Advaisor: A Rule Based Expert System for Academic Advising

by

Jacob Hell

Advisor

Joshua Eckroth

A senior research paper submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in the Department of Mathematics and Computer Science in the College of Arts and Science at Stetson University DeLand, Florida

Spring Term

2017
Table of Contents

I. Introduction
II. Background
   ● Academic advising
   ● Rule Based Expert Systems
   ● Web Based Expert Systems
III. Related Work
   ● Ajman University of Science and Technology Prototype Advising System
   ● Advising Tool at Sabanci University
   ● AdviseMe
   ● Limitations of these Systems
IV. Stetson University’s Current System
   ● Ellucian DegreeWorks
   ● Course Lookup
   ● Lessons Learned from Current System
V. Implementation
VI. Conclusion
Abstract

Academic advising is an essential process for improving student retention and academic performance. When an advisor and student come together in an academic advising session, the student should leave with their questions answered and insight gained toward their academic future. However, there are some questions that a student might have that an advisor might not be prepared to answer. Such questions might relate to majors and minors in other disciplines with which the advisor is not familiar, or complex constraint satisfaction questions regarding graduation requirements and timelines. We believe that the advisor may be freed of much of the burden of delving into the Stetson Catalog and Degree Audit tools, and that advising time may be better spent interacting with the student than solving such queries. We propose a system that will meet this need by engaging in hypothetical reasoning to produce alternative pathways that meet students’ needs and interests. The system will internally represent facts and constraints from the Stetson Catalog and provide advisors and students with a user interface that supports complex queries about coursework, degree requirements, and elective courses based on student interests.
1 Introduction

Every semester, each student at a given university or college is required to select a schedule based on their major requirements, preferences, and career goals. Academic advisors might be assigned in order to aid students during this process. The advisor, which may be one of the student’s professors or an employee who has experience in academic advising, must guide students to a schedule that satisfies all of their needs. This can be overwhelming for the advisor, as he or she must possess in-depth knowledge of the institution and the intricate details regarding prerequisites and requirements for every class and major offered.

The only tools that a Stetson student and advisor have available to research their academic concerns is Ellucian’s DegreeWorks and the class lookup application found in myStetson. Although DegreeWorks is able to show the degree requirement for every major and myStetson shows which classes are being offered in subsequent and previous semesters, there is no bridge between these two tools. A student or advisor must constantly switch back and forth between the tools, which can be time consuming and limit the number and complexity of “what if” scenarios that can be explored during an advising session.

The goal of our research is to reduce the student’s and advisor’s research load during an advising session. Using soft constraints such as students preferences and hard constraints such as course prerequisites, we propose to build an application that will satisfy most academic advising concerns. The system will support complex queries, such as queries about switching majors, obtaining a double major, computing years to completion, and the pathway to take a certain class.

The rest of the paper is organized as follows. Section 2 gives a background on academic advising, rule based expert systems, and web based expert systems. Section 3 reviews
previous implementations. Section 4 describes Stetson University’s existing advising technologies. Section 5 outlines our proposed implementation, while Section 6 describes our preliminary work. Section 7 offers conclusive remarks.

2 Background

Academic Advising

Academic advising is more than simply “bookkeeping” courses and grades of student advisees. Research has shown that having a mentoring relationship with a faculty member improves retention rate of college students (Drake 2011). This mentor may be a professor, club sponsor, or academic advisor. Advising is not only concerned with choosing a schedule for the upcoming semester. As Burns Crookston said in “A Developmental View of Academic Advising as Teaching,” advisors should also be concerned with “facilitating the student's rational processes, environmental and interpersonal interactions, behavioral awareness, and problem-solving, decision-making, and evaluation skills” (Crookston 1994). In addition, the advisor is expected to resolve any other academic issue and provide the student with accurate information (Szymanska 2011).

Historically, there are two paradigms of academic advising: prescriptive and developmental (Crookston 1994). Prescriptive advising focuses on the advisor “prescribing” a solution for a student. For example, under prescriptive advising, a student will bring to the advisor their preferences and constraints, and the advisor will construct a complete solution. This model of advising is authoritarian, and the student might feel as if he or she does not have any say in what classes he or she can take. If the student did not enjoy the class, he or she may just blame the advisor for prescribing the classes, since the student had little to no involvement in the decision making (Crookston 1994).
The other paradigm is developmental advising. Developmental advising is more focused on building a relationship between the student and advisor while giving the student more power in deciding his or her class schedule. With developmental advising, the advisee and advisor work together to determine who will research the major, though both parties are involved in building the schedule (Crookston 1994). In Stetson University’s case, either the professor or the student determines the classes that are offered using tools provided by the university, such as the class lookup and DegreeWorks. This process less authoritarian, as either role may have more responsibility. For example, if it is the case that a student chooses a schedule he or she does not like, then the student will feel more responsible for his or her academic performance.

Recently, a new kind of advising, known as transformational advising, has been described. A transformational advisor is encouraged to follow four dimensions when advising. (1) *Individual consideration*, in which the advisor acts as an empathic listener and thereby increases a student's willingness to develop. (2) *Intellectual stimulation*, in which the advisor challenges the student’s beliefs and encourages them to think for themselves. (3) *Inspirational motivation*, in which the advisor envisions a future for the student and motivates him or her to excel. Lastly, (4) *Idealized influence*, in which the advisor reveals how to pursue objectives and the student thereby trusts the advisor more (Babuto 2009). Our tool will stress transformational advising because this is the paradigm that Stetson chooses to implement.

The transformational approach encourages advisors spend more time interacting with their students. However, it is possible for an advisor to get stuck on low-level research tasks when advising. Some institutions have integrated technology within their academic advising process. Utilization of technology is not only helpful for the advisor, but advisees actually prefer advising when the low level systemic tasks were handled with software of some kind (Kolamkarian 2014).
Despite how effective integrating technology in academic advising can be, face to face interaction should not be replaced by technology. Gaines showed that advisees respond better to both visual and verbal cues (2014). Also, students have been found to be more skeptical of purely technology-based advising systems (Kolamkarian 2014). On the other hand, students feel more confident that their questions and concerns have been addressed when advising is primarily face-to-face.

**Rule Based Expert Systems**

We plan to build our tool as a rule based expert system. The goal of designing an expert system is to deliver a system that can produce the same output a human expert would for a certain domain. A rule-based system has the following framework: A knowledge base, that contains facts about the problem, a user interface that allows for user input, and a rule interpreter which is what solves the problem. The rules themselves should look something like IF <condition> THEN <action> (Buchanan 1983). If a condition is evaluated as true, the rule is “fired” and the action associate with that rule is performed. The rule could include a function call or a creation of a fact in the knowledge base (Buchanan 1983).

A knowledge base is composed of two different types of knowledge: factual and heuristic. Factual knowledge is widely accepted by scholars of a domain and is acquired by taking in data from reputable sources. Heuristic knowledge is obtained by manipulating the factual knowledge given (Edwards 1993).

There are two main methods of implementing a rule based expert system: forward chaining and backward chaining. Forward chaining is the process of obtaining information from a user in order to make a conclusion (Abraham 2005). For example, a doctor gets all symptoms of patient in order to conclude on a prognosis. Backward chaining on the other hand works the other way. It begins with a conclusion and then finds supporting facts to reach this conclusion.
(Abraham 2005). An example would be relevant for the purpose of this research, having a conclusion that you have in mind (completing the major) and determining what courses a student has to take to graduate.

Another way to represent backward chaining is to utilize the transitive closure relational structure. Transitive closures are graph theory problems which determine whether or not a node is reachable from a different node (Cohen 1997). Taken in the context of backward chaining, one could determine whether or not a conclusion (a node) is reachable by subgoals (other nodes). A visual representation of this thinking can be seen in Figure 1.

![Figure 1: Example of Transitive Closure](image)

**Web Based Expert Systems**

Our goal is to make Advaisor available on the internet. There are several reasons for this. First, internet applications are easily accessible all over the world, so a student or advisor can utilize our product anywhere they please. Second, web browsers provide a similar template for all web applications, so users should require little training to use them. Lastly, internet based expert systems are more portable. It is easier for the maintainers of the web application to update the code on the servers rather than have users download a patch (Grove 2000).
3 Related Work

Ajman University of Science and Technology Prototype Advising System

Figure 1: Output after user has made submission in the application.

Ajman University’s prototype advising system uses an expert system to simulate a competent academic advisor. In this system, a user will enter his or her ID, and then be presented by a list of suggested classes, presented in recommended order. All information about courses and student records is stored in an object oriented database. Course information includes prerequisites, subject, and course components (such as lab, lecture, etc.), while a student record includes a student’s complete transcript, i.e., classes taken, major, and GPA for each class.

Their expert system includes two types of rules, academic rules and student preference rules. The academic rules, such as course prerequisites, are derived from the course catalog at Ajman University. The student preference rules are derived from student input regarding preferences (such as programming, mathematics, science, etc.).

The researchers at this university utilized the Kappa-PC ES shell to support the rule-based reasoning. There are three main steps in solving an advising query. First, courses
that can be registered by the student are added to an “allowed courses” list (meaning prerequisites are met). Then, the courses are ranked by classes needed for the student's major and preferences. In the last step, the system determines how many classes will be displayed by either the GPA of the student or the amount of classes the student would prefer to take (Ayman 2011).

**Academic Advising Tool at Sabanci University**

![Figure 2: Output after submission of web application.](image)

Sabanci University also developed an expert system that represents an academic advisor. In this case, the system is much simpler compared to Ajman University’s, as it is focused only on showing students which classes they can take based on prerequisites, and the scholarships that they can apply for based on GPA. Unlike Ajman University’s system, Sabanci University’s system is web-based (Engin 2014).

*AdviseMe*
AdviseMe, also a web based academic advising expert system, was developed at the University of the West Indies. The goal of AdviseMe is to provide assistance in general advisory cases for the student. Queries to the system may include: number of credits completed, whether or not a student can graduate currently, list of courses a student can take, and the progress the student has made in their program, along with the progress in other programs if they chose to switch majors.

**Limitations of these Systems**

Given these three expert systems we want to make a point on how they can be improved. Although Ajman University’s advisor has much more features compared to the other two systems, its biggest drawback is that it is not web-based and must be installed locally. Knowing that web-based systems are easier to maintain and require less training than an executable application (Grove 2000), the program developed at Ajman would be hard to integrate into a school’s infrastructure.

AdviseMe and the advising tool developed at Sabanci University on the other hand are web applications. However, the drawback from these systems is that they do not offer what Ajman University has in an expert system. By this I mean that there is no course
recommendation beyond displaying what courses the student has prerequisite for, and would not remove as much administrative tasks that the student and advisor would perform during the academic advising process.

Knowing these drawbacks, we had a more clear idea on what we want to do with our advising system. As said before, an expert system can be limited if it is not web-based, so our application runs on a web server. Also, our goal is to remove as much administrative work from the students and advisors as possible, so we attempted to reduce the amount of inhibitors between the student-advisor relationship.

4 Stetson University’s Current System

Advisors and students at Stetson University have access to three tools for advising purposes.

*Ellucian DegreeWorks*

![Ellucian DegreeWorks Report](image)

Figure 4: Main view of DegreeWorks.
In its current state, DegreeWorks provides some features to help students graduate on time. These include: degree requirements and progress, GPA calculators, and a very simple planning tool. Note that in this planning tool, a student must manually fill out all information themselves, and it has no intelligence behind it, e.g., the system does not have knowledge about whether a class is only offered during Fall semester. This could be problematic because a student’s plan, established using DegreeWorks, may prove to be worthless.

Course Lookup

![Course Lookup](image)

Figure 5: View of class lookup application.

Advisors and students may also refer to the Class Lookup application found in myStetson. The purpose of this application is to allow students to see what classes will be offered along with the information regarding each section (instructor, time, location, etc.).

Queries regarding course offerings and registration may be complicated. For example, consider a student who wishes to build a plan for completing his degree in four years. He could
use DegreeWorks to view the degree requirements then visit Class Lookup to see when each individual class is usually offered (spring, summer, fall, biannually, or triannually). This can become convoluted or impossible if the student is trying to plan for several classes in the future, as typically classes are only listed for the next semester or two semesters.

**Lessons Learned from Current System**

One area that we are focusing on is the lack of knowledge beyond subsequent semester. By this, we mean that none of the three applications will give the user any information besides the closest semester that one can register for. This can create a lot of researching tasks for both the student and the advisor as they will have to delve into previous semesters and take note what courses are offered during certain seasons and whether they can be registered for biannually or every year.

Another point we are focusing on is the integration between the two applications. Despite that they are both important in the academic advising process at Stetson, no information is shared between the two, and one would have to perform research on either application in order to determine what classes to take.

**5 Implementation**

**Data Collection and Cleaning**

The first part of this section will be concerned with the collection of the data needed for the knowledge base. The data was obtained using several python scripts to scrape both the Course Lookup and Course Catalog applications. Using a combination of the requests module and the BeautifulSoup module, we were able to successfully create interchangeable data files that contained what courses are needed for major completion and information about each
individual course such as prerequisites. This data can be seen in Figures 6 and 7.

Figure 6: Courses with information

Figure 7: Major Requirements example

After collecting the information for every course, we then obtained the information for when every course has been offered since 2010. This information was collected in a numerical
fashion, in the format YYYYSS where the YYYY represents the term year and SS represents the semester (15 for fall and 25 for spring). This information can be seen in Figure 8.

```
"201": ["201715", "201615", "201515", "201415", "201315", "201215", "201115", "201015"],
"221": ["201725", "201715", "201625", "201525", "201515", "201425", "201325", "201225", "201115", "201015"],
```

Figure 8: Offer History for CSCI 201 and CSCI 221

From this course offering history, we were able to make a forecast of when each course will be offered for a given period. The periods we used were everyFall (the chance the course is offered every fall), everySpring (the chance the course is offered every spring), otherFall (the chance the course is offered every other fall), and otherSpring (the chance the course is offered every spring). We also found the percentages isFall (the chance a course is offered in the fall), isSpring (the chance the course is offered in the spring), and all (the chance the course is offered both seasons). This data can be seen in Figure 9.
Program Structure

The design of our program follows an object oriented paradigm. There are four main classes that I will discuss in this section: Semester, Prerequisites, SemesterGenerator, and Output.

The semester class is an abstraction for a school semester. The class contains a list of courses to represent the classes that a given student will have in a semester. The semester class also contains a term field which indicates what season this semester is. The term field is
used in conjunction with the usually offered data to determine whether the course is offered in the expert system.

   The Prerequisite class is used to connect course one to course two, indicating that course one is a prerequisite to course two. The way that it is structured is very important in backward chaining in our rule based expert system, which will be discussed later.

   The SemesterGenerator class is what actually starts and adds information to the rule based expert system. By making queries to the database, the class is able generate prerequisites and course information before the expert system is started.

   The output class is where all subgoals created by backward chaining are added to display the results to the user. The output class contains a list of semesters that are used to represent the entire schedule generated for a student.

**Rule Based Expert System**

In our rule based expert system, we utilize backward chaining in order to determine whether or not a course is a prerequisite for a student's major. The prerequisite class, as discussed earlier, is essential in this process. Figure 10 shows the code of how the Prerequisite class is used in drools.

```
query isPrerequisite(String x, String y)
   Prerequisite(x, y;)
   or
   (Prerequisite(z, y; ) and isPrerequisite(x, z;))
end
```

*Figure 10: Backward Chaining Query*

In figure 10, the query presented is called when one wants to determine whether or not x is a prerequisite for y. The first line in the query checks to see if a Prerequisite object exists with the arguments x and y, where x and y are both courses. After the or, it checks to see if there is a prerequisite for z and y, and also x is a prerequisite for z. By doing this, we can recursively
check to see if \( x \) is an eventual prerequisite for \( y \). This can be visualized with the transitive closure presented in Figure 11.

![Transitive Closure For Course Prerequisites](image)

After determining whether or not a possible course can be used for a prerequisite, and determining that a course is available for a given semester based on the term, we want to add it to a generated semester. Once there are no more courses that can be added to a semester, or we have reached our goal, we move on to the next semester term, or we end the knowledge based reasoning.

After a semester can not be manipulated further by adding more courses, we add it to a degree linked list, which contains all the semesters generated. Once our goal has been reached or we can not generate any further semesters, the reasoning ends and the result is displayed for the user.
**User Interface**

There are three main views for the Advaisor program: the home page, the generation page, and the research page. The home and generation page can be seen in figures 12 and 13.

*Figure 12: View of the Main Page for Advisor*

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSCI11I</td>
<td>Introduction to Computing</td>
<td>An introduction to computing for non-computer science majors or those who have no previous programming experience. Introduction to elementary computer theory, algorithmic thinking, terminology and software applications in either a robotics or multimedia context.</td>
</tr>
</tbody>
</table>

*Figure 13: Schedule Generator View*

The home page shows basic information about the student for the user to see. One can see the name of the student and the courses that have been completed. This page is similar to the DegreeWorks application in Figure 4.
The generation page allows for user input to generate semesters for a student based on degree requirements. The backend for this page uses the rule based expert system to determine what courses will meet major prerequisites, and then displays the output. The information is displayed in an ordered manner, with the closest semester courses generated at the top of the table and the subsequent semesters can be seen below that.

The research page displays for the user the courses that are required for the student’s major and the progress that a student has for that course. If a student already has met the requirements for a given course, it will be indicated that the student can register for it once it is available.

6 Conclusion

Academic advising is an essential process for student retention and academic performance. In order to enhance both, we have made the steps to implement an intelligent advisor. Such a system in practice should not only reduce the amount of researching required for both the advisor and student greatly, but may encourage stronger student advisor relationships.
References