_CO_List).

%insertCourseOpeningsForStudent(?_Student,?_Year,?_Semester,
?Course_CO_pair_list):-

   ?Course_CO_pair_list = [].

%insertCourseOpeningsForStudent(?Student,?Year,?Semester,
[?Course_CO_pair|?Course_CO_pair_list]):-

   ?Course_CO_pair = (?_Course, ?CO),

   t_insert{ CO_assigned_to_student(?Student,?Year, ?Semester, ?CO) }@m3,

   %insertCourseOpeningsForStudent(?Student,?Year,?Semester,
?Course_CO_pair_list).
```

B4: Source Code for the Utilities and Constraints

```
load_instances:-
   [ +'D://Flora//classroom_instances'],
   [ +'D://Flora//address_instances'],
   [ +'D://Flora//course_instances'],
   [ +'D://Flora//room_day_period_instances'],
   [ +'D://Flora//instances'],
   [ +'D://Flora//course_opening_instances'],
   [ +'D://Flora//CourseCodes'],
   [ +'D://Flora//course_opening_requests'],
   [ +'D://Flora//syllabus_instances'].
pp_COs :-
  ?_x = setof{ ?Y | ?Y = 1,
                ?CO[year->?Year,
                   groupNo->?GroupNo,
                   ofCourse->?OfCourse,
                   semester->?Semester]:CourseOpening@m1,
                 pp_CO(?CO, ?Year, ?GroupNo, ?OfCourse, ?Semester)
           }.
pp_CO(?CO, ?Year, ?GroupNo, ?OfCourse, ?Semester):-
  writeln('\#:CourseOpening[')@\prolog,
  write('     groupNo ->' )@\prolog, write(?GroupNo)@\prolog, writeln(',')@\prolog,
  write('     ofCourse -> ')@\prolog, write(?OfCourse)@\prolog, writeln(',')@\prolog,
  write('     year -> ')@\prolog, write(?Year)@\prolog, writeln(',')@\prolog,
  write('     semester -> ' )@\prolog, write(?Semester)@\prolog, writeln(',')@\prolog,
  ?_X = setof{ ?Y | ?Y = 1,
                ?CO[teachingTimes->?TT]@m1,
                ?TT[room->?Room, day->?Day, period->?Period,
                       duration->?Duration]@m1,
                write('     teachingTimes ->' )@\prolog,  writeln('\#[')@\prolog,

                                write('             room -> ')@\prolog,
write(?Room)@\prolog, write(',')@\prolog, nl@\prolog,

                                write('             day -> ')@\prolog, write(?Day)@\prolog,
write(',')@\prolog, nl@\prolog,

                                write('             period -> ')@\prolog,
write(?Period)@\prolog, write(',')@\prolog, nl@\prolog,

                                write('             duration -> ')@\prolog,
write(?Duration)@\prolog, writeln('],')@\prolog,

                                nl@\prolog,

                nl@\prolog },

            writeln('].')@\prolog.

member2(?X,[?Y|?_A]):-?X=?Y.

member2(?X,[?_H|?R]):-member2(?X,?R).

%member3(?X,[?Y|?_A]):-?X=?Y.

%member3(?X,[?_H|?R]):-%member3(?X,?R).

member5(?X,[(?_,?Y, ?_2, ?_3, ?_4)|?_A]):-?X=?Y,!.

member5(?X,[(?_,?_1, ?Y, ?_3, ?_4)|?_A]):-?X=?Y,!.

member5(?X,[(?_,?_1, ?_2, ?Y, ?_4)|?_A]):-?X=?Y,!.

member5(?X,[(?_,?_1, ?_2, ?_3, ?Y)|?_A]):-?X=?Y,!.

member5(?X,[?_H|?R]):-member5(?X,?R).

%printlist([]):- nl@\prolog.

%printlist([?H|?T]):- writeln(?H)@\prolog, %printlist(?T).

app([],?L,?L2):-?L2=?L.

app([?H1|?T],?L,[?H2|?T2]):- ?H1=?H2 , app(?T,?L,?T2).

%appX([],?L,?L2):-?L2=?L.

%appX([?H1|?T],?L,[?H2|?T2]):- ?H1=?H2 , %appX(?T,?L,?T2).

%setSubtract(?L1, [],?L2):-  ?L2=?L1, !.

%setSubtract(?L1, [?H|?T], ?Result):- ?L1[delete(?H)-> ?Temp]@\btp,

                            %setSubtract(?Temp, ?T, ?Result).
```

```
%semester_is_before(?Year1, ?_Semester1,  ?Year2, ?_Semester2):-

?Year1 < ?Year2, !.

%semester_is_before(?Year1, ?Semester1,  ?Year2, ?Semester2):-  ?Year1 = ?Year2,

                                        ?Semester1=spring,

                                        ( ?Semester2 = summer ; ?Semester2 = fall),

                                        !.

%semester_is_before(?Year1, ?Semester1,  ?Year2, ?Semester2):-  ?Year1 = ?Year2,

                                        ?Semester1=summer,

                                        ?Semester2 = fall.

%year_semester_subtract(?Y1,?S1,?Y2,?S2, ?R):- ?Y1=?Y2, ?S1=?S2, ?R=0, !.

%year_semester_subtract(?Y1,?S1,?Y2,?S2, ?R):-
%semester_is_before(?Y2,?S2,?Y1,?S1),

                                %get_sem_before(?Y1,?S1, ?PrevYear, ?PrevSem),

                                %year_semester_subtract(?PrevYear,?PrevSem, ?Y2,
?S2, ?Temp),

                                ?R \is ?Temp + 1.

%get_sem_before(?Y1,?S1, ?PrevYear, ?PrevSem):- ?S1=fall,

                                 !,

                                 ?PrevSem = spring,

                                 ?PrevYear = ?Y1.

%get_sem_before(?Y1,?S1, ?PrevYear, ?PrevSem):- ?S1=spring,

                                 !,

                                 ?PrevSem = fall,

                                 ?PrevYear \is ?Y1-1.

%showCOs(?Course_CO_pair_list):- ?Course_CO_pair_list = [],

writeln(' ')@\prolog.

%showCOs(?Course_CO_pair_list):-

          ?Course_CO_pair_list = [?CourseCOPair|?CourseCOPair_list],

          ?CourseCOPair = (?Course, ?CO),

          ?CO:CourseOpening@m1,
                    ?TeachingTimes = setof {?TT | ?CO[ teachingTimes -> ?TT]@m1},
                    write(?Course - ' ')@\prolog,
                    %showTeachingTimes(?TeachingTimes),
                    writeln(' ')@\prolog,
                    %showCOs(?CourseCOPair_list).
%showCOsList(?X):- ?X=[], nl@\prolog.
%showCOsList([?First|?Rest]):-
   ?First = (?Course_CO_pair_list , ?HM_Clashes),
   %showCOs(?Course_CO_pair_list),
   writeln([with,?HM_Clashes, clashes])@\prolog,
   writeln('**********************************')@\prolog,
   %showCOsList(?Rest).
%showTeachingTimes(?TT_list):- ?TT_list = [],  writeln(' ==== ')@\prolog.
%showTeachingTimes([?TT|?TT_list]):-
   ?TT: RoomDayPeriodDuration@m1,
   ?TT[day -> ?Day, period -> ?Period]@m1,
   write(?Day - ?Period - ' ')@\prolog,
   %showTeachingTimes(?TT_list).
%showAllCOs(?COs):-
   ?COs = setof {?X | ?CO:CourseOpening@m1,
                   ?CO[ofCourse->?Course]@m1,
                   ?X= (?Course, ?CO)}, %showCOs(?COs).
%isort([],?R):- ?R=[], !.
%isort([?H|?T], ?Result):-
  %isort(?T,?Temp),
  %insert(?H, ?Temp, ?Result).
%insert(?X,[],?R):- ?R = [?X], !.
%insert(?X,[?H|?T],?R):- %is_smaller_or_equal(?X,?H),
                    !,
                    ?R = [?X, ?H | ?T].
%insert(?X,[?H|?T],?R):- %insert(?X, ?T, ?Temp),
                    ?R = [ ?H | ?Temp].
%is_smaller_or_equal(?F,?S):- ?F=(?_CO_LIST1, ?HM1),
                        ?S=(?_CO_LIST2, ?HM2),
                        ?HM1 =< ?HM2.
%alreadyTakenRDPD(?Room, ?Day, ?Period):-
  ?_RDPD[room->?Room, day->?Day,
 period->?Period]:RoomDayPeriodDuration@m1,!.
%onWednesdayAfternoon(?Day,?Period):-
  ?Day=wednesday,
  ( ?Period = 7 ; ?Period = 8).
%clashesWithSameYearCourse( ?Program, ?Course1,?Day1, ?Period1):-
  ?_CourseOpening[ofCourse->?Course2, teachingTimes->?_RDPD[day-
>?Day2,period->?Period2]]:CourseOpening@m1,
  ?Course1 \= ?Course2,
  \+( ?Course1[hasPrerequisite->?Course2] ),
  \+( ?Course2[hasPrerequisite->?Course1] ),
  ?Day1 = ?Day2,
  ?Period1 = ?Period2,
  ?Program: AcademicProgram,
  ?Program[syllabus -> ?Syllabus],
  %courseInSyllabus(?Syllabus, ?_SyllabusEntry1, ?Course1, ?Year1,
?Semester1,?CourseType1),
  %courseInSyllabus(?Syllabus, ?_SyllabusEntry2, ?Course2, ?Year2,
?Semester2,?CourseType2),
  (( ?Year1 = ?Year2,?Semester1 = ?Semester2, ?CourseType1 = NormalCourse,
?CourseType2=NormalCourse) ;
  ( 3=4, ?CourseType1 = CmpeAreaElective, ?CourseType2 = CmpeAreaElective);
  ( 3=4, ?CourseType1 = CmpeAreaElective, ?Year2 = senior);
  ( 3=4, ?CourseType2 = CmpeAreaElective, ?Year1 = senior)   ),
  !.

%courseInSyllabus(?Syllabus, ?SyllabusEntry, ?Course, ?Year,
?Semester,?CourseType ):-

 ?Syllabus:Syllabus,

 ?Syllabus[syllabusEntry -> ?SyllabusEntry],

 ?SyllabusEntry[course -> ?Course,

                whichYear -> ?Year,

                whichSemester -> ?Semester,

                courseType -> ?CourseType],

                !.
```

Appendix C: Source Code for the Instances

C1: Instances of the Transcript Knowledge Base

```
ayse_transcript: Transcript.
ayse_transcript[student-> std066161,
            syllabus -> cmpeUndergraduateSyllabus,
            semesterData -> ayse_aa1,
            semesterData -> ayse_aa2,
            semesterData -> ayse_aa3,
            semesterData -> ayse_aa4,
            semesterData -> ayse_aa5,
            semesterData -> ayse_aa6,
            semesterData -> ayse_aa7,
            semesterData -> ayse_aa8,
            registrationStatus -> registered
            ].
ayse_aa1 : TranscriptSemesterData.
ayse_aa1[
      whichYear->2006,
      whichSemester->fall,
      registrationStatus -> registered
     ].
ayse_aa1 [
       transcriptEntry-> {
                    \#[syllabusEntry-> SE_25711,
                       letterGrade-> D]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25712,
                       letterGrade-> D]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25713,
                       letterGrade-> D]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25714,
                      letterGrade-> D]:TranscriptEntry,
                    \#[syllabusEntry-> SE_25715,
                      letterGrade-> D]:TranscriptEntry
                   }
      ].
ayse_aa2 : TranscriptSemesterData.
ayse_aa2[
      whichYear->2007,
      whichSemester->spring,
      registrationStatus -> registered
      ].
ayse_aa2 [
       transcriptEntry-> {
                     \#[syllabusEntry-> SE_25721,
                      letterGrade-> S]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25722,
                      letterGrade-> D ]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25723,
                      letterGrade-> B]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25724,
                      letterGrade-> A]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25725,
                      letterGrade-> A]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25726,
                      actualCourse->hist280,
                      letterGrade-> A]:TranscriptEntry
                   }
      ].
```

```
ayse_aa3 : TranscriptSemesterData.
ayse_aa3[
          whichYear->2007,
          whichSemester->fall,
          registrationStatus -> registered
         ].
ayse_aa3 [
           transcriptEntry-> {
                                \#[syllabusEntry-> SE_25731,
                                  letterGrade-> D]:TranscriptEntry,
                                \#[syllabusEntry-> SE_25732,
                                  letterGrade-> D]:TranscriptEntry,
                                \#[syllabusEntry-> SE_25733,
                                  letterGrade-> D_PLUS]:TranscriptEntry,
                                \#[syllabusEntry-> SE_25734,
                                  letterGrade-> C_MINUS]:TranscriptEntry,
                               \#[syllabusEntry-> SE_25735,
                                  letterGrade-> C]:TranscriptEntry
                              }
         ].
ayse_aa4 : TranscriptSemesterData.
ayse_aa4[
          whichYear->2008,
          whichSemester->spring,
          registrationStatus -> registered
         ].
ayse_aa4 [
           transcriptEntry-> {
                                  \#[syllabusEntry-> SE_25741,
                              letterGrade-> D]:TranscriptEntry,
                            \#[syllabusEntry-> SE_25742,
                              letterGrade-> A]:TranscriptEntry,
                            \#[syllabusEntry-> SE_25743,
                              letterGrade-> B]:TranscriptEntry,
                            \#[syllabusEntry-> SE_25744,
                              letterGrade-> B]:TranscriptEntry,
                            \#[syllabusEntry-> SE_25745,
                              actualCourse->chem101,
                              letterGrade-> A]:TranscriptEntry
                           }
       ].
ayse_aa5 : TranscriptSemesterData.
ayse_aa5[
        whichYear->2008,
        whichSemester->fall,
        registrationStatus -> registered
       ].
ayse_aa5 [
         transcriptEntry-> {
                           \#[syllabusEntry-> SE_25751,
                              letterGrade-> C]:TranscriptEntry,
                            \#[syllabusEntry-> SE_25752,
                              letterGrade-> D]:TranscriptEntry,
                            \#[syllabusEntry-> SE_25753,
                              letterGrade-> B]:TranscriptEntry,
                            \#[syllabusEntry-> SE_25754,
                              letterGrade-> C_MINUS]:TranscriptEntry,
                            \#[syllabusEntry-> SE_25755,
                      letterGrade-> B_PLUS]:TranscriptEntry
                    }
          ].
ayse_aa6 : TranscriptSemesterData.
ayse_aa6[
          whichYear->2009,
          whichSemester->spring,
          registrationStatus -> registered
         ].
ayse_aa6 [
           transcriptEntry-> {
                    \#[syllabusEntry-> SE_25761,
                      letterGrade-> A]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25762,
                      letterGrade-> B]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25763,
                      letterGrade-> B]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25764,
                      letterGrade-> A_MINUS]:TranscriptEntry,
                     \#[syllabusEntry-> SE_25765,
                      actualCourse->nutd121,
                      letterGrade-> B_PLUS]:TranscriptEntry
                    }
          ].
ayse_aa7 : TranscriptSemesterData.
ayse_aa7[
          whichYear->2009,
          whichSemester->fall,
          registrationStatus -> registered].
```

```
ayse_aa7 [
          transcriptEntry-> {
                            \#[syllabusEntry-> SE_25772,
                                  actualCourse->cmpe418 ,
                                  letterGrade-> B_PLUS]:TranscriptEntry,
                               \#[syllabusEntry-> SE_25773,
                                  actualCourse->cmpe466,
                                  letterGrade-> B]:TranscriptEntry,
                               \#[syllabusEntry-> SE_25774,
                                  actualCourse->cmpe415,
                                  letterGrade-> A]:TranscriptEntry,
                               \#[syllabusEntry-> SE_25775,
                                  letterGrade-> A]:TranscriptEntry,
                                \#[syllabusEntry-> SE_25776,
                                  letterGrade-> A]:TranscriptEntry
                               }
        ].
ayse_aa8 : TranscriptSemesterData.

ayse_aa8[
         whichYear->2010,
         whichSemester->spring,
         registrationStatus -> registered
        ].
ayse_aa8 [
          transcriptEntry-> { \#[syllabusEntry-> SE_25781,
                                  actualCourse->cmpe423,
                                  letterGrade-> A]:TranscriptEntry,
                               \#[syllabusEntry-> SE_25782,
                                  actualCourse->cmpe416,
                                  letterGrade-> A]:TranscriptEntry,
                               \#[syllabusEntry-> SE_25783,
                                  actualCourse->soc101,
                                  letterGrade-> A]:TranscriptEntry,
                               \#[syllabusEntry-> SE_25784,
                                  actualCourse->psy101,
                                  letterGrade-> A]:TranscriptEntry,
                               \#[syllabusEntry-> SE_25785,
                                  letterGrade-> A]:TranscriptEntry,
                               \#[syllabusEntry-> SE_25771,
                                  letterGrade-> S]:TranscriptEntry,
                                \#[syllabusEntry-> SE_25777,
                                  actualCourse->ieng355,
                                  letterGrade-> B]:TranscriptEntry
                               }
        ].
```

C2: Instances of the Registration Knowledge Base

```
cmpe100:Course[
        courseCode->"cmpe100",
        courseName->"Introduction to Computer Engineering",
        // hasPrerequisite->none,
        lecture_hours->2,
        lab_hours->0,
        credits->0,
        instructionLanguage->english,
        ects->2].
cmpe101: Course [
        courseCode->"cmpe101",
        courseName->"Foundations of Computer Engineering",
        // hasPrerequisite->none,
        lecture_hours->3,
        lab_hours->1,
        credits->3,
        instructionLanguage->english,
        ects->6].
cmpe108: Course [
        courseCode->"cmpe108",
        courseName->"Algorithms and Programming",
        // hasPrerequisite->none,
        lecture_hours->3,
        lab_hours->1,
        credits->3,
        instructionLanguage->english,
        ects->6].
```

```
cmpe110: Course [
        courseCode->"cmpe110",
        courseName->"Fundamentals of Programming",
        // hasPrerequisite->none,
        lecture_hours->3,
        lab_hours->1,
        credits->3,
        instructionLanguage->english,
        ects->6].
math163: Course [
        courseCode->"math163",
        courseName->"Discrete Mathematics",
        // hasPrerequisite->none,
        lecture_hours->3,
        lab_hours->1,
        credits->3,
        instructionLanguage->english,
        ects->5].
engl191: Course [
        courseCode->"engl191",
        courseName->"Communication in English I",
        // hasPrerequisite->none,
        lecture_hours->3,
        lab_hours->1,
        credits->3,
        instructionLanguage->english,
        ects->4].
```

```
math151: Course [
        courseCode->"math151",
        courseName->"Calculus I",
        // hasPrerequisite->none,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->7].
phys101: Course [
        courseCode->"phys101",
        courseName->"Physics I",
        // hasPrerequisite->none,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->7].
cmpe112: Course [
        courseCode->"cmpe112",
        courseName->"Programming Fundamentals",
        hasPrerequisite->cmpe101,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->7].
```

```
engl192: Course [
        courseCode->"engl192",
        courseName->"Communication in English II",
        hasPrerequisite->engl191,
        lecture_hours->3,
        lab_hours->1,
        credits->3,
        instructionLanguage->english,
        ects->4].
math152: Course [
        courseCode->"math152",
        courseName->"Calculus II",
        hasPrerequisite->math151,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->7].
phys102: Course [
        courseCode->"phys102",
        courseName->"Physics II",
        hasPrerequisite->phys101,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->6].
```

```
tusl181: Course [
        courseCode->"tusl181",
        courseName->"Turkish as a second language",
        // hasPrerequisite->none,
        lecture_hours->2,
        lab_hours->0,
        credits->2,
        instructionLanguage->english,
        ects->3].
hist280: Course [
        courseCode->"hist280",
        courseName->"History of Turkish Reforms",
        // hasPrerequisite->none,
        lecture_hours->2,
        lab_hours->0,
        credits->2,
        instructionLanguage->turkish,
        ects->3].
cmpe223: Course [
        courseCode->"cmpe223",
        courseName->"Digital Logic Design",
        hasPrerequisite->math163,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->7].
```

```
cmpe231: Course [
        courseCode->"cmpe231",
        courseName->"Data Structures",
        hasPrerequisite->cmpe112,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->7].
cmpe211: Course [
        courseCode->"cmpe211",
        courseName->"Object-Oriented Programming",
        hasPrerequisite->cmpe112,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->7].
engl201: Course [
        courseCode->"engl201",
        courseName->"Communication skills",
        hasPrerequisite->engl192,
        lecture_hours->3,
        lab_hours->1,
        credits->3,
        instructionLanguage->english,
        ects->4].
```

```
math241: Course [
        courseCode->"math241",
        courseName->"Linear Algebra and Ordinary Diff. Equations",
        hasPrerequisite->math151,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->6].
cmpe224: Course [
        courseCode->"cmpe224",
        courseName->"Digital Logic System",
        hasPrerequisite->cmpe223,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->7].
cmpe226: Course [
        courseCode->"cmpe226",
        courseName->"Electronics for Computer Engineers",
        hasPrerequisite->math241,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->6].
```

```
cmpe242: Course [
        courseCode->"cmpe242",
        courseName->"Operating Systems",
        hasPrerequisite->cmpe112,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->6].
math373: Course [
        courseCode->"math373",
        courseName->"Numerical Analysis for Engineers",
        hasPrerequisite->math241,
        lecture_hours->3,
        lab_hours->1,
        credits->3,
        instructionLanguage->english,
        ects->6].
biol124: Course [
        courseCode->"biol124",
        courseName->"Introduction to Molecular Biology & Genetics ",
        // hasPrerequisite->none,
        lecture_hours->3,
        lab_hours->1,
        credits->3,
        instructionLanguage->english,
        ects->6].
```

```
chem101: Course [
        courseCode->"chem101",
        courseName->"General Chemistry",
        // hasPrerequisite->none,
        lecture_hours->3,
        lab_hours->1,
        credits->3,
        instructionLanguage->english,
        ects->6].
biol105: Course [
        courseCode->"biol105",
        courseName->"Biological Basis of Behavior",
        // hasPrerequisite->none,
        lecture_hours->3,
        lab_hours->1,
        credits->3,
        instructionLanguage->english,
        ects->6].
biol316: Course [
        courseCode->"biol316",
        courseName->"Environmental Management",
        // hasPrerequisite->none,
        lecture_hours->3,
        lab_hours->1,
        credits->3,
        instructionLanguage->english,
        ects->6].
```

```
cmpe323: Course [
        courseCode->"cmpe323",
        courseName->"Microprocessors",
        hasPrerequisite->cmpe224,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->7].
cmpe343: Course [
        courseCode->"cmpe343",
        courseName->"Systems Programming",
        hasPrerequisite->cmpe242,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->6].
cmpe371: Course [
        courseCode->"cmpe371",
        courseName->"Analysis of Algorithms",
        hasPrerequisite->cmpe231,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->7].
```

```
cmpe321: Course [
        courseCode->"cmpe321",
        courseName->"Basics of Signals and Systems",
        hasPrerequisite->cmpe226,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->6].
math322: Course [
        courseCode->"math322",
        courseName->"Probability and Statistical Methods",
        hasPrerequisite->math151,
        lecture_hours->3,
        lab_hours->1,
        credits->3,
        instructionLanguage->english,
        ects->5].
cmpe324: Course [
        courseCode->"cmpe324",
        courseName->"Computer Architecture and Organization",
        hasPrerequisite->cmpe224,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->7].
```

```
cmpe344: Course [
        courseCode->"cmpe344",
        courseName->"Computer Networks",
        hasPrerequisite->cmpe343,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->7].
cmpe354: Course [
        courseCode->"cmpe354",
        courseName->"Database Management Systems",
        hasPrerequisite->cmpe231,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->6].
cmpe318: Course [
        courseCode->"cmpe318",
        courseName->"Principles of Programming Languages",
        hasPrerequisite->cmpe211,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->6].
```

```
cmpe400: Course [
        courseCode->"cmpe400",
        courseName->"Summer Training",
        // hasPrerequisite->none,
        lecture_hours->0,
        lab_hours->0,
        credits->0,
        instructionLanguage->english,
        ects->1].
cmpe418: Course [
        courseCode->"cmpe418",
        courseName->"Internet Programming",
        hasPrerequisite->cmpe354,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->6].
cmpe415: Course [
        courseCode->"cmpe415",
        courseName->"Visual Programming",
        hasPrerequisite->cmpe354,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->6].
```

```
cmpe411: Course [
        courseCode->"cmpe411",
        courseName->"Information Security",
        hasPrerequisite->cmpe354,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->6].
cmpe416: Course [
        courseCode->"cmpe416",
        courseName->"Object-Oriented Programming & Graphical User Interfaces",
        hasPrerequisite->cmpe354,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->6].
cmpe423: Course [
        courseCode->"cmpe423",
        courseName->"Embedded Systems",
        hasPrerequisite->cmpe354,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->6].
```

```
cmpe466: Course [
        courseCode->"cmpe466",
        courseName->"Computer Graphics",
        hasPrerequisite->cmpe354,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->6].
cmpe471: Course [
        courseCode->"cmpe471",
        courseName->"Automata Theory",
        hasPrerequisite->math163,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->6].
cmpe405: Course [
        courseCode->"cmpe405",
        courseName->"Graduation Project I",
        // hasPrerequisite->none,
        lecture_hours->0,
        lab_hours->0,
        credits->1,
        instructionLanguage->english,
        ects->3].
```

```
ieng355: Course [
        courseCode->"ieng355",
        courseName->"Ethics in Engineering",
        // hasPrerequisite->none,
        lecture_hours->3,
        lab_hours->0,
        credits->3,
        instructionLanguage->english,
        ects->4].
phil401: Course [
        courseCode->"phil401",
        courseName->"Ethics in Professional Life",
        // hasPrerequisite->none,
        lecture_hours->3,
        lab_hours->0,
        credits->3,
        instructionLanguage->english,
        ects->4].
phil215: Course [
        courseCode->"phil215",
        courseName->"Applied Ethics",
        // hasPrerequisite->none,
        lecture_hours->3,
        lab_hours->0,
        credits->3,
        instructionLanguage->english,
        ects->4].
```

```
econ101: Course [
        courseCode->"econ101",
        courseName->"Introduction to Economics I",
        // hasPrerequisite->none,
        lecture_hours->3,
        lab_hours->0,
        credits->3,
        instructionLanguage->english,
        ects->4].
fina301: Course [
        courseCode->"fina301",
        courseName->"Financial Management",
        // hasPrerequisite->none,
        lecture_hours->3,
        lab_hours->0,
        credits->3,
        instructionLanguage->english,
        ects->4].
ieng450: Course [
        courseCode->"ieng450",
        courseName->"Management",
        // hasPrerequisite->none,
        lecture_hours->3,
        lab_hours->0,
        credits->3,
        instructionLanguage->english,
        ects->5].
```

```
ieng420: Course [
        courseCode->"ieng420",
        courseName->"Engineering Economy",
        // hasPrerequisite->none,
        lecture_hours->3,
        lab_hours->0,
        credits->3,
        instructionLanguage->english,
        ects->5].
nutd121: Course [
        courseCode->"nutd121",
        courseName->"Healthy Living and Nutrition",
        // hasPrerequisite->none,
        lecture_hours->3,
        lab_hours->0,
        credits->3,
        instructionLanguage->english,
        ects->4].
soc101: Course [
        courseCode->"soc101",
        courseName->"Sociology",
        // hasPrerequisite->none,
        lecture_hours->3,
        lab_hours->1,
        credits->3,
        instructionLanguage->english,
        ects->4].
```

```
psy101: Course [
        courseCode->"psy101",
        courseName->"psychology",
        // hasPrerequisite->none,
        lecture_hours->3,
        lab_hours->1,
        credits->3,
        instructionLanguage->english,
        ects->4].
cmpe406: Course [
        courseCode->"cmpe406",
        courseName->"Graduation Project II",
        hasPrerequisite->cmpe405,
        lecture_hours->0,
        lab_hours->1,
        credits->3,
        instructionLanguage->english,
        ects->7].
cmpe405: CmpeGraduationProject1.
cmpe406: CmpeGraduationProject2.
cmpe400: CmpeSummerTraining.
cmpe466: CmpeAreaElective.
cmpe423: CmpeAreaElective.
cmpe416: CmpeAreaElective.
cmpe415: CmpeAreaElective.
cmpe411: CmpeAreaElective.
cmpe418: CmpeAreaElective.
cmse321: CmpeAreaElective.
cmse326: CmpeAreaElective.
```

```
cmse323: CmpeAreaElective.
biol316: CmpeScienceCourse.
biol105: CmpeScienceCourse.
biol124: CmpeScienceCourse.
chem101: CmpeScienceCourse.
econ101: CmpeFinanceCourse.
fina301: CmpeFinanceCourse.
ieng450: CmpeFinanceCourse.
ieng420: CmpeFinanceCourse.
nutd121: CmpeUniversityElective.
psy101: CmpeUniversityElective.
soc101: CmpeUniversityElective.
ieng355: CmpeEthicsCourse.
phil401: CmpeEthicsCourse.
phil215: CmpeEthicsCourse.
hist280: HistoryCourse.
tusl181: TurkishCourse.
cmse321: Course [
    courseCode->"cmse321",
    courseName->"Software Requirements Analysis and Specifications",
    // hasPrerequisite->none,
    lecture_hours->4,
    lab_hours->1,
    credits->4,
    instructionLanguage->english,
    ects->7].
```

```
cmse323: Course [
        courseCode->"cmse323",
        courseName->"Human Computer Interaction",
        // hasPrerequisite->none,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->7].
cmse326: Course [
        courseCode->"cmse326",
        courseName->"Software Quality Assurance & Testing",
        // hasPrerequisite->none,
        lecture_hours->4,
        lab_hours->1,
        credits->4,
        instructionLanguage->english,
        ects->7].
std066161 : Student [ id->"066161",
                     gender->female,
                     date_of_birth->"1987-11-13"^^\date,
                     name->"Ayse",
                     lastName->"Demir",
                     address->\#[street->"Meydan", city->"Magosa", country-
>"Turkey"]:Address
                    ].
std066161 : Student [ yearEnrolled->2006,
                    inProgram->cmpeUndergraduateProgram,
                    semesterEnrolled->fall,
                    nationality-> turkish                 ].
```

C3: Instances of the Curriculum (syllabus and curriculum are used interchangeably)

```
cmpeUndergraduateSyllabus: Syllabus [
    forProgram -> cmpeUndergraduateProgram,
    syllabusEntry -> { SE_25711[whichYear->freshman,
                       whichSemester->fall,
                       referenceCode->"25711",
                       course->cmpe101,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                      SE_25712[whichYear->freshman,
                       whichSemester->fall,
                       referenceCode->"25712",
                       course->math163,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                      SE_25713[whichYear->freshman,
                       whichSemester->fall,
                       referenceCode->"25713",
                       course->engl191,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                      SE_25714[whichYear->freshman,
                       whichSemester->fall,
                       referenceCode->"25714",
                       course->math151,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                       SE_25715[whichYear->freshman,
                       whichSemester->fall,
                       referenceCode->"25715",
                       course->phys101,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                      SE_25721[whichYear->freshman,
                       whichSemester->spring,
                       referenceCode->"25721",
                       course->cmpe100,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                      SE_25722[whichYear->freshman,
                       whichSemester->spring,
                       referenceCode->"25722",
                       course->cmpe112,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                      SE_25723[whichYear->freshman,
                       whichSemester->spring,
                       referenceCode->"25723",
                       course->engl192,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                      SE_25724[whichYear->freshman,
                       whichSemester->spring,
                       referenceCode->"25724",
                       course->math152,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                      SE_25725[whichYear->freshman,
                       whichSemester->spring,
                       referenceCode->"25725",
                       course->phys102,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                      SE_25726[whichYear->freshman,
                       whichSemester->spring,
                       referenceCode->"25726",
                       course->{tusl181,hist280},
                       courseType->
                      TurkishOrHistoryCourse]:TurkishOrHistorySyllabusEntry,
                       SE_25731[whichYear->sophomore,
                       whichSemester->fall,
                       referenceCode->"25731",
                       course->cmpe223,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                       SE_25732[whichYear->sophomore,
                       whichSemester->fall,
                       referenceCode->"25732",
                       course->cmpe231,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                       SE_25733[whichYear->sophomore,
                       whichSemester->fall,
                       referenceCode->"25733",
                       course->cmpe211,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                       SE_25734[whichYear->sophomore,
                       whichSemester->fall,
                       referenceCode->"25734",
                       course->engl201,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                       SE_25735[whichYear->sophomore,
                       whichSemester->fall,
                       referenceCode->"25735",
                       course->math241,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                       SE_25741[whichYear->sophomore,
                       whichSemester->spring,
                       referenceCode->"25741",
                       course->cmpe224,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                       SE_25742[whichYear->sophomore,
                       whichSemester->spring,
                       referenceCode->"25742",
                       course->cmpe226,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                       SE_25743[whichYear->sophomore,
                       whichSemester->spring,
                       referenceCode->"25743",
                       course->cmpe242,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                       SE_25744[whichYear->sophomore,
                       whichSemester->spring,
                       referenceCode->"25744",
                       course->math373,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                       SE_25745[whichYear->sophomore,
                       whichSemester->spring,
                       referenceCode->"25745",
                       courseType->
                       CmpeScienceCourse]:RestrictedElectiveScienceSyllabusEntry,
                       SE_25751[whichYear->junior,
                       whichSemester->fall,
                       referenceCode->"25751",
                       course->cmpe323,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                       SE_25752[whichYear->junior,
                       whichSemester->fall,
                       referenceCode->"25752",
                       course->cmpe343,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                       SE_25753[whichYear->junior,
                       whichSemester->fall,
                       referenceCode->"25753",
                       course->cmpe371,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                       SE_25754[whichYear->junior,
                       whichSemester->fall,
                       referenceCode->"25754",
                       course->cmpe321,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                       SE_25755[whichYear->junior,
                       whichSemester->fall,
                       referenceCode->"25755",
                       course->math322,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                       SE_25761[whichYear->junior,
                       whichSemester->spring,
                       referenceCode->"25761",
                       course->cmpe324,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                       SE_25762[whichYear->junior,
                       whichSemester->spring,
                       referenceCode->"25762",
                       course->cmpe344,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                       SE_25763[whichYear->junior,
                       whichSemester->spring,
                       referenceCode->"25763",
                       course->cmpe354,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                       SE_25764[whichYear->junior,
                       whichSemester->spring,
                       referenceCode->"25764",
                       course->cmpe318,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                       SE_25765[whichYear->junior,
                       whichSemester->spring,
                       referenceCode->"25765",
                       courseType->
                       CmpeUniversityElective]:UniversityElectiveSyllabusEntry,
                       SE_25771[whichYear->senior,
                       whichSemester->fall,
                       referenceCode->"25771",
                       course->cmpe400,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                       SE_25772[whichYear->senior,
                       whichSemester->fall,
                       referenceCode->"25772",
                       courseType-> CmpeAreaElective]:AreaElectiveSyllabusEntry,
                       SE_25773[whichYear->senior,
                       whichSemester->fall,
                       referenceCode->"25773",
                       courseType-> CmpeAreaElective]:AreaElectiveSyllabusEntry,
                       SE_25774[whichYear->senior,
                       whichSemester->fall,
                       referenceCode->"25774",
                       courseType-> CmpeAreaElective]:AreaElectiveSyllabusEntry,
                       SE_25775[whichYear->senior,
                       whichSemester->fall,
                       referenceCode->"25775",
                       course->cmpe471,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                       SE_25776[whichYear->senior,
                       whichSemester->fall,
                       referenceCode->"25776",
                       course->cmpe405,
                       courseType-> NormalCourse]:NormalSyllabusEntry,
                       SE_25777[whichYear->senior,
                       whichSemester->fall,
                       referenceCode->"25777",
                       courseType->CmpeEthicsCourse]:EthicsElectiveSyllabusEntry,
                       SE_25781[whichYear->senior,
                       whichSemester->spring,
                       referenceCode->"25781",
                       courseType-> CmpeAreaElective]:AreaElectiveSyllabusEntry,
                       SE_25782[whichYear->senior,
                       whichSemester->spring,
                       referenceCode->"25782",
                       courseType-> CmpeAreaElective]:AreaElectiveSyllabusEntry,
                       SE_25783[whichYear->senior,
                       whichSemester->spring,
                       referenceCode->"25783",
                       courseType->
```

```
                        CmpeUniversityElective]:UniversityElectiveSyllabusEntry,
                        SE_25784[whichYear->senior,
                        whichSemester->spring,
                        referenceCode->"25784",
                        courseType->
                        CmpeFinanceCourse]:RestrictedElectiveFinanceSyllabusEntry,
                        SE_25785[whichYear->senior,
                        whichSemester->spring,
                        referenceCode->"25785",
                        course->cmpe406,
                        courseType-> NormalCourse]:NormalSyllabusEntry
                    }].
SE_25772[course -> ?C]:- ?C:CmpeAreaElective.
SE_25773[course -> ?C]:- ?C:CmpeAreaElective.
SE_25774[course -> ?C]:- ?C:CmpeAreaElective.
SE_25781[course -> ?C]:- ?C:CmpeAreaElective.
SE_25782[course -> ?C]:- ?C:CmpeAreaElective.
SE_25745: RestricredElectiveScienceSyllabusEntry.
SE_25745[course->?C]:- ?C:CmpeScienceCourse.
SE_25784: RestricredElectiveFinanceSyllabusEntry.
SE_25784[course->?C]:- ?C:CmpeFinanceCourse.
SE_25765: UniversityElectiveSyllabusEntry.
SE_25765[course -> ?C]:- ?C:CmpeUniversityElective.
SE_25783: UniversityElectiveSyllabusEntry.
SE_25783[course -> ?C]:- ?C:CmpeUniversityElective.
SE_25777: EthicsElectiveSyllabusEntry.
SE_25777[course -> ?C]:- ?C:CmpeEthicsCourse.
SE_25726: TurkishOrHistorySyllabusEntry.
```

C4: Other Related Instances

A:Grade.

A_MINUS:Grade.

B_PLUS:Grade.

B:Grade.

B_MINUS:Grade.

C_PLUS:Grade.

C:Grade.

C_MINUS:Grade.

D_PLUS:Grade.

D:Grade.

D_MINUS:Grade.

F:Grade.

W:Grade.

NG:Grade.

I:Grade.

S:Grade.

U:Grade.

A:RealGrade.

A_MINUS:RealGrade.

B_PLUS:RealGrade.

B:RealGrade.

B_MINUS:RealGrade.

C_PLUS:RealGrade.

C:RealGrade.

C_MINUS:RealGrade.

D_PLUS:RealGrade.

D:RealGrade.

D_MINUS:RealGrade.

```
F:RealGrade.
NG:RealGrade.
W:FailingGrade.
F:FailingGrade.
NG:FailingGrade.
U:FailingGrade.
D_MINUS:FailingGrade.
A:PassingGrade.
A_MINUS:PassingGrade.
B_PLUS:PassingGrade.
B:PassingGrade.
B_MINUS:PassingGrade.
C_PLUS:PassingGrade.
C:PassingGrade.
C_MINUS:PassingGrade.
D_PLUS:PassingGrade.
D:PassingGrade.
S:PassingGrade.
registered: RegistrationStatus.
leaveOfAbsence: RegistrationStatus.
graduated: RegistrationStatus.
unsatisfactory: Status.
onProbation: Status.
satisfactory: Status.
dismissed: Status.
turkish: Nationality.
turkish_cypriot: Nationality.
nigerian: Nationality.
iranian: Nationality.
```

```
turkish: TurkishOrForeign.
foreign: TurkishOrForeign.
male: Gender.
female: Gender.
fall:Semester.
spring:Semester.
summer:Semester.
freshman:YearType.
sophomore: YearType.
junior: YearType.
senior:YearType.
monday:Day.
tuesday:Day.
wednesday:Day.
thursday:Day.
friday:Day.
turkish: Language.
english: Language.
before(monday,wednesday):-\true.
before(monday,thursday):-\true.
before(tuesday,thursday):-\true.
before(tuesday,friday):-\true.
before(wednesday,friday):-\true.
before(wednesday,thursday):-\true.
before(monday,friday):-\true.
before(monday,tuesday):-\true.
before(tuesday,wednesday):-\true.
before(thursday,friday):-\true.
```

```
emu: University
      [locatedAt -> emu_address].
engineering_faculty: Faculty
       [facultyName -> "Faculty of Engineering",
        atUniversity -> emu].
cmpe_building: Building.
ieng_building: Building.
cmpe_department: Department[
         deptName-> "Computer Engineering",
         inFaculty-> engineering_faculty].
period(1):-\true.
period(2):-\true.
period(3):-\true.
period(4):-\true.
period(5):-\true.
period(6):-\true.
period(7):-\true.
period(8):-\true.
doublePeriod(1,2):-\true.
doublePeriod(3,4):-\true.
doublePeriod(5,6):-\true.
doublePeriod(7,8):-\true.
duration(1).
duration(2).
cmpeUndergraduateProgram: EnglishUndergraduateProgram.
blgmUndergraduateProgram: TurkishUndergraduateProgram.
cmseUndergraduateProgram: EnglishUndergraduateProgram.
```

```
cmpeUndergraduateProgram[ programID -> "025",
                          belongsTo -> cmpe_department,
                          syllabus -> cmpeUndergraduateSyllabus
                        ].
```

C5: Instances of the CourseOpening

```
\#:CourseOpening[
    groupNo ->1,
    ofCourse -> cmse321,
    year -> 2014,
    semester -> fall,
    teachingTimes ->\#[
            room -> cmpe131,
            day -> thursday,
            period -> 1,
            duration -> 1].
    teachingTimes ->\#[
            room -> cmpe131,
            day -> thursday,
            period -> 2,
            duration -> 1].
    teachingTimes ->\#[
            room -> cmpe128,
            day -> friday,
            period -> 5,
            duration -> 1].
    teachingTimes ->\#[
            room -> cmpe128,
            day -> friday,
            period -> 6,
            duration -> 1].
].
```

```
\#:CourseOpening[
    groupNo ->1,
    ofCourse -> cmpe223,
    year -> 2014,
    semester -> fall,
    teachingTimes ->\#[
            room -> cmpe025,
            day -> monday,
            period -> 5,
            duration -> 1].
    teachingTimes ->\#[
            room -> cmpe025,
            day -> monday,
            period -> 6,
            duration -> 1].
    teachingTimes ->\#[
            room -> cmpe033,
            day -> tuesday,
            period -> 5,
            duration -> 1].
    teachingTimes ->\#[
            room -> cmpe033,
            day -> tuesday,
            period -> 6,
            duration -> 1].
].
```

```
\#:CourseOpening[
    groupNo ->2,
    ofCourse -> cmpe423,
    year -> 2014,
    semester -> fall,
    teachingTimes ->\#[
            room -> cmpe128,
            day -> monday,
            period -> 3,
            duration -> 1].
    teachingTimes ->\#[
            room -> cmpe128,
            day -> monday,
            period -> 4,
            duration -> 1].
    teachingTimes ->\#[
            room -> cmpe026,
            day -> tuesday,
            period -> 3,
            duration -> 1].
    teachingTimes ->\#[
            room -> cmpe026,
            day -> tuesday,
            period -> 4,
            duration -> 1].
].

\#:CourseOpening[
    groupNo ->1,
    ofCourse -> math373,
    year -> 2014,
    semester -> fall,
    teachingTimes ->\#[
            room -> cmpe129,
            day -> tuesday,
            period -> 3,
            duration -> 1].
    teachingTimes ->\#[
            room -> cmpe129,
            day -> tuesday,
            period -> 4,
            duration -> 1].
    teachingTimes ->\#[
            room -> cmpe026,
            day -> friday,
            period -> 8,
            duration -> 1].
].
\#:CourseOpening[
    groupNo ->1,
    ofCourse -> cmpe100,
    year -> 2014,
    semester -> fall,
    teachingTimes ->\#[
            room -> cmpe025,
            day -> tuesday,
            period -> 7,
            duration -> 1].
    teachingTimes ->\#[
          room -> cmpe025,
          day -> tuesday,
          period -> 8,
          duration -> 1].
].
\#:CourseOpening[
   groupNo ->1,
   ofCourse -> ieng450,
   year -> 2014,
   semester -> fall,
   teachingTimes ->\#[
          room -> cmpe131,
          day -> monday,
          period -> 5,
          duration -> 1].
   teachingTimes ->\#[
          room -> cmpe131,
          day -> monday,
          period -> 6,
          duration -> 1].
   teachingTimes ->\#[
          room -> cmpe129,
          day -> tuesday,
          period -> 7,
          duration -> 1].
].
\#:CourseOpening[
   groupNo ->1,
   ofCourse -> cmse326,
   year -> 2014,
   semester -> fall,
   teachingTimes ->\#[
          room -> cmpe131,
          day -> wednesday,
          period -> 5,
          duration -> 1].
   teachingTimes ->\#[
          room -> cmpe131,
          day -> wednesday,
          period -> 6,
          duration -> 1].
   teachingTimes ->\#[
          room -> cmpe129,
          day -> thursday,
          period -> 7,
          duration -> 1].
   teachingTimes ->\#[
          room -> cmpe129,
          day -> thursday,
          period -> 8,
          duration -> 1].
].
\#:CourseOpening[
   groupNo ->1,
   ofCourse -> math152,
   year -> 2014,
   semester -> fall,
   teachingTimes ->\#[
          room -> cmpe129,
          day -> thursday,
          period -> 1,
          duration -> 1].
   teachingTimes ->\#[
          room -> cmpe129,
          day -> thursday,
          period -> 2,
          duration -> 1].
   teachingTimes ->\#[
          room -> cmpe128,
          day -> friday,
          period -> 1,
          duration -> 1].
   teachingTimes ->\#[
          room -> cmpe128,
          day -> friday,
          period -> 2,
          duration -> 1].
].
\#:CourseOpening[
   groupNo ->3,
   ofCourse -> cmpe423,
   year -> 2014,
   semester -> fall,
   teachingTimes ->\#[
          room -> cmpe128,
          day -> monday,
          period -> 1,
          duration -> 1].
      teachingTimes ->\#[
          room -> cmpe128,
          day -> monday,
          period -> 2,
          duration -> 1].
      teachingTimes ->\#[
          room -> cmpe026,
          day -> tuesday,
          period -> 1,
          duration -> 1].
      teachingTimes ->\#[
          room -> cmpe026,
          day -> tuesday,
          period -> 2,
          duration -> 1].
].
```