**The Problem Description**

The Expert System project is meant to design an intelligent system to help students find out what is the right Profession for them, and the system is also supposed to guide students into the suitable Area based on their Profession and interests. In order to design a such system, I need to come up with the inference engine that can pull rules from the knowledge base. The inference engine will be two parts, the first part is the backward chaining methodology that helps user define or examine the Profession for them. The second part is the forward chaining that will give out the suggestion of Area based on the Profession that generated by backward chaining.

For the knowledge base part of the system, it has to be divided into two parts due to the reason that I have both forward chaining and backward chaining in our inference engine. Then the problem would be how to design the decision tree of the backward chaining to cover all possible professions defined by interests. The decision tree for the forward chaining is more straightly forward, but I still need to decide which second interest that could help the user define the Area.

During the development of the Expert System, the biggest problem was how to efficiently store and access the rules of the forward chaining and backward chaining. The most straight forward way to implement IF/THEN rules in the knowledge base is to use the IF statement. Every single rule will be specifically written down in the IF statement, however it will increase the complexity of the inference engine of the Expert System. This method is not only just hard for programmers to add new rules into the knowledge base, but also difficult for the rules to be maintained. In other words, this method has the poor scalability and maintainability. Then problems now are what can I do to make all rules to be well organized and easy to be maintained? How can I properly access those rules to match with the right prompt, and generate the right suggestion?

**The Domain**

The design of the decision tree is mainly the basic idea of the forward chaining and the backward chaining. In order to find out the most efficient design of the backward chaining, I need to look at how the relation ship between Professions and interest. the design should try to avoid of getting too much variables involved in the decision, because it takes extra memories to store those unnecessary variables. That is being said that I need to keep our tree simple, and efficient. For the forward chaining I just need to design the decisions consequently to gain more efficiency.

In order to store and access all rules in an organized way, the design will have to focus on what is the optimal data structure to store all rules properly without using extra memory. The layout of the rules should also be straight forward, and easy to scale. Since the forward chaining and backward chaining takes different design on the decision tree, then the data structure should be used in different ways for the knowledge base of forward chaining and backward chaining for the best interest.

After I figure out the right data structure for rules, then I need to find out an efficient algorithm to help the inference engine to access the rule. Therefore, the inference engine can use less time to find the right rule, and to generate the precise suggestion based on the rule.

**Methodologies**

In the backward chaining methodology, the design should always go from the conclusion of the decision tree. According to the conclusion, the system should prompt possible questions to figure out which is the right rule that applies to the specific case. The purpose of the backward chaining part of my Expert System is supposed to help users find their Professions. In this case, the inference engine should prompt the user with all possible Professions as the first question. Since all Professions are associated with certain rules, then the inference engine should pull the rule that matches Profession that the user chooses. Once the user chooses the profession, the inference engine will go back to the knowledge base of the backward chaining. The algorithm will access and parse the right rule from the data structure, then break those rules into variables. Those variables are formed by the interests and answers. The inference engine will prompt the interest variables to the user, then compare the user’s answer with the answer variable provided by the rule. If the user’s answer does not match the answer from the rule. Then the inference engine will either tell the user that is not the Profession for them, or it will prompt another question to the user depends on the rule. If inference engine tells the user that the Profession is not right, then the backward chaining just proved that the Profession is not the right choice for the user. On the other hand, if answers match, then the backward chaining just successfully helped the user find his or her Profession. All possible conditions are covered in the backward chaining methodology. No matter the users’ answers are “yes” or “no”, my backward chaining will always generate the proper conclusion to the Profession chose by the user.

Forward chaining follows the IF/THEN rules as well. It simulates the forward logic to get the result based on conditions. In other words, the forward chaining doing the rule by the forward order. The forward chaining methodology of my inference engine takes the conclusion of the backward chaining as the input. The inference engine will go back to the knowledge base of the forward chaining, then find the matching rule. The algorithm will again extract all variables from the rule, except for all variables are interests and Areas this time. All the Areas are right next to interests that match to them. Then the inference engine will prompt the user with the interest. If the user’s answer is “no”, then the forward chaining will either prompt the next question, or it will tell the user that there is no Area in this Profession is right for the user. The answer, “yes”, will lead the inference engine to tell the user that Area is the right one. All forward chaining rules covered all Areas in each Profession, and every single possible answer has a conclusion.

**Decision Trees:**

**Backward Chaining:**

decisionMaking

grading

crafting

APA

FNA

EDU

reading

art

writing

BUS

COM

LBA

communicate

HTC

SCI

MED

EGR

science

experiment

nursing

designing

**Forward Chaining:**

Electrical

electronics

EGR

Mechanical

machine

construction

Civil

Petroleum

oil

Chemical

chemistry

Biology

livingBeings

SCI sc

Physics

physicalRules

ComputerScience

computer

chemicalReaction

Chemistry

AnimalScience

livestock

MentalHealth

humanMind

MED sc

Oncologist

cancerTreatment

heart

Cardiologist

musculoskeletalSystem

Orthopedic

GeneralPractitioner

generalCare

Biology

livingBeings

SCI sc

Physics

physicalRules

ComputerScience

computer

chemicalReaction

Chemistry

AnimalScience

livestock

**Rules:** Provide the two sets of rules; one for Backward Chaining and one for Forward Chaining.

**Program Implementation:** Explain in detail the program implementations. Also, explain important features you have implemented.

**Source Code:** Attach a copy of the source code.

**A Copy of the program Run:** Provide a copy of one of you program run.

**Analysis of the program:** Explain what features you have added or modified and why.

**Analysis of the results:** Explain in detail the effect on your results of your program implementations and modified feature.

**Conclusion:** Explain the conclusion and what you learned from the project.

**References:** Add all the references you have consulted in completing the project.