README

Name : Tushar Kumar
NetId : tusharku
Collaborators : None

October 26, 2018

1 Introduction

This project, which was implemented by myself, and **did not have any collaboration from anybody else**, implements two inference algorithms. One is the naive truth table enumeration method and the other is avis-Putnam-Logemann-Lovelan(DPLL) algorithm . This project was undertaken as part of the graduate course(CSC 442) at University of Rochester.

2 Technology Used

- Java as the programming language (Java 10.0.2) with no external libraries
- Eclipse used as IDE
- Draw.io for creating design artifacts

3 Building the project

- Download the **AutomatedReasoning-tusharku.zip** file(which you would have if you are reading this file.
- Unzip the file to get the AutomatedReasoning-tusharku folder
- Run the following commands in sequence once you are in the location where you downloaded the zip file: Please mind the line break created because the command being of greater length than width of the page. Actual command would be
  "javac -d executable -sourcepath src
  -cp . src/com/uofr/course/csc442/hw/hw2/reasoning/ReasoningProblemSolver.java"

```
cd AutomatedReasoning-tusharku

javac -d executable -sourcepath src -cp .
 ↪  src/com/uofr/course/csc442/hw/hw2/reasoning/ReasoningProblemSolver.java
```

In case , you were not able to compile this, not to worry, I have provided the already compiled binaries of classes in the bin folder. So you can use that straight away to run the application

4 Running the application

- Run the following command to execute and solve all problems

```
cd AutomatedReasoning-tusharku
java -cp executable
 ↪  com.uofr.course.csc442.hw.hw2.reasoning.ReasoningProblemSolver
```

This, by default would solve ALL the problems and print their output to console

5 Solving Problem

There is a commandline argument which accepts the problem index as its value which can be used to solve that specific problem.

- **index** : This represents the index of the question you want the program to solve
  Possible Values : {1, 2, 3, 4a, 4b, 5, 6a, 6b}
  1 - Modus Ponens
  2 - Simple Wumpus World
  3 - Horn Clauses
  4a - Liars and Truth-tellers-a (OSSMB 82-12)
  4b - Liars and Truth-tellers-b (OSSMB 83-11)
  5 - More Liars and Truth-tellers (adapted from JRM14 392)
  6a - The Doors of Enlightenment (from CRUX 357) : Smullyan's problem
  6b - The Doors of Enlightenment (from CRUX 357) : Liu's problem

5.1 Sample Commands

Here are some sample scenarios with their commands provided:

- **Solving All Problems**

```
java -cp executable
 ↪  com.uofr.course.csc442.hw.hw2.reasoning.ReasoningProblemSolver
```

- **Solving Modus Ponens**

```
java -cp executable
↪  com.uofr.course.csc442.hw.hw2.reasoning.ReasoningProblemSolver
↪  -index 1
```

- **Solving Simple Wumpus World**

```
java -cp executable
↪  com.uofr.course.csc442.hw.hw2.reasoning.ReasoningProblemSolver
↪  -index 2
```

- **Solving Horn Clauses**

```
java -cp executable
↪  com.uofr.course.csc442.hw.hw2.reasoning.ReasoningProblemSolver
↪  -index 3
```

- **Solving Liars and Truth-tellers-a (OSSMB 82-12)**

```
java -cp executable
↪  com.uofr.course.csc442.hw.hw2.reasoning.ReasoningProblemSolver
↪  -index 4a
```

- **Solving Liars and Truth-tellers-b (OSSMB 83-11)**

```
java -cp executable
↪  com.uofr.course.csc442.hw.hw2.reasoning.ReasoningProblemSolver
↪  -index 4b
```

- **Solving More Liars and Truth-tellers (adapted from JRM14 392)**

```
java -cp executable
↪  com.uofr.course.csc442.hw.hw2.reasoning.ReasoningProblemSolver
↪  -index 5
```

- **Solving Doors of Enlightenment : Smullyan's problem**

```
java -cp executable
↪  com.uofr.course.csc442.hw.hw2.reasoning.ReasoningProblemSolver
↪  -index 6a
```

- **Solving Doors of Enlightenment : Liu's problem**

```
java -cp executable
 ↪  com.uofr.course.csc442.hw.hw2.reasoning.ReasoningProblemSolver
 ↪  -index 6b
```