CS4346: Artificial Intelligence

Project #1, Fall 25

Project Points: 100 **Due Dates**: Sept. 17,2025

Project Description

Create an intelligent computer expert system for a mental health clinic to diagnose (*using Backward Chaining*) mental disorders; including Bipolar Disorder, Schizophrenia, Schizo-affective Disorder, Major Depressive Disorder, Panic Disorder with Agoraphobia, Dissociative Identity Disorder, Dysthymia, and Generalized Anxiety Disorder. After diagnosing the diseases, the system should recommend the treatment (*using Forward Chaining*) based on the diagnosis. Perform research using Web or any other source to collect knowledge about the symptoms, diagnoses, as well as treatments of these diseases. The clinic staff will feed the symptoms of the patient.

After collecting knowledge, develop a decision tree for Backward Chaining and a decision tree for the Forward Chaining. Then transform the decision trees into rules. The Backward Chaining decision tree should be big enough to generate a minimum of thirty rules (more rules are fine). The number of Forward Chaining rules will be equal to the number of Disorders you use. The rules should contain variables.

Developing the Source code by Implementation the Methodologies

Implement the expert system program, employing Backward Chaining for diagnosis and Forward Chaining for determining treatments. In developing source code for both methodologies, you must implement the two algorithms (backward chaining algorithm and Forward chaining algorithm) based on the algorithms given in the project description and described in my power point slides, taught in the class. These algorithms are also provided on Canvas. Using any other source code from any other source including web will be treated as plagiarism subject to severe punishment. You can name the two functions as “diagnose Disease” based on Backward chaining and “treat Patient” based on Forward chaining and call them in your “main” function. Knowledge base and Inference Engine parts of each program should be separated. You should bring efficiency in functionality and output using your creativity. Efficiency methods include dynamic memory management, use of objects, and Hashing functions. You can use tools including AI tools, provided your source code is based on the given Backward Chaining and Forward Chaining algorithms and detailed project report is based on the Report Format given in the project description.

Develop a user-friendly interface that receives input data from a staff in a restricted English format, uses keyword matching, and responds in a restricted English format. If you have knowledge of graphics, you could also write a graphical user interface

Project Teams and Their Functionalities

It is a team project. You should form a team of a maximum of three students. You are expected to develop the decision trees, rules, interface, and the inference engines as a team. But the project report must be written individually. **The format of the report is given at the end.**

Steps for Project Report Preparation

After programs are developed, each member of the team will follow the following six steps individually:

1. Run the program with the rules for a minimum of three goals (three Poisons). Each run will be initiated by a user through the Interface. The program will identify the type of the poison and provide a list of treatments for that poison.
2. Print results of at least one run to trace your program.
3. Analyze the results, and the efficiency of the program. It should include how good the results are, how much memory is used, how fast the program was, and **how the changes you made could have affected the efficiency and processing of the program.**
4. Write his/her own detailed REPORT containing the explanation of the problem and the domain, decision tree, rules, methodologies used (Backward and Forward), the program (source code), detailed explanations of the modifications to the source code you implemented, the program run, and the analysis of the program and the analyses of results as defined in item #3 above. The report must contain all the items listed above. Attachments are NOT allowed. Also, identify clearly the contributions of each team member. **The format of the detailed project report is given at the end. You must use this format in writing the detailed project report.**
5. Upload all the information required and explained in the project submission instructions given below.

Warning: EACH STUDENT MUST PERFORM STEPS 1 TO 5 INDIVIDUALLY. DO NOT COPY PROGRAMS FROM ANY SOURCE EXCEPT THE PROGRAMS I HAVE PROVIDED THROUGH TRACS. ALSO, DO NOT COPY REPORT FROM YOUR PARTNER OR ANY OTHER PERSON.

Note: _Each team must demonstrate the execution of the programs in Dr. Ali's office, if asked._

Project Submission Instructions

REPORT SUBMISSION:

- Upload your project report on CANVAS as a file **[ABSOLUTELY NO EMAILS; Minus 50 points for email submissions]** and make sure it can be downloaded easily.
- Also, upload a Readme file containing detailed instructions on how to run your program.

SOURSE CODE and related files SUBMISSION:

Upload source code file as well as other files needed to run your program and verify your results. Source code file naming convention must be: Project1-student ID.cpp; Naming convention of other files will be similar except the file extensions. Your source code must run on a PC computer.

The file uploading instructions are given below:

Login to CANVAS

Access CS 4346 spring 22 course, then follow the following steps:

- Click on [Assignments].
- Select Project #1
- Click on 'Submit Assignment'
- Choose your Files one by one, and UPLOAD them
- Agree to tool's End-User License Agreement
- Click on 'Submit Assignment'

CS 4346- Project #1 Report Format

[Please note that it is just a format. For project requirements, you must read the project description given above].

Each item below must be described in detail. [Rubric points in brackets]

- The Problem Description: Explain the problem for which you have developed the Expert System. [2]
- The Domain: Explain the Domain of the problem. [2]
- Methodologies: Explain in detail the methodologies (Backward and forward chaining) you used. [At least two pages long] [6]
- Decision Trees: Provide the two decision trees, one for Backward Chaining and one for Forward Chaining. [6]
- Rules: Provide two sets of rules, one for Backward Chaining and one for Forward Chaining. [6]
- Source Code Implementation: Explain in detail the source code implementations. Also, explain the important features you have implemented. [At least one page long] [10]
- Source Code: It must be the implementation of the required algorithms given in this project description. [23]

YOU MUST Include a copy of the source code in the report. Attachments will not be accepted [8]

- The Copies of the program Run: Provide copies of your program runs for the required number of patients. [10]
- Analysis of the program: Explain in detail what features you have added or modified and why. [At least one page long] [10]
- Analysis of the results: Explain in detail the effect of your program implementations and modified features on the results. Provide a table with explanations and show the improvements made by your added features and changes. [At least one page long] [10]
- Conclusion: Explain the conclusion and what you learned from the

project. [2]

• References: Add all the references you consulted to complete the project. [2]

• **Contributions: Explain in detail the work each member of the team performed. Team members are expected to contribute to the project equally. [3]**

…………………………………..

Backward chaining Algorithm

Define the Following Functions:

1. **search_con** (*string variable*): with passing the variable as a parameter (variable). This function will find the matching variable in the conclusion list and the corresponding rule number, Ri.

2. **rule_to_clause** (*integer variable*): - This function will convert Rule # Ri to clause number Ci using the following formula. If the rule numbers are sequenced like 1,2,3,4,5, ……), formula is:
   CLAUSE NUMBER (Ci) = 4* (RULE NUMBER (Ri) - 1) + 1
   If the rule numbers are sequenced like 10,20,30,40,50, ……), formula is:
   CLAUSE NUMBER = 4* (RULE NUMBER / 10 - 1) + 1
   It has been assumed that four slots have been assigned for each rule in the Clause Variable list. If a 'number' other than four has been assigned, replace 4 with that number.

3. **update_VL** (*integer variable*): with passing Ci to it. It will all questions and will instantiate all (maximum of 4) variables starting from the location Ci in the variable list. If the variable is not in the variable list, it could be in the then clause of some rule. call Process (variable) to find its value-it will be a recursive call.

4. **validate_Ri** (*integer variable, string conclusion*): with passing, Ri, to it. It will check if the values of the variables in the 'if' clauses of the rule, Ri, are satisfied with the values in the variable list and the derived global variable list. If they do, it will assign the conclusion of the rule to the variable conclusion; otherwise, it will not assign any value, and then it will return.

5. Process (variable)
    - Start a loop. In each cycle of the loop, do the following:
    - **call search_con (variable)** by passing the goal variable as a parameter (variable). It will find the matching variable in the conclusion list and the corresponding rule number, Ri.
    - call **Rule_to_clause (Ri)** by passing Ri as a parameter. This function will convert the rule number, Ri, to clause number, Ci.
    - call **Update_VL (Ci)** with passing Ci to it. It may ask questions and will instantiate all (maximum of 4) variables starting from the location Ci in the variable list. If the variable is not in the variable list, it could be in the then clause of some rule. call Process (variable) to find its value-it will be a recursive call.
    - Initialize conclusion to null and call **Validate_Ri (Ri, Conclusion)** *with passing, Ri, to it*. This function will check if the values of the variables in the 'if' clauses of the rule, Ri, are satisfied with the values in the variable list and the derived global variable list. If they do, it will assign the conclusion of the rule to the **conclusion** variable. Save the value of the conclusion variable in the derived global variable list and **return**. If the values of the variables in the 'if' clauses of the rule, Ri, are not satisfied with the values in the variable list and the derived global variable list, use the

'continue' statement to continue the loop. It will repeat the process with the next entry of the conclusion list.

6. main function

Declaration

(you are allowed to declare variables and lists global)

- Write functions' prototypes
- Declare variables and arrays
- Create a list of rules. It may need some organization. Determine which format you want to use for efficient processing by the **Validate_Ri** function.
- Create a Variable List (could be an array).
- Create a conclusion list (could be an array).
- Create a Clause Variable list (could be an array).
- Create the Derived Global Variable list.

Processing in the main function:

- Identify the Goal variable (the variable whose value needs to be determined)
- call Process (Goal variable)
- After the program is complete, check if the derived global variable list contains the goal variable and is not null. If it does, that is the answer to processing the backward chaining. Print the value of the variable and pass it to the Forward chaining program. If it is null, print a comment that the Goal cannot be determined.
- An example of a derived global variable list is given below:

Variable	Instantiated	Value
Qualify	I	yes
Position	I	Service Engineer

……………………………

Forward chaining Algorithm

Define the Following Functions:

1. **search_cvl** (*double variable*): This function will search for an entry in the clause variable list and, find the entry that matches the argument variable, and return the clause number, Ci, corresponding to the matching entry. Then, first call **update_VL** (*Ci*). Then call **clause_to_rule** (*Ci*):

2. **clause_to_rule** (*integer variable*): - This function will convert Clause number, Ci, to rule number, Ri, using the following formula. After computing the value of Ri, it will call **validate_Ri** (*Ri*):
   If the clause numbers in the clause variable list are sequenced like 1,2,3,4,5, ……), the formula is:
   Rule # = [({Quotient (Clause # / 4)} +1)]
   If the rule numbers are sequenced like 10,20,30,40,50, ……), the formula is:
   Rule # = [({Quotient (Clause # / 4)} +1) * 10]
   It has been assumed that four slots have been assigned for each rule in the Clause Variable list. If a number other than four has been assigned, replace 4 with that number.

3. **update_VL** (*integer variable*)**:** For each variable (maximum of four) starting location Ci, it will check whether the variable is instantiated in the variable list. If not, it will ask the user to provide the variables' values and instantiate them.

4. **validate_Ri** (*integer variable*): It will check if the values of the variable in the 'if' clauses of the rule, Ri, are satisfied with the values in the variable list. If they do, add the rule's conclusion to the 'global' derived conclusion list and the Global Conclusion Variable Queue and return.

5. **process (variable)**

a. Instantiate the value of the variable in the variable list. Call **search_cvl**(*variable*)
b. return

6. FC_main function

Declaration of the FC_main function

- Write functions' prototypes.
- Declare variables and arrays.
- Create a list of rules. It may need some organization. Determine which format you want to use for efficient processing by the **Validate_Ri** function.
- Create a Variable List as an array.
- Create a Clause Variable list as an array.
- Create a Global Conclusion variable queue as an array.

Processing of the main function:

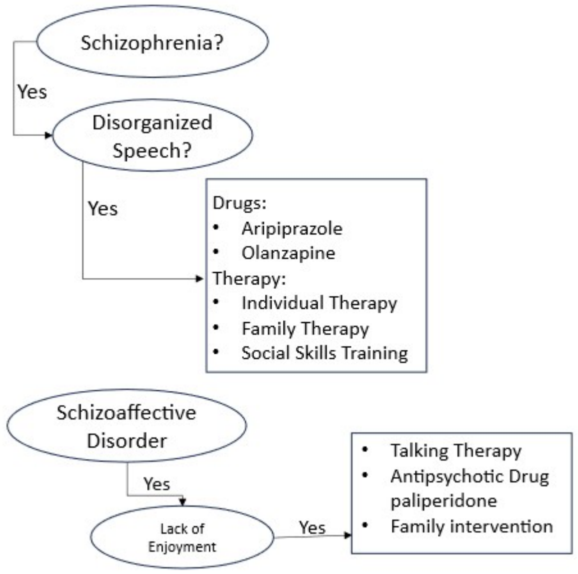
a. Read the value of the given variable (the variable whose value is provided) and **call process (variable)**
b. If the Global Conclusion Variable Queue is not empty, in a loop, call process (variable) for each variable in the Global Conclusion Variable Queue, which occurs in the clause variable list. After the function returns, it will delete the variable from the Global Conclusion Variable Queue.
c. Print all the derived conclusions from the Derived Conclusion List

Example of the Global Derived Conclusion list

Interest = FALL
Stock = Rise

Partial Decision Tree Examples for Backward Chaining and Forward Chaining for Project #1 (mental Diseases) are given in these power point Slides

FORWARD CHAINING

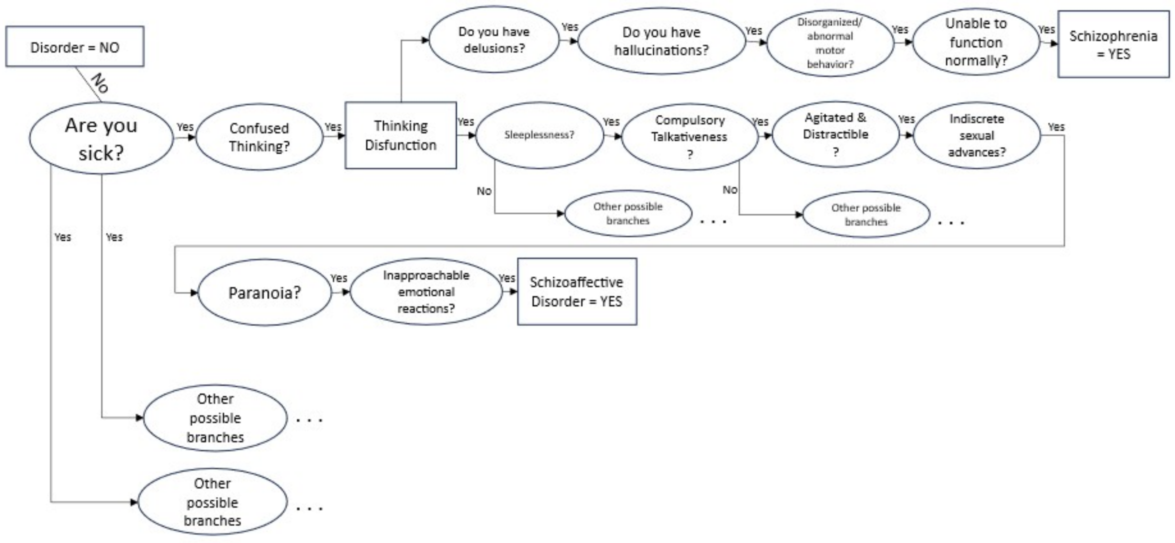


.

.

.

Backward Chaining



Disorder = NO
No
Are you sick?
Yes
Confused Thinking?
Yes
Thinking Disfunction
Do you have delusions?
Yes
Do you have hallucinations?
Yes
Disorganized/ abnormal motor behavior?
Yes
Unable to function normally?
Yes
Schizophrenia = YES
Yes
Sleeplessness?
Yes
Compulsory Talkativeness ?
Yes
Agitated & Distractible ?
Yes
Indiscrete sexual advances?
Yes
No
Other possible branches
. . .
No
Other possible branches
. . .
Paranoia?
Yes
Inapproachable emotional reactions?
Yes
Schizoaffective Disorder = YES
Yes
Yes
Other possible branches
. . .
Other possible branches
. . .