GRP_5: Schedule Sensei

Corey McCann (21CCM12)

Mattias Khan (22MHK)

Hayden Jenkins (21HJDM)

Vasilije Petrovic (21VMP8)

Course Modelling Project

CISC/CMPE 204

Logic for Computing Science

December 4, 2023

Abstract

In the realm of academic course scheduling, our project introduces an innovative logical model designed to assist students in navigating the complexities of curriculum planning. This tool is structured around a series of predefined test cases, each representing typical scheduling scenarios students might encounter. These cases are meticulously crafted to showcase the model's proficiency in handling a variety of scheduling challenges.

The model operates by evaluating crucial factors such as program-specific requirements, course prerequisites, and potential time-slot conflicts. Leveraging the principles of natural deduction, it assesses the feasibility of each test schedule, ensuring that it aligns with academic standards and institutional constraints.

Our approach offers students a practical and insightful tool for academic planning. By interacting with these predefined test cases, students gain valuable foresight into how their course choices interact within the constraints of their academic program. This not only aids in informed decision-making but also simplifies the often complex process of course selection. In essence, our project serves as a bridge between theoretical course planning and real-world academic scheduling, providing a user-friendly platform for students to optimize their educational journey

Propositions

StudentEnrolledCourse_x_y: True if student x is enrolled in course y

StudentEnrolledCourseTerm_x_y_z: True if student x is enrolled in course y in term z

StudentEnrolledCourseSection_x_y_z: True if student x is enrolled in course y in section z

CourseTermSectionTimeConflict_x_y: True if there is a time conflict between two courses x and y (courses are indexed by section and term as well)

CourseTermSectionAvailableCapacity_x_y_z: True if there are seats available for course x in term y in section z

CheckCourseExclusionsExists_x_y_z: True if a course x (excluded_course) that is an exclusion for another course y (course) is present in a student z schedule

CourseExclusionRequirement_x_y: True if the exclusion requirement is satisfied for course x and student y

CheckCoursePrerequisitesExists_x_y_z: True if a prerequisite course x (required_course) for another course y (course) has been completed by a student z

CoursePrerequisiteRequirement_x_y: True if the prerequisite requirement is satisfied for course x and student y

CheckCourseCorequisitesExists_x_y_z: True if a corequisite course x (required_course) for another course y (course) is being taken concurrently by a student z

CourseCorequisiteRequirement_x_y: True if the corequisite requirement is satisfied for a course x and a student y

StudentCourseRequiredTerm_x_y_z: True if a student x must take a course y in term z in order to satisfy some prerequisite(s)

Friendship_x_y: True if student x and y are friends

Constraints

Enrollment Rules

Single Term Enrollment: For every student and course, if a student is taking a course they must be taking the course in one of the terms.

Single Term: For every student and course, they can be enrolled in the course during only one term ex: one of "Fall", "Winter", "Summer" depending on a courses offering

Single Section: For every student and course, they can be enrolled in exactly one section of a course

TermEnrollmentConsistencyCheck: For every student and course, if they are enrolled in a course in a specific term they must be taking the course during that term.

At Most 10: A student can take at most 10 courses, and wishes to take the most possible courses up to 10

Enrollment Restrictions

Time Conflict: For every student and every course if any sections of a course have a time conflict, both of the sections cannot be taken.

Section Enrollment Capacity: A Student can only enroll in a section if there is capacity

Enroll Until Max Capacity: Enroll only as many Students in a Section as there is room

Enrollment Requirements

Course Exclusion Rule: For every student and every course in a students wishlist, if a course that a student wishes to take has an exclusion rule in its requirements, then no course in the students course history or in the courses they wish to take should contain an excluded course. If an exclusion is present then the propositon CourseExclusionExists will be true

Prerequisite Constraint: For every student and every course in a students wishlist, if a course that a student wishes to take has a prerequisite rule in its requirements, then a prerequisite course must be in the students course history.

Course Corequisite Constraint: For every student and every course in a students wishlist, if a course that a student wishes to take has a corequisite rule in its requirements, then a coerequisite course must be in the students course history

Course Requirements Check: For every student and every course in a students wishlist, if a courses exclusions, prerequisites, corequisistes and program requirements are all satisfied then the student can enroll in the course.

Friendship Constraints

Friendship Consistency Check: If students are friends, then they must have a friendship

Restrictions Between Friends: If students are friends and wish to be enrolled in the same course, they may. If and only if there are no prior restrictions that affect them both.

Model Exploration

Original Plan

The project's initial goal was ambitious: allowing students to input their desired course schedules into our model for feasibility assessment. Rooted in complex data manipulation and constraint application, this plan aimed to address the intricacies of academic scheduling.

Evolution of the Project

Post-proposal, a suggestion to create schedules for multiple students led us to explore the concept of "friendship" between students. This innovative approach expanded our model to accommodate the dynamics of group scheduling, resulting in the development of new constraints and propositions to facilitate this feature.

Current Status

As the project progressed, the complexity of handling an extensive course dataset proved challenging, limiting our ability to fully implement user-driven schedule inputs. Instead, we pivoted to a more controlled approach:

- **Predefined Test Cases:** We developed specific test cases that users can select from. These cases are carefully designed to demonstrate the capabilities of our model in various realistic scenarios, including the new "friendship" feature among students.
- **Student Wishlist Integration:** Instead of a full schedule input from users, we shifted to a system where students have a "wishlist" of courses, which the model uses as a basis for feasibility testing.
- **Web Application Interface:** To enhance user experience and provide a clearer understanding of our model, we developed a web application. This platform allows users to run selected test cases and visually explore the resulting course schedules for each term.
- **Scaled Scope:** Recognizing the limitations imposed by data complexity, we scaled down the test cases to include only a few students and courses. Despite this reduction, we believe these scenarios effectively illustrate the model's functionality.

Challenges and Solutions

- **Constraint Logic Complexity:** Enforcing the correct logic within the student schedules, especially considering the new "friendship" feature, presented significant challenges. HAYDEN YOU CAN WRITE HERE IF YOU LIKE, OTHERWISE DELETE THIS SECTION
- **Data Management:** Managing and organizing the large course dataset for practical use in the model was a major hurdle. We addressed this by focusing on smaller, more manageable datasets in our test cases.

- **Adaptability and Flexibility:** Throughout the project, our ability to adapt to unforeseen challenges, particularly in constraint logic and data management, was crucial. This adaptability has been a key factor in the project's progression and in reaching a functional model, albeit on a smaller scale than originally envisioned.

First-Order Extension

PREDICATES

StudentProgram(x,y): student x is in program y
EnrolledCourseSection(w,x,y,z): student w is enrolled in course x in term y in section z
EnrolledCourse(x,y): student x is enrolled in course y
Prerequisite(x,y): course x is a prerequisite for course y
CourseTerm(x,y): course x is in term y
MandatoryCourse(x,y): course x is a mandatory course for program y

Types for Individual Objects

Student(x): x is a student
Course(x): x is a course
Term(x): x is a term
Section(x): x is a section
Program(x): x is a program
Time(x): x is a specific course time
assume that we have equality for objects $(x = y)$ or $\neg(x = y)$
We also need to assume that we can index the EnrolledCourse predicate by year for readability and conciseness

CONSTRAINTS

A student can only be enrolled in exactly one section of a course.

$$\begin{aligned}\forall v \forall w \forall x \forall y \forall z.\, ((&\text{Student}(v) \wedge \text{Course}(w) \wedge \text{Term}(x) \\ &\wedge \text{Section}(y) \wedge \text{Section}(z) \wedge \neg(y = z)) \\ \rightarrow\ (&\text{EnrolledCourseSection}(v, w, x, y) \\ &\rightarrow \neg\text{EnrolledCourseSection}(v, w, x, z)))\end{aligned}$$

A student cannot take two courses simultaneously if those courses have a time conflict

$$\begin{aligned}\forall s\forall t\forall u\forall v\forall w\forall x\forall y\forall z.\,(&(\mathrm{Student}(s)\wedge \mathrm{Course}(t)\wedge \mathrm{Course}(u)\wedge \mathrm{Term}(v)\\ &\wedge \mathrm{Section}(w)\wedge \mathrm{Section}(x)\wedge \mathrm{Time}(y)\wedge \mathrm{Time}(z)\\ &\wedge \neg(u=t))\\ \rightarrow\ &((y=z)\rightarrow(\mathrm{EnrolledCourseSection}(s,t,v,w)\\ &\vee \mathrm{EnrolledCourseSection}(s,u,v,x))))\end{aligned}$$

If a student is enrolled in a program, then all mandatory classes for that program must be included in the student's schedule.

$$\begin{aligned}\forall x\forall y\forall z.\,(&(\mathrm{Student}(x)\wedge \mathrm{Course}(y)\wedge \mathrm{Program}(z))\\ \rightarrow\ &((\mathrm{StudentProgram}(x,z)\wedge \mathrm{MandatoryCourse}(y,z))\\ &\rightarrow \mathrm{EnrolledCourse}(x,y)))\end{aligned}$$

If a course C1 is a prerequisite for another course C2, then the student must take course C1 either in a previous year or before course C2 in the same year

$$\begin{aligned}&\forall x\forall y\forall z.\,((\mathrm{Student}(x)\wedge \mathrm{Course}(y)\wedge \mathrm{Course}(z)\wedge \neg(y=z))\\ &\quad\rightarrow (\mathrm{prerequisite}(y,z)\\ &\quad\rightarrow ((\mathrm{EnrolledCourse}(x,y)_{\mathrm{term1}}\wedge \mathrm{EnrolledCourse}(x,z)_{\mathrm{term2}})\\ &\quad\quad\vee (\mathrm{EnrolledCourse}(x,y)_{\mathrm{previous\ year}}\wedge \mathrm{EnrolledCourse}(x,z)_{\mathrm{current\ year}}))))\end{aligned}$$

Jape Proofs

In order to explore how our premises can be used to prove other premises about our theory our model will need to be simplified. This allows us to bring provable sequents about our model into jape. Our domain of discourse will be narrowed down to just a few courses and students depending on the specific proof.

NOTE: the actual jape proofs in the .j file will look different because jape restricts the use of certain characters (predicates used will just be P, Q, R). However, the sequents will still be the exact same.

Proof 1:
E(x,y) = Student x is enrolled in course y
A(x,y) = there are seats available for student x in course y
T(x,y) = there is a time conflict between course x and course y
This proof describes the following:
If there are two classes and one student in our domain of discourse and that student is enrolled in both classes then you must be able to deduce that there are open seats in both classes for the student and there must not be a time conflict between the two classes.

$\exists x.\exists y.\exists z.((E(x,y)\wedge E(x,z))\wedge((E(x,y)\wedge E(x,z)) \rightarrow (A(x,y)\wedge A(x,z)\wedge\neg T(y,z)))) \vdash \exists x.\exists y.\exists z.(A(x,y) \wedge A(x,z) \wedge \neg T(y,z))$

Proof 2:
P(x,y) = student x is in program y
E(x,y) = student x is enrolled in course y
M(x,y) = course x is mandatory for program y
This proof describes the following:
For all students, all programs and all courses. If all courses z in our domain are mandatory for program y, there exists a student x in program y and that student is enrolled in all courses in the domain. Then we should be able to deduce that there exists a student in any course z and that course z must be a mandatory class for program y.

$\forall x.\forall y.\forall z.(P(x,y)\wedge M(z,y)), \exists x.(P(x,y)), \exists x.\forall z.(E(x,z)) \vdash \exists x.\forall y.\forall z.(E(x,z)\wedge M(z,y))$

Proof 3:
PR(x,y) = course x is a prerequisite for course y
D(x,y) = student x has completed course y
E(x,y) = student x is enrolled in course y
This proof describes the following:
If there exists a student z enrolled in course y and we say that if course x is a prerequisite for course y that implies the student has completed course x then we should be able to deduce that the student z is enrolled in course y which means either course x is not a prerequisite for course y or the student has already completed course x.

$\exists x.\exists y.\exists z.(E(z,y) \wedge (PR(x,y) \rightarrow D(z,x))) \vdash \exists x.\exists y.\exists z.(E(z,y) \wedge (\neg PR(x,y) \vee D(z,x)))$