

|  |  |
| --- | --- |
| Assignment 2 – Inference Engine | |
|  |  |
| Mihir Bhadauria (103075328)Kingsley Brodie (102147941) Group: **COS30019\_A02\_T022** |  |

Contents

[Assignment 2 – Inference Engine 1](#_Toc135506339)

[Mihir Bhadauria (103075328) 1](#_Toc135506340)

[Kingsley Brodie (102147941) 1](#_Toc135506341)

## Instructions:

The program is developed in python and is stored in a zip file with all the necessary components.

To execute the following program in CMD, you need to enter in this format:

|  |
| --- |
| ..\user> main.exe <method> <filename> |

To break this down

* main.exe runs the program.
* <method> is the type of search you want to do and those are shown below.
* <filename> is the name of the input file which in this case is *“test\_HornKB.txt”*

|  |  |
| --- | --- |
| Truth Table | main.exe TT <filename> |
| Forward Chaining | main.exe FC <filename> |
| Backward Chaining | main.exe BC <filename> |

*Table 1: Command line program execution instruction*

The <method> is not case sensitive however if any issues occur type the <method> shown above in lowercase.

## Features/Bugs/Missing

The main features of the program comprise of:

* Running inference engine with the following methods
  + Truth table
  + Forward Chaining
  + Backward Chaining
* They currently work only with Horn input.

The current program has some limitations that should be addressed:

* **Lack of Input Validation:** The program assumes that the input is always in a specific format as follows:

A picture containing text, font, algebra, screenshot

Description automatically generated

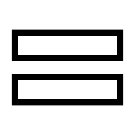
* Solution: To mitigate this issue, we propose two potential solutions:
  + Implement robust input validation methods within the program to ensure the correctness of the input data structure.
  + In addition, or alternatively, we can prepare detailed documentation outlining the required input format, thus guiding the user to provide the data points in the correct sequence.

These improvements will enhance the program's robustness and usability.

## Test Cases

To ensure this program is robust and reliable we will test it in a variety of situations to ensure it works.

1. A black background with white text

   Description automatically generated with low confidenceA picture containing text, font, white, screenshot

   Description automatically generatedSingle literal Knowledge Base with single literal query:
2. A picture containing font, screenshot, text, number

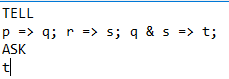
   Description automatically generated![A picture containing black, darkness

   Description automatically generated](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAIcAAACHCAMAAAALObo4AAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAYUExURQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFY3HCoAAAAHdFJOUwAKCxwdYGHSVhpCAAAACXBIWXMAACHVAAAh1QEEnLSdAAAApklEQVR4Xu3VK5LDQBBEQWm9n/vfeEFNgTYxsSYUjkzWqB7rAwAAALjE+bPXuXafPf72+l67z3RMOqabdazrShnSURnSURnSURnSURnSURnSURnSURnSURnSURl60bGNjknHdPcOAPhk/v6kY9IxrY51XSlDOipDOipDOipDOipDOipDOipDOipDOipDOipDLzq20THpmO7ecf7u9bV2AQAAgHc6jn/snVvNteT84gAAAABJRU5ErkJggg==)A black text on a white background

   Description automatically generated with low confidenceSingle clause Knowledge Base with single literal query:
3. Single clause Knowledge Base with conjunction query:
4. Multiple clause Knowledge Base with single literal query:

A picture containing text, font, screenshot

Description automatically generated![A picture containing black, darkness

Description automatically generated](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAIcAAACHCAMAAAALObo4AAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAYUExURQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFY3HCoAAAAHdFJOUwAKCxwdYGHSVhpCAAAACXBIWXMAACHVAAAh1QEEnLSdAAAApklEQVR4Xu3VK5LDQBBEQWm9n/vfeEFNgTYxsSYUjkzWqB7rAwAAALjE+bPXuXafPf72+l67z3RMOqabdazrShnSURnSURnSURnSURnSURnSURnSURnSURnSURl60bGNjknHdPcOAPhk/v6kY9IxrY51XSlDOipDOipDOipDOipDOipDOipDOipDOipDOipDLzq20THpmO7ecf7u9bV2AQAAgHc6jn/snVvNteT84gAAAABJRU5ErkJggg==)

1. Multiple clause Knowledge Base with disjunction query:
2. Knowledge Base with negation, with single literal query:

## Acknowledgement/Resources

<https://www.geeksforgeeks.org/proposition-logic/>

<https://cs50.harvard.edu/ai/2020/notes/1/>

<http://aima.cs.berkeley.edu/python/logic.py>

* Aside from lectures and tutorials this has aided me in developing a greater understanding of propositional logic and its fundamentals.
* These resources have also enabled me to understand how to go about coding this inference engine.

I have used (Mihir) the parser coding style from my assignment #1.

## Research

## Team Summary Report

**Contribution Matrix**

|  |  |
| --- | --- |
| Kinglsey Brodie (102147941) | Mihir Bhadauria (103075328) |
| * Forward Chaining Implementation * Backward Chaining Implementation * Report writing | * Input parser * Command Line Operation * Truth Table implementation * Report writing |
| **Percentage Contribution:** 50% | **Percentage Contribution:** 50% |