Abhra Haldar
6.871 Final Project – Paper

sEElect: An Expert System for Recommending Course VI Classes

Introduction
This paper describes sEElect, a rule-based expert system for recommending classes within MIT's course VI, the department of Electrical Engineering and Computer Science (EECS).

Overview of Problem
Many students come to MIT knowing its reputation for EECS, and are thus eager to major in this field. Yet EECS has over 100 classes and five very separate acknowledged subdivisions; mapping out a path blindly among these classes is an overwhelming task for beginning students.

The primary reason for this difficulty is that freshmen usually have neither a clear idea of what an EECS major entails in the long run, or of their own individual interests within the field. The goal of sEElect is to guide incoming freshmen that have committed to an EECS major to the classes that are most appropriate for the student, given his talents and goals.

The Role of sEElect
sEElect is a rule-based system built on the Joshua Lisp framework, and probes the student with questions directed at revealing the student's preferred learning style and potentially interesting EECS concentrations. These questions require a fair amount of expertise in two areas: 1) student learning habits, and 2) EECS subdisciplines. The questions were constructed specifically with this expertise. Therefore, because of this knowledge, sEElect is capable of inferring things about the student that would not be easily expressed or extracted in a direct question-answer setting.

A sample run of sEElect would involve the student answering a varied set of questions and receiving an ordered list of classes that the system deems 'suitable' for the student. This term is highly subjective, of course, so now we look in further detail at how sEElect makes it choices.

Problem-solving Paradigm

Most Suitable Classes
There are a variety of metrics we can use to measure suitability of a class for a student. For instance, if the student is concerned only with getting good marks, then the primary

reasoning would be what classes the student usually performs well in. On the other hand, if the student is interested only in learning about his preferred field, the metric would determine what that preferred field might is.

In this problem, sEElect recommends classes that potentially have the most positive learning experience for the student. This involves two separate metrics used in tandem:

- What *classroom setting* is most effective learning environment for the student?
- What *EECS field* is most interesting to the student?

The following two sections explore these concepts in detail.

Classroom settings/Learning styles

Different students undoubtedly have different environments in which they learn the most effectively. One might find reading the assigned textbook to be the best way to learn, whereas another might not understand the material unless presented with an opportunity to apply the theory (in a lab setting for instance). Similarly, classes can be broken down by how much they emphasize each of these different styles.

There is myriad number of learning styles possible, and a brief overview of the ones considered by sEElect is presented below:

- *Book-based :* student learns best by reading the assigned material in the textbook
- *Hands-on :* student learns best when forced to apply the material to a real-world problem
- *Lecture-based* : student learns best during lectures
- *Discussion-based :* student learns best in classes where the format is teacher-directed group discussion
- *Independent :* student learns best when given an open-ended problem, w/ limited instructor supervision

The key realization for this problem is that none of these qualities are absolute; rather, they exist in varying degrees in different students. A student might learn best w/ a combination of these strategies. Therefore, there is no easy notion of a set of rules that will deterministically tell us which of these groups a student belongs to. The job of sEElect is to reason which one among these is the *most* effective for a given student, and to pick out the classes that cater to this learning style. In this reasoning process, it assigns *certainty* values to each style, reflecting the strength of the belief that the student learns best with that style.

For an informal example (a formal one is given later), assume that sEElect reasons through its questions that a student is a book-based learner. It would then filter out everything but the classes that, for instance, have excellent textbooks.

Note one drawback of the system as it currently stands. We just concluded that all of these styles contribute to the student's optimal learning environment. Yet sEElect is only capable of inferring the one style that features most prominently in the student. Although sorting by style would appear to be a simple solution, the problem lies in combining styles w/ the EECS disciplines (next section). Currently, sEElect simply chooses the classes corresponding to both the highest ranked learning style and discipline. Sorting with two such factors becomes a non-trivial problem with no apparent solution.

Another, classes typically offer complex amalgams of all these styles. Ideally, sEElect would be able to analyze the breakdown of these styles for the student and return the classes whose breakdowns are most similar. For instance, if the student were equally comfortable in the book-based and hands-on learning environments, a more capable program would match courses that have textbooks and labs of "comparable educational value". The ambiguity of this language indicates one of the main problems with this proposed idea. Perhaps with more expertise and trial, these class breakdowns could be accurately sketched.

Matching EECS Subdisciplines

A course could match a student's learning style perfectly, but if the material does not engage the student, the class of limited value. Thus another crucial task for sEElect is the identification of the most interesting subdiscipline of EECS for our student. Once this determination is made, sEElect considers only those classes in this category.

According to MIT's own 'Area' breakdowns there are five acknowledged subcategories of EECS. However, these are too broad to be useful for this program.
The breakdown sEElect uses is presented in Figure 1:

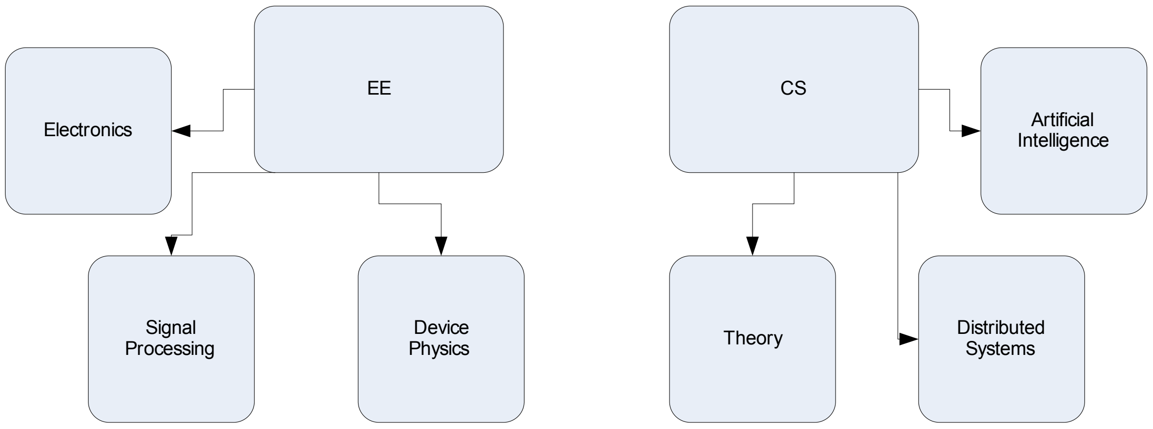


Figure 1: EECS Subcategory Breakdown

Much like with learning styles, students prefer the different subcategories in degrees. sEElect, in reasoning about his most preferred subcategory, ranks each with a *certainty* value. The same problem arises as in learning styles as well. Although preferences for the different subcategories are in degrees, sEElect makes an absolute choice for the most preferred. This is due mainly to the difficulty of sorting the combination of the two sets of ranked factors.

Tradeoffs

One potentially interesting issue not explored by sEElect is the interaction between these two factors: optimal learning style and preferred subdisciplines. As mentioned, the program currently takes the intersection of the sets of classes corresponding to the maximum certainty element in each case. A smarter version of the program would attempt to infer the tradeoffs a student is willing to make between the two factors.

For instance, if a student is so interested in device physics that he is willing to take classes that might not cater to his optimal learning style, this program would adjust its output accordingly (in this case by returning all classes in device physics).

Basic Problem-Solving Strategy

With the conceptual goal of the system now established, we can look at how exactly sEElect chooses classes.

Each of the elements in the two sets of factors has its own boolean-valued predicate in Joshua. One example, of hands-on learning style, is illustrated below:

```
(define-predicate-with-ancillary-info (hands-on value-is-boolean-mixin)
  :possessive-suffix "" :prompt 1 "SHOULDN'T SEE THIS"
      :prompt2 "" :prompt3 "")
```

These predicates <u>all</u> take the boolean value *true*, since they all exist in some degree in each student. Note immediately the awkwardness of representing these factors as predicates (essentially variables), but that all have the same value. This indicates the awkwardness of this representation, and is explored later in the paper. The next step is generate a set of rules that allow us to determine this degree approximately. The natural approach here is backward-chaining, because we have our desired conclusion ([hands-on ?*user* YES]), and want to check whether any other predicates imply this conclusion.

Here is a sample rule for this hands-on predication:

```
(defrule hands-on-sub1 (:backward :certainty .3 :importance 100)
  if [and [tinkers ?user YES]
      (> (get-certainty [tinkers ?user YES]) .2)]
  then [hands-on ?user YES])
```

This rule illustrates the key strategies used in sEElect. First and foremost, each of the predicates in both factors is divided into a set of subgoals. These represent predicates that, if true with some reasonable certainty, lead to a higher certainty for our factor predicate. The subgoals are then further subdivided into more predicates.

In this example, `[tinkers ?user YES]` is the subgoal that, if true, supports the conclusion `[hands-on ?user YES]`. As will be shown in the next section, `[tinkers`

`?user YES]` is broken down into more subgoals. By backward-chaining, we try to show `[tinkers ?user YES]` to conclude `[hands-on ?user YES]`. Likewise, we try to show the subgoals of `[tinkers ?user YES]` to assert that it is a valid conclusion. Ultimately, we reach a base set of predicates with no subgoals. These will then be posed as questions to the user. This chain of reasoning is demonstrated in Figure 2.

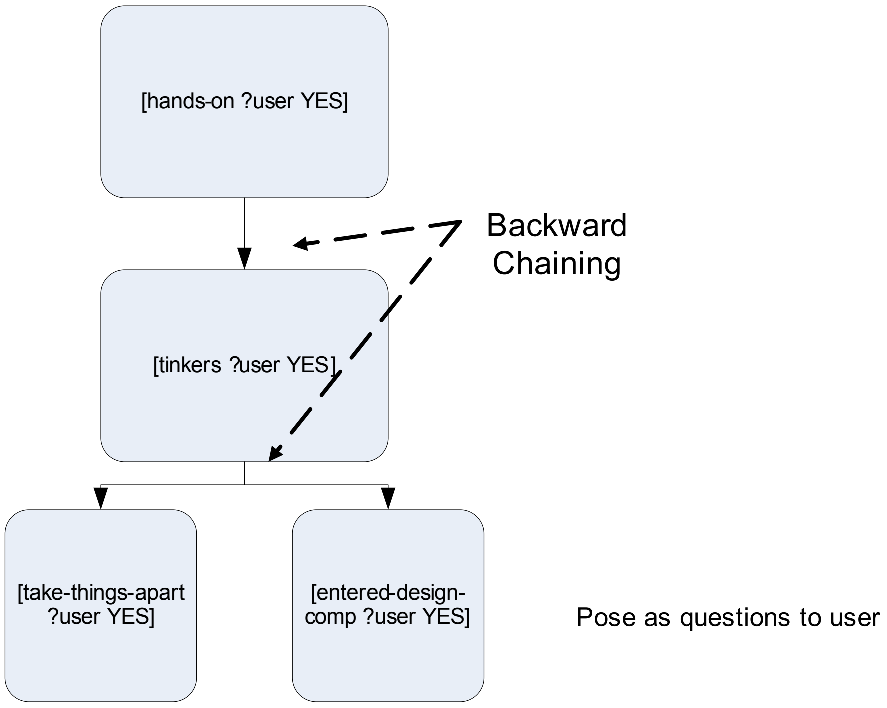

[hands-on ?user YES]
Backward Chaining
[tinkers ?user YES]
[take-things-apart ?user YES]
[entered-design-comp ?user YES]
Pose as questions to user

Figure 2: Backward chaining until you reach predicates with no subgoals

One of the preliminary goals of select is to weigh the two sets of factors by certainty. Fortunately, this is accomplished easily through the `:certainty` keyword in Joshua rules. To illustrate, assume that `[hands-on ?user YES]` has three separate backward chaining rules (or equivalently, three separate subgoals). If each of these carry a certainty of .3 for instance, then using the algebra of certainty, here are the possible resulting certainties for this predication:

Subgoals satisfied	**Resulting Certainty**
3	.51 + .3(1-.51) = .657
2	.3 + .3(1-.3) = .51
1	.3

Note that if none of the goals are satisfied, the predication is not entered into the Joshua database. Therefore, each of the predications that have subgoals have a special *initialization* rule that sets up the predication with some negligible certainty after the other rules have failed to fire. Also notice that a failed subgoal carries no negative certainty, primarily for convenience, since the certainties are highly subjective and inexact in any case.

One final point regarding certainties: in the sample rule provided, note the conditional in the second line of the *if* statement. This type of check allows us to establish certainties for subgoals and only to fire a rule if we find the certainty high enough. This is equivalent to testing how strongly we believe a subgoal is met. Of course, predications with no subgoals have certainty 1, since they are directly answered by the user.

These are the main Joshua mechanisms used in sEElect. Once certainty values are achieved for every single predicate in each factor, we rank each set and choose the predications that have the highest certainty (one learning style and one subdiscipline). We then return the intersection of the set of classes corresponding to these two as the recommended classes for the student.

In summary, here are the steps used in created sEElect's rules database:

1. Set up boolean predicates for each learning style and subdiscipline.
2. Identify conditions (subgoals)that could be used to support any of these predicates. Instantiate them as predicates as well and write an appropriate backward-chaining rule having the subgoal as the antecedent and our boolean predicate as the consequent. If possible, further divide subgoals into more subgoals and write additional backward-chaining rules.
3. Try out different certainties corresponding to different numbers of subgoals satisfied. This is a completely subjective process and there is no exact way to do this. Since all the boolean predicates will be compared on the same scale, we try to maintain some consistency in choosing certainty values.

Domain Knowledge

The two primary forms of domain knowledge in this application come from the two separate factors: learning style and preferred discipline. For the first, the knowledge is "what determines a student's optimal learning environment?."

The second required some outside sources. The HKN Underground Guide to Course VI had comprehensive listings about classes, and I used this resource to classify courses into a particular discipline category. I also used the guide to classify courses into different learning style categories.

Rules Database

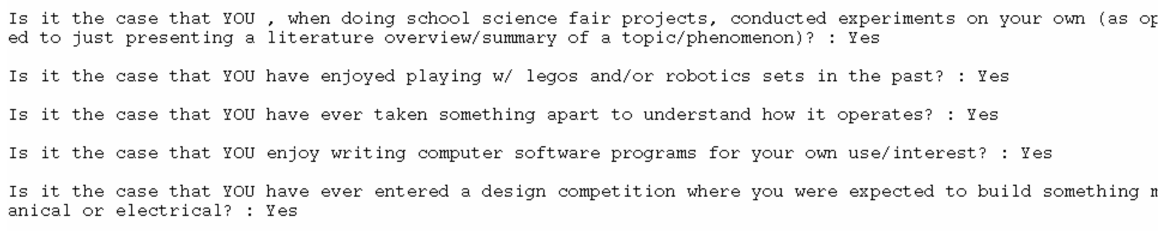
The following section contains a comprehensive breakdown of all the rules used in the system and the reasoning behind them. For each predicate in either factor, I begin by showing the subgoals and informally explaining the reasoning behind them. Next, I give a screenshot of program execution containing only the questions that establish that particular predicate.

Learning Style Rules

As mentioned in the previous section, each predicate was broken down into subgoals, and further if possible. The subgoals for **hands-on learning** are the following:

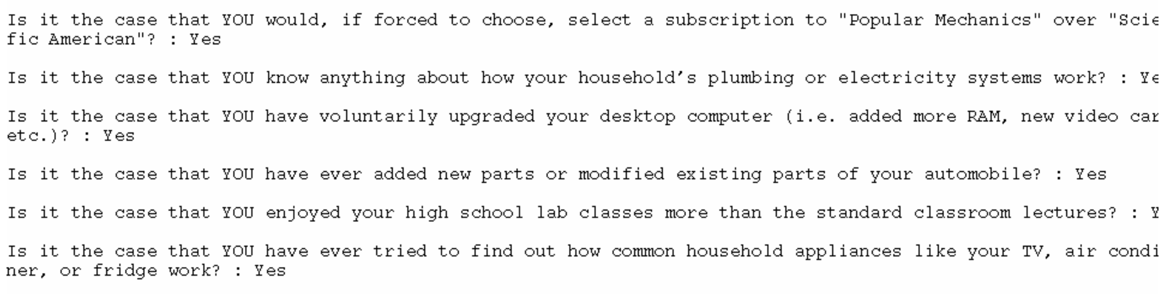
- *Tinkering*: When a person enjoys building things or modifying things in some way, that supports the conclusion that they are hands-on learners.
  <broken down further into>
  - *Taken things apart* – If the student has taken something apart and reassembled it to get a better understanding of how it works, that is evidence that he enjoys tinkering with things.
  - *Enjoys playing w/ Legos and or robotics kits* – Once again, shows that he enjoys building things
  - *Entered design competition w/ something mechanical or electrical* – Enjoys building things
  - *Enjoys writing software* – another form of tinkering, although not with anything tangible or physical; still a predilection for hands-on tasks
  - *Science fair entry an experiment rather than a literature survey or topic overview:* shows he would rather do something concrete rather than just read about it

Questions from actual execution (from subgoals listed above):



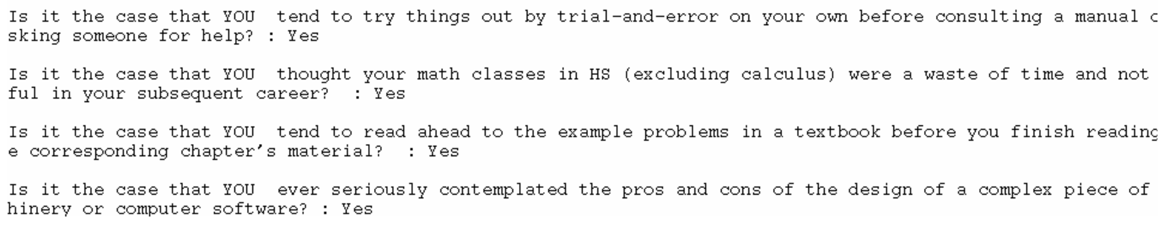
```
Is it the case that YOU , when doing school science fair projects, conducted experiments on your own (as op
ed to just presenting a literature overview/summary of a topic/phenomenon)? : Yes

Is it the case that YOU have enjoyed playing w/ legos and/or robotics sets in the past? : Yes

Is it the case that YOU have ever taken something apart to understand how it operates? : Yes

Is it the case that YOU enjoy writing computer software programs for your own use/interest? : Yes

Is it the case that YOU have ever entered a design competition where you were expected to build something m
anical or electrical? : Yes
```

- *Interest in hands-on tasks:* An enthusiastic hands-on learner would acquaint himself with tasks that require hands-on skills and know how
  - <broken down further into>
  - *Knows how household electricity works* – Shows interest in a hands-on field
  - *Upgraded computer himself* – Shows willingness to engage in hands-on task
  - *Upgraded some part of automobile* – Ditto
  - *Liked high school labs more than classes* – Labs obviously provide more emphasis on hands-on skills than do typical classes
  - *Explored how common everyday appliances work* – Only those who have an interest in hands-on tasks would bother to find out the workings of everyday appliances
  - *Popular Mechanics vs. Scientific American* – The former has a greater emphasis on applications and doing (hands-on skills) while the latter is more theory-oriented



```
Is it the case that YOU would, if forced to choose, select a subscription to "Popular Mechanics" over "Scie
fic American"? : Yes

Is it the case that YOU know anything about how your household's plumbing or electricity systems work? : Ye

Is it the case that YOU have voluntarily upgraded your desktop computer (i.e. added more RAM, new video car
etc.)? : Yes

Is it the case that YOU have ever added new parts or modified existing parts of your automobile? : Yes

Is it the case that YOU enjoyed your high school lab classes more than the standard classroom lectures? : Y

Is it the case that YOU have ever tried to find out how common household appliances like your TV, air condi
ner, or fridge work? : Yes
```

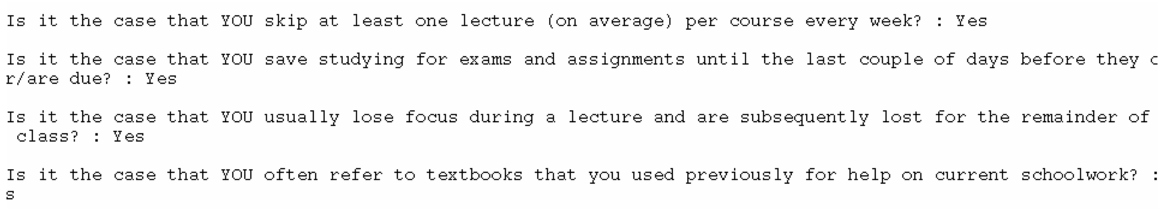
- *Thought processes like a hands-on learner:* Hands-on learners have particular ways of thinking about things that are different from those who rely on book knowledge
    - *Considers HS Math useless* – Those who would think this probably do not see a need for theory that they might not put into practice in the real world. They concern themselves more with things that have concrete applications.
    - *Reads examples before finishing chapter* – Standard example of learning by doing
    - *Design breakdown* – Hands-on learners would try to break things down into smaller pieces and evaluate how they fit together
    - *Try something before consulting another person or the manual* – Hands-on learners would normally try building something without checking the manual, only doing so when running into problems



```
Is it the case that YOU  tend to try things out by trial-and-error on your own before consulting a manual c
sking someone for help? : Yes

Is it the case that YOU  thought your math classes in HS (excluding calculus) were a waste of time and not
ful in your subsequent career?  : Yes

Is it the case that YOU  tend to read ahead to the example problems in a textbook before you finish reading
e corresponding chapter's material?  : Yes

Is it the case that YOU  ever seriously contemplated the pros and cons of the design of a complex piece of
hinery or computer software? : Yes
```

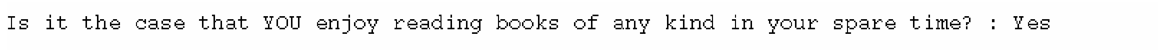
The subgoals for **book-based learning** are now presented with their subgoals below:

- *Classroom behavior conducive to book learning*
    - *Cramming or procrastinating* – Those who procrastinate on homework and exam preparation have little else but the textbook to rely on
    - *Low lecture attendance* – Similarly, the textbook is the only resource for those who miss many lectures
    - *Use old textbooks in current classes* – Indicates that student is reliant on textbook knowledge as his primary resource
    - *Lack of focus during lectures* – Similar to low lecture attendance, if students "zone out" during class, they must rely on the textbook to catch up on the material



```
Is it the case that YOU skip at least one lecture (on average) per course every week? : Yes

Is it the case that YOU save studying for exams and assignments until the last couple of days before they c
r/are due? : Yes

Is it the case that YOU usually lose focus during a lecture and are subsequently lost for the remainder of
 class? : Yes

Is it the case that YOU often refer to textbooks that you used previously for help on current schoolwork? :
s
```

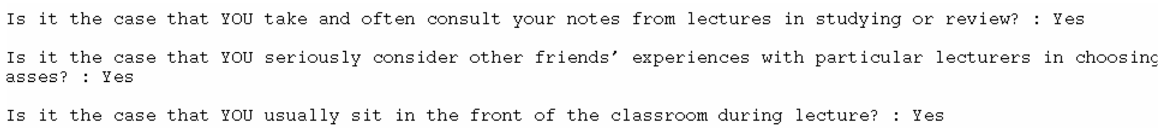
- *Thought processes like a book learner*
    - *High school math useful* – This indicates that the student values knowledge from textbooks that might not necessarily be used in daily life (opposite of a hands-on learner)
    - *Reads chapter material before looking at example problems* – Similarly indicates that student feels compelled to know all the book material before simply applying it
    - *Reads manual before trying to do something* - <Same reason as last>
    - *Enjoys reading books* – self-explanatory



```
Is it the case that YOU enjoy reading books of any kind in your spare time? : Yes
```

The subgoals for **lecturer-based learning** are now presented:

- *Takes notes and consults them during exam prep:* This would indicate that the quality of the lecturer in terms of, say, blackboard technique, is important to the student.
- *Considers classmates' experiences with their lecturers in choosing classes*: This indicates that the student finds the lecturer to be an important part of the learning experience, since his class choices may hinge on his friends' opinion of the lecturers.
- *Sits near the front of the class:* Indicates that the student finds it important to concentrate on exactly what the lecturer is saying

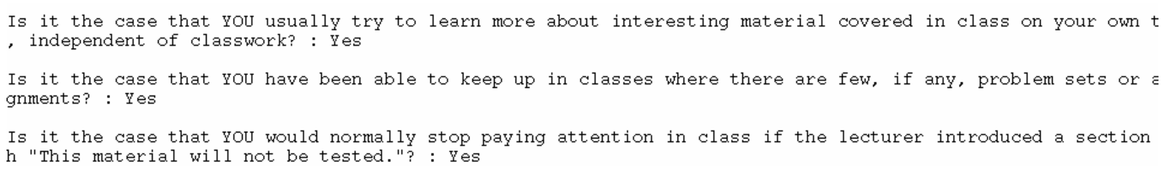


```
Is it the case that YOU take and often consult your notes from lectures in studying or review? : Yes

Is it the case that YOU seriously consider other friends' experiences with particular lecturers in choosinc
asses? : Yes

Is it the case that YOU usually sit in the front of the classroom during lecture? : Yes
```

The subgoals for **independent learning** are now shown:

- *Self-motivated* – Can push himself to learn the material without constant attention or supervision from teaching staff
    - *Can keep up in a class without many assignments:* Shows that he does not need to be forced to keep up with the material through periodic problem sets
    - *Reads and researches interesting material on his own, independent of class:* Once again, demonstrates his ability to motivate himself without being forced

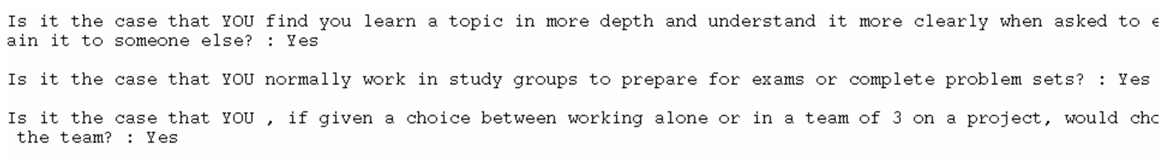
- *Learns more than strictly necessary in classes that interest him:* One test of this is to ask: when the teacher prefaces a section of material with "You will not be tested on this", does the student immediately stop paying attention? The sign of a self-motivated student is one who would continue to focus if the material engaged him.



```
Is it the case that YOU usually try to learn more about interesting material covered in class on your own t
, independent of classwork? : Yes

Is it the case that YOU have been able to keep up in classes where there are few, if any, problem sets or a
gnments? : Yes

Is it the case that YOU would normally stop paying attention in class if the lecturer introduced a section
h "This material will not be tested."? : Yes
```

The subgoals for **discussion-based learning** are now presented:

- *Functions well in a team-*
    a. *Enjoys working in a team* – Self explanatory
    b. *Normally works in study groups for problem sets or exam preparation:* Also indicates that the student works well in a team environment
    c. *Understands better by explaining to others* – If this is the case, then the team environment is beneficial for the student
- *Lecture attendance is good* – if the student does not attend classes regularly, then a class centered on discussion will not be of much use to him



```
Is it the case that YOU find you learn a topic in more depth and understand it more clearly when asked to e
ain it to someone else? : Yes

Is it the case that YOU normally work in study groups to prepare for exams or complete problem sets? : Yes

Is it the case that YOU , if given a choice between working alone or in a team of 3 on a project, would cho
 the team? : Yes
```

The process of establishing most-preferred discipline is slightly different. First off, there are two predicate values representing EE and CS. A series of questions determines which of the two are more suitable for the student (i.e. produce a higher certainty).

Once that is established, we concentrate within the three subdisciplines in that most suitable field. A set of 2 (EE) or 3 (CS) multiple choice questions are used to determine the most preferred discipline by highest certainty.

After both discipline and learning style are established, all that it remains is to take the intersection of the class lists corresponding to each predicate. Instructions on how exactly to run the application on Joshua are given at the end of the paper.

Reflections

What went well

The problem domain was well-suited to modular breakup, as in the two separate factors. For the most part, I could treat the learning style and preferred discipline as completely apart from one another, using separate rules with little or no interaction to establish each factor independently. Moreover, each of the individual predicates within each factor lent itself to subdivision into subgoals and often deeper, so this facilitated reasoning. It also made the rule-based representation with backward chaining particular appropriate.

Also, the fact that Joshua had the concept of certainties built in made it very easy to have a set of predicates and gradually modify the certainties with each rule triggered.

What went badly/Potential Improvements

Perhaps the biggest drawback of the system is its "all or nothing" reasoning process. It chooses only a single predicate from each of the two factors and combines the result of these into the set of recommended classes. As mentioned previously, however, students prefer the disciplines in varying degrees, and similarly exhibit different amounts of each learning style. An improved system would take these into account and somehow rank pairs of predicates from the factors. A related problem is that by taking the direct intersection, one immediately runs the risk of returning no results if there were a small number of classes for any predicate (as there were in this problem).

One significant improvement would be to have rules inferring the student's tradeoff between learning style and discipline. These might also help resolve the problems mentioned in the previous paragraph.

The concept of weights was somewhat troublesome in this application. Certainties were built into Joshua and this was definitely appreciated, but given that the sole determination of recommended classes was done based on certainties, I would have preferred more fine-grained control over the weights, instead of having to reason indirectly about the net certainty from a given number of 'yes' responses.

Another gripe with my representation was the use of multiple choice questions to refine predicate weights. Multiple choice questions are inevitably limiting in scope, and thus might have skewed the overall accuracy of the reasoning. Yet it soon made me realize that in an automated system such as this, it really is not possible to permit open-ended responses. The only workaround is to have a human expert present, ready to interpret the student's open-ended responses into the system.

One realization I had was that perhaps frames would be a conceptually more solid framework to reason about this domain. My code was littered with esoteric initialization procedures to ensure that every predicate had a value of TRUE. Since these predicates different only in their certainties (i.e. degree), I later imagined having a student frame with slots for each of the learning styles and disciplines. At any time, each slot would contain the quantitative strength of our belief that the student prefers that style or discipline.

What I learned

The most important lesson I learned was about the suitability of rule-based systems for handling a problem with distinct subgoals. Here, even though the final solution was not derived directly from any rules (it took into account the certainties of the factor predicates), the rule-based framework fit perfectly well for each of the individual predicates (i.e. hands-on learning). This made me come to the conclusion that whenever a problem involves some combination of different factors, and these factors can be further subdivided, the rule-based approach with backward chaining is excellent.

My initial fear in this problem was optimization. I had initially envisioned return classes that satisfied all degree requirements and prerequisites. Yet as I worked on the project, I realized that that task was certainly a complex system, but with less specialized expertise. The real benefit of this expert system is not in returning a foolproof, guaranteed result, but in capturing expert reasoning to *suggest* possible courses of action. This freed me to think of more and more creative ways of proving a claim, knowing that the student would still examine my system's results carefully and critically.

Running the program

The relevant rules and predicates for the system are located in the Appendix

Once this file is compiled and Joshua is run, type (find-classes) at the Joshua prompt and answer the questions. Once the system prints out your optimal learning style and most preferred subdiscipline, you must enter (intersection <optimal-learning-style>-class-list <discipline>-class-list) at the prompt to get the list of classes.

For example, if the optimal learning style is [hands-on you yes] and the preferred discipline is [electronics you yes], you would enter (intersection hands-on-class-list electronics-class-list) to see the recommended classes.