Project 3 AI Report

Cameron Davis, Aboubacar Abdoulaye, Brian Gerkens

March 2023

1 Overview of the Project

This program is an intelligent system that collects user preferences and reasons about them using a knowledge-based approach. The program collects the names and values of attributes (A), hard constraints represented as propositional formulas in conjunctive normal form (CNF) (H), and preference languages of penalty logic, possibilistic logic, and qualitative choice logic with preference formulas (T), also in CNF form. This data is gathered from user input or files with the purpose of performing various tasks.

2 Design

The program is written in Python and uses the Tkinter GUI toolkit for the program's graphical user interface. It leverages pysat to perform tasks such as checking for the existence of feasible objects and for penalty, possibilistic, and qualitative logics.

3 How to start the program

To start the program, follow these three simple steps:

1. Open a terminal and change directory to the "src" directory in the project using the following command:

   ```
   cd /path/to/project/src
   ```

2. Once your inside the src directory Type the following command to run the program using Python 3.11:

   ```
   python3.11 input.py
   ```

   Note: You can check your current directory by typing **pwd** on unix or **cd** on windows to verify your in the src directory.

3. Press the "Enter" key to start the program and follow the on-screen instructions to input your logic attributes, constraints, penalties, and logic types.

Note: Make sure to install Python 3.11 or a later version before running the program.

4 Input Screen

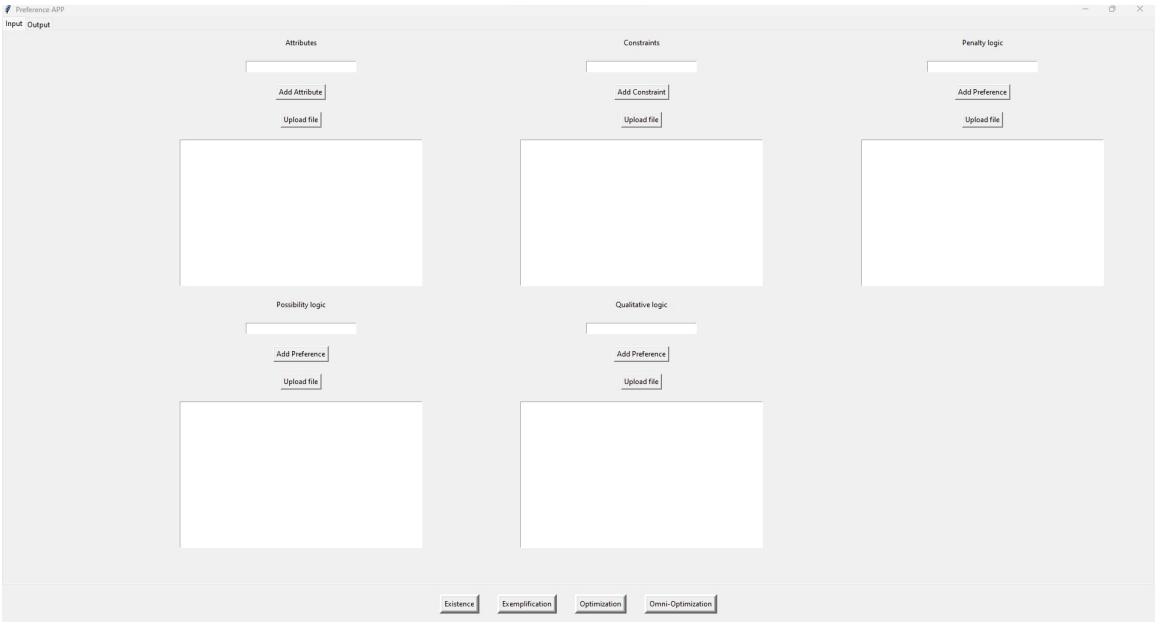

Figure 1: Start Screen of the Program

If the program ran correctly this screen will be shown to the user. On this screen the user can either directly type in the attributes and logic or **upload a file with the Upload File Button**. The Upload file button will open up the default file exploerer on your device and allow you to select a textfile that matches the rules for upload. Inside of the *Testing Case directory* there are files that we used for testing such as beachAttribute and beachConstraints upload these files to the correct location on the input screen if you wish to use our premade testing.

Below is a list of what each text box should contain and what should be uploaded.

4.1 Attributes

The first required input is binary attributes. The program requires at least one or more binary attributes to function. Binary attributes are entered one by one by user input or via a file upload, which is recommended. **Example of attribute, attribute: value1,value2**

4.2 Constraints

The second required input is hard constraints. The program requires at least one or more hard constraints to function. Hard constraints can be entered one by one or in bulk via a file upload, which is recommended. Additional hard constraints can be entered directly in addition to those loaded from a file. **Example of Hard Constraints NOT value1 OR value2**

5 Input Preference Logics

The third and final required input is preference logic, including penalty logic, possibilistic logic, and qualitative choice logic. The program requires at least one or more preference logics for each of the three types to function. Preference logics can be entered one by one or in bulk via a file upload, which is recommended. Additional preference logics can be entered directly in addition to those loaded from a file.

5.1 Penalty Logic

These are rules of penalty logic if a statement is true the knowledge system will know and assign it a penalty value. s **Example is value AND value2, 20**, the penalty is going to be the number in this file.

5.2 Possibilistic Logic

These are rules similar to penalty but instead the value is subtracted from 1 if the CNF statement is true. **Example value1 AND value2, 0.8**

5.3 Qualitative Logic

These are qualitative logic choices in which BT is used instead of ¿ and IF is the equal operator. **Example, value BT value2 IF value3**.

6 Output Page

After your file have been uploaded select the **Output** tab in the top of the page.Upon selecting this tab you will see a page that looks like the page below if done correctly.

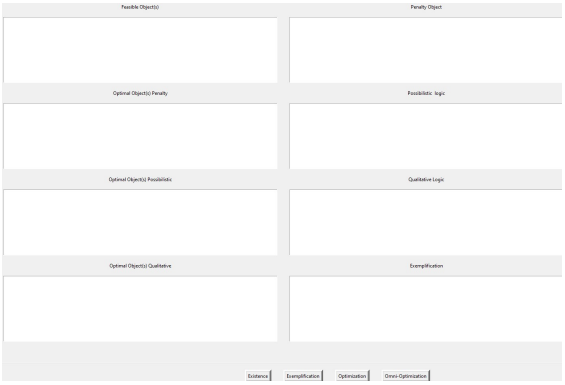

Figure 2: Output screen of the Program

7 Using the output page

There are Four buttons located on the bottom of the page *Existence, Exemplification, Optimization, and Omni-Optimization*, upon a user clicking a button and the required fields having input in them (see section above) the button will generated the required output.

Below is a list of all the buttons how there code works and the output they generate

7.1 Existence Button

Existence button will generate a list of feasible objects. Which are identified by passing all the hard constraints (which are converted by pysat to a acceptable format). as clauses to match for satisfactory and obtaining the sets of objects returned by the solver. The result is stored in a list and used by other functions.

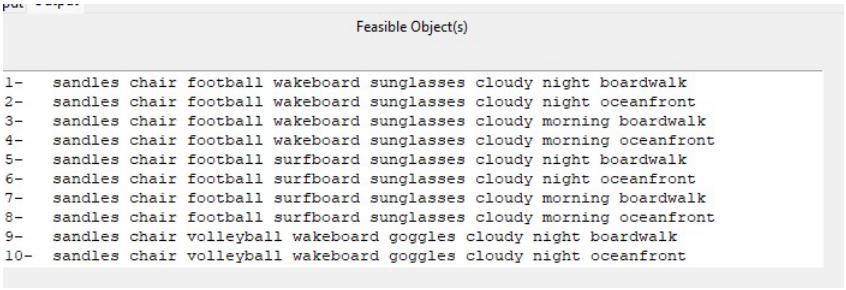

Figure 3: Output of Existence Button being triggered

7.2 Exemplification Button

Exemplification is done in the preferences.py function. First, we need to find the penalty, tolerances and satisfiability of the feasible models based on respectively the penalty logic, possibility logic and qualitative logic. Once we figure out all those values and complete our tables, we can start by first randomly picking two feasible objects and then comparing them based on each logic. We used pysat to facilitate the study of the satisfiability of each preference for each model.

NOTE: that clicking on the exemplification button multiple times will compare two new feasible objects and display them in the exemplification output box.

Exemplification will then populate the Penalty Logic, Possibilistic Logic, and qualitative choice logic tables with the necessary underlying data used for exemplification named above. The program output provides a comparison between two objects (preferred, equivalent and incomparable) based on each logic.

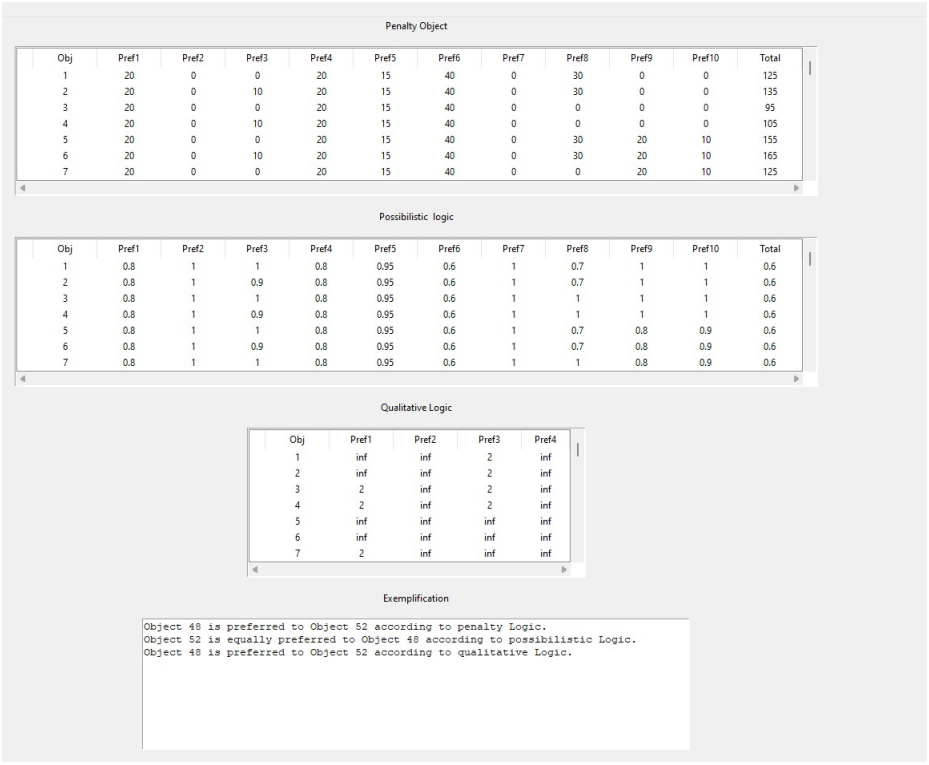

Figure 4: Output of Exemplification Button being triggered

7.3 Optimization

For penalty logic the optimal object is obtained by returning the first element in a list of models sorted in ascending order by total penalty value. For possibility logic the optimal object is obtained by returning the first element in a list of models sorted in descending order by total tolerance value. For qualitative logic first we need to trough each feasible model and compare the satisfaction value obtained by each preference and figure all the optimal objects and saved it in list. The optimal object is obtained by returning the first element of that list.

The output for optimization is generated in 3 output boxes *Optimal Object Penalty,Optimal Object Possibilistic, Optimal Object Qualitative* these boxes will show the most optimal object in each of the tables generated by exemplification.

Figure 5: Output of Optimization Button being triggered

7.4 Omni-Optimization

Omni optimization is very similar to optimization since we have to give all optimal objects rather than one. For penalty we display all the models having the same value as optimal penalty and for the possibility logic we display all the models having the same value as optimal possibility value. For qualitative logic we display the content of the list we find by figure out all the optimal qualitative object in the optimization section

Similar to the output for optimization is generated in 3 output boxes *Optimal Object Penalty,Optimal Object Possibilistic, Optimal Object Qualitative* these boxes will show populated with optimal objects based on each logic, along with the total number of optimal objects for each preference

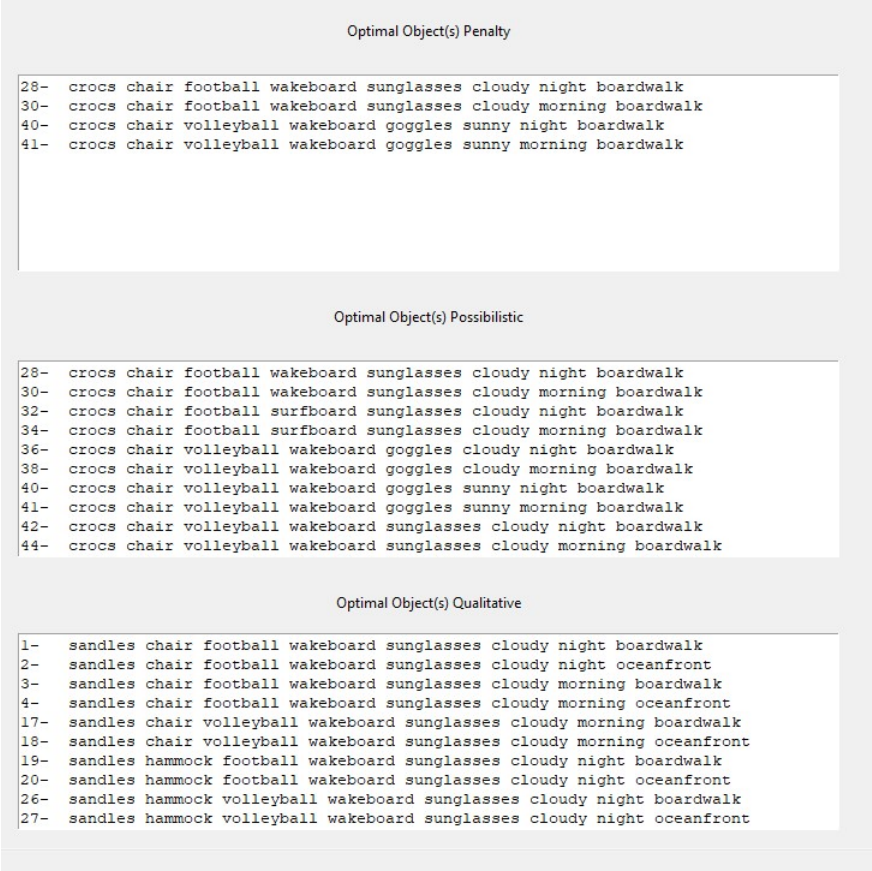

Figure 6: Output of Omni-Optimization Button being triggered