ASSIGNMENT 2 – INTEFERENCE ENGINE

COS30019-INTRODUCTION TO AI

Tran Duy Dao & Vuthy Yi
STUDENT ID | 102848516 & 102871305

Tran Duy Dao and Vuthy Yi
Student ID: 102848516 / 102871305

CONTENTS

INTRODUCTION: 2

WORKLOAD 2

IMPLEMENTED: 2

TEST CASES: 3

ACKNOWLEDGEMENT: 5

REFERENCES: 6

INTRODUCTION:

The theme of our project is showing the application of the interference engine for propositional logic. With the use of Forward Chaining (FC) algorithms, Backward Chaining (BC) algorithms and the Truth tables (TT). We can develop an Interference Engine that can take a Horn-form knowledge base and a query. The Truth table is checking the algorithm and we must make sure that we have provided enough test cases to check the Forward Chaining (FC) and Backward chaining (BC). We did face some issues because Truth table checking can be versatile with all types of knowledge bases while the Forward Chaining (FC) and Backward Chaining (BC) algorithms work with horn-form knowledge bases. The interference engine for this assignment is implemented in Java with the help of Vuthy (student ID: 102871305) and Tran (student ID: 102848516).

WORKLOAD

Tran Duy Dao: 102848516 (50% workload)

- Writes the report
- Set up iEngine.java
- Handles the Forward Chaining.java codes
- Help fixes bugs
- Does research on forward and backward chaining

Vuthy Yi: 102871305 (50% workload)

- Handles the logic of PropositionalLogic.java
- Handles all the codes for BackwardChaining.java

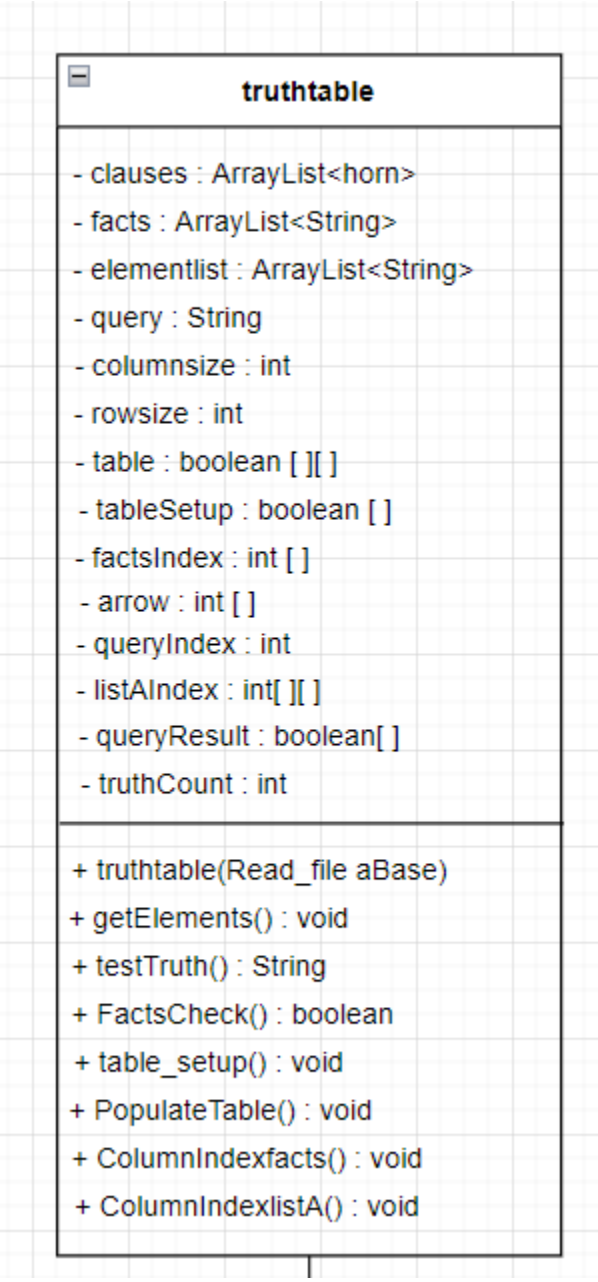
- Handles the Symbol.java

- Debugs and compiles

- Edits the report

CONCEPTUAL DESIGN:

This is the attempt of making the Truth table before we start working on the project:



IMPLEMENTED:

1) THE MAIN FIVE CLASSES:

This program has five classes, the Driver class included namely:

- BackwardChaining.java
- ForwardChaining.java

- iEngine.java
- PropositionalLogic.java
- Symbol.java

2) FEATURES:

In the iEngine.java class the program is started and the arguments (the method and the filename) are passed from the arguments. Depending on the relevant method initials given the appropriate class is called through the *readTell* and *readAsk* functions in the respective classes.

- The method classes (Backward chaining, Forward Chaining, and PropositionalLogic) will receive the contents of the given file and separate the line TELL line containing statements separated by semicolons. The line will be split using the semicolon as regex for splitting.
- Each statement will now be analyzed on how many connectives they contain. The program will be able to check which connectives are contained in each statement. This program handles perfectively the statements with one connective, and the statements with multiple connectives, the program at this point considers the implication and the conditional to process it. The statements with no connectives are inferred as facts.
- The ForwardChaining class in the ForwardChaining will infer from the facts to propositions in the statements. The PropositionalLogic will keep count on the Propositions we keep visiting. The BackwardChaining class will find the rule/statement with the query in its conclusion will continue to prove it recursively.
- The symbol class holds the attributes of a statement, which is the proposition are store in a hashmap (the key being the proposition to the left of the symbol and the value being the proposition to the right of the symbol), the symbols stored as a string, symbol in words also stored in a string as well as facts.

In the method classes, we utilized recursion in checking through the statements to determine which ones are related.

3) BUGS:

There are no evident bugs in the way the program works as far as the correct format of the program is concerned. Which is:

- before the line with the statements add the word TELL
- The statements will be separated with semicolons
- Before the line having the query add the word ASK
- The query will be a proposition

The format is also shown in the test cases

4) MISSING:

We have added the commands to run the file iEnginerun.bat, and will run from the command prompt as shown in the screenshot below

TEST CASES:

These are the tests run on Vuthy's personal computer:

- TELL
- p2=> p3; p3 => p1; c => e; b&e => f; f&g => h; p1=>d; p1&p3 => c; a; b; p2;
- ASK
- d

```
C:\Users\Vuthy\Desktop\Intro to AI\Assignment 2>java iEngine TT test_HornKB.txt
TELL: p2=> p3; p3 => p1; c => e; b&e => f; f&g => h; p1=>d; p1&p3 => c; a; b; p2;
ASK: d
YES: 3

C:\Users\Vuthy\Desktop\Intro to AI\Assignment 2>java iEngine FC test_HornKB.txt
TELL: p2=> p3; p3 => p1; c => e; b&e => f; f&g => h; p1=>d; p1&p3 => c; a; b; p2;
ASK: d
YES: b, a, p2, p3, p1, d

C:\Users\Vuthy\Desktop\Intro to AI\Assignment 2>java iEngine BC test_HornKB.txt
TELL: p2=> p3; p3 => p1; c => e; b&e => f; f&g => h; p1=>d; p1&p3 => c; a; b; p2;
ASK: d
YES: p2, p3, p1, d
```

Other tests:

- TELL
- a => p3; p3 => p1; c => e; a; b; p2;
- ASK
- p1

```
C:\Users\Vuthy\Desktop\Intro to AI\Assignment 2>java iEngine TT test.txt
TELL: a => p3; p3 => p1; c => e;  a; b; p2;
ASK: p1
YES: 2

C:\Users\Vuthy\Desktop\Intro to AI\Assignment 2>java iEngine FC test.txt
TELL: a => p3; p3 => p1; c => e;  a; b; p2;
ASK: p1
YES: p2, b, a, p3, p1

C:\Users\Vuthy\Desktop\Intro to AI\Assignment 2>java iEngine BC test.txt
TELL: a => p3; p3 => p1; c => e;  a; b; p2;
ASK: p1
YES: a, p3, p1
```

- TELL

- b&e => f; f&g => h; p1=>d; p1&p3 => c; a; b; p2;
- ASK
- h

```
C:\Users\Vuthy\Desktop\Intro to AI\Assignment 2>java iEngine TT test.txt
TELL: b&e => f; f&g => h; p1=>d; p1&p3 => c; a; b; p2;
ASK: h
YES: 1

C:\Users\Vuthy\Desktop\Intro to AI\Assignment 2>java iEngine FC test.txt
TELL: b&e => f; f&g => h; p1=>d; p1&p3 => c; a; b; p2;
ASK: h
NO: p2, b, a, h

C:\Users\Vuthy\Desktop\Intro to AI\Assignment 2>java iEngine BC test.txt
TELL: b&e => f; f&g => h; p1=>d; p1&p3 => c; a; b; p2;
ASK: h
YES: f&g, h
```

- TELL
- p2=> p3; p3 => p1; c => e; b&e => f; f&g => h; p1=>d; p1&p3 => c; a; b; p2;
- ASK
- c

```
C:\Users\Vuthy\Desktop\Intro to AI\Assignment 2>java iEngine TT test.txt
TELL: p2=> p3; p3 => p1; c => e; b&e => f; f&g => h; p1=>d; p1&p3 => c; a; b; p2;
ASK: c
YES: 1

C:\Users\Vuthy\Desktop\Intro to AI\Assignment 2>java iEngine FC test.txt
TELL: p2=> p3; p3 => p1; c => e; b&e => f; f&g => h; p1=>d; p1&p3 => c; a; b; p2;
ASK: c
NO: p2, b, a, c

C:\Users\Vuthy\Desktop\Intro to AI\Assignment 2>java iEngine BC test.txt
TELL: p2=> p3; p3 => p1; c => e; b&e => f; f&g => h; p1=>d; p1&p3 => c; a; b; p2;
ASK: c
YES: p1&p3, c
```

- TELL
- a => b;
- ASK
- c;

```
C:\Users\Vuthy\Desktop\Intro to AI\Assignment 2>java iEngine TT test.txt
TELL: a => b;
ASK: c;
NO: 0

C:\Users\Vuthy\Desktop\Intro to AI\Assignment 2>java iEngine FC test.txt
TELL: a => b;
ASK: c;
NO: , c;

C:\Users\Vuthy\Desktop\Intro to AI\Assignment 2>java iEngine BC test.txt
TELL: a => b;
ASK: c;
NO: , c;
```

ACKNOWLEDGEMENT:

(Hauskrecht)

By Milos Hasukrecht.

This document provides us with the foundation of the Knowledge base in Horn form and the Algorithm for the different methods. By understanding the Complexity of inference for knowledge bases in Horn form. The linear implementation of Forward Chaining and proving by Backward Chaining.

We also got to understand that forward chaining is data-driven and suitable for object recognition, routine decisions among others. While Backward Chaining is goal-driven and is

appropriate for problem-solving. This is how we also got to know the limitations of the Horn Form for bug fixing.

(Forward and Backward Chaining)

By Francisco Lacobelli

This is an informative video about Horn form to help me further understand the uses of it in Knowledge Bases conjunctions with the flexibility of being able to rewrite as implications. Forward chaining is data-driven that can open to object recognition and even routine decisions for the algorithms. Also, the complexity of Backward Chaining can be much less than the linear size of Knowledge Bases.

(Javatpoint, 2020)

By javatpoint

This web page helps our team with most of the codebase related issues and the theoretical foundations of an inference engine, with forward chaining and backward chaining as its core with Java as a development language. They also helped a bit with the syntax that was used in our assignment.

Tutor: Online sessions with the tutor have helped us clear some of our group concepts regarding the assignment. Especially with the video explaining the Propositional Logic with a detailed semantic and syntax to help us make it logical and return the correct answer in the tests with Tutorial 7. Tutorial 6 provided us with the knowledge on how the computer thinks, and it is indeed thinking rationally, also gaining a deeper understanding of the Knowledge-based in general for handling the logic.

Lecture Notes- The lecture notes provided in this unit have ample information. Moreover, it is presented in simple terms (or words) which has helped us reinforce our knowledge of the subject.

REFERENCES:

Research references:

(1)

Hauskrecht, M. (n.d.). Knowledge Representation. Retrieved from people.cs.pitt.edu: https://people.cs.pitt.edu/~milos/courses/cs2740/Lectures/class6.pdf, viewed 20 May 2022.

(2)

Javatpoint. (2020). Forward Chaining and backward chaining in AI. Retrieved from javatpoint.com: https://www.javatpoint.com/forward-chaining-and-backward-chaining-in-ai, viewed 20 May 2022.

Tutors and lecture notes references:

(3)

Vo, B 2022, 'Lecture 6. Logical Agents Knowledge Representation', COS30019 Introduction to Artificial Intelligence, Learning materials via Canvas, Swinburne University of Technology, 04 April, viewed 05 May 2022.

(4)

Vo, B 2022, 'Lecture 7. Propositional Logic, COS30019 Introduction to Artificial Intelligence, Learning materials via Canvas, Swinburne University of Technology, 11 April, viewed 12 May 2022.

(5)

Iacobelli, F 2015, *Forward and Backward Chaining*, 16 July, viewed 05 May 2022, <https://www.youtube.com/watch?v=EZJs6w2YFRM>.