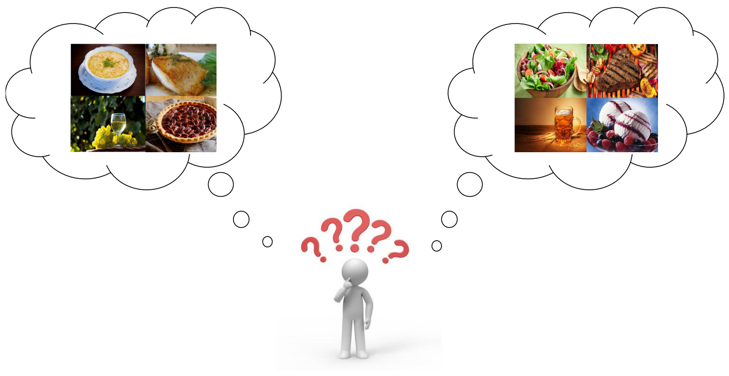
CAP4630 Project 3 – Preference Agent (PrefAgent)

Due Date: Friday, 3/28/2025 11:59 PM



In this individual Python project, you will design and build a knowledge-based intelligent system to solve a preference problem via collecting user preferences and reasoning about them.

The system should have an easy-to-use interface for collecting from the user attributes and their values, hard constraints, and preferences. The system should allow reading in these data from files. (See section "File Formats" for formats of these files.) The three components of a preference problem are the following:

1. Attributes: in this project all attributes are binary.
2. Hard constraints (H): represented as propositional formulas in the Conjunctional Normal Form (CNF).
3. Preference theory (T): your program should support two preference logics of penalty logic and qualitative choice logic. Formulas involved in the preference theories (i.e., the ϕ's and the ψ's) are of CNF as well.

The system should support the following tasks:

1. **Encoding**: encode all objects using binary codes. The order of attributes as listed in the attributes file (see "File Formats" section) dictates the order of binary bits, and the two values per each attribute will be encoded as 1 and o in the order as listed. For example, the object <cake, beer, fish, salad> will be encoded as 1010 in the example below in "File Formats" section.
2. **Feasibility Checking**: decide whether there are feasible objects w.r.t H, that is, whether there are models of H that are truth assignments making H true.
3. **Show the Table**: present the table to show all feasible objects w.r.t H and for each one show the penalty or satisfaction degree values across all preference rules.
4. **Exemplification:** generate, if possible, two random feasible objects, and show the preference between the two (strict preference, equivalence, or incomparison) w.r.t T.

5. **Omni-optimization**: find all optimal feasible objects w.r.t T.

Implementation Requirements:

1. Your submission should contain all the files and directories as described in the next section "Deliverables." I will use Python 3.11 to run your submissions, and for this I will run command `` `python3.11 main.py` `` in your project root directory assuming your main program to start is main.py. If there is any package that your program installs and imports, make sure to include such information in the README.
2. For testing, the system should solve an instance, developed by you, that contains at least **6 hard constraints** and at least ***3 preference rules*** per each logic over at least ***8 binary attributes***.
3. In *ExampleTestCase* directory, a small example instance is provided that has three attributes, one constraint, two rules in a penalty logic theory, and four rules in qualitative choice logic. Expected results are provided for this small instance in the "Example Output" section.
4. PySAT is a Python package that can be used to check if a propositional formula is satisfiable, if a truth assignment satisfies or falsifies a formula, etc. A tutorial is published on Canvas. Feel free to use PySAT or other ways in your project to check satisfiability of formulas.
5. Submissions will be checked for peer similarity for **academic integrity**. In case of violation, the minimum penalty will be a zero on the project and a misconduct report.

Deliverables:

Each student will zip the following into [your-last-name]_Project3.zip and submit to Canvas.

1. A directory called "ExampleTestCase" that contains text files of the instance I provided for testing. Refer to "Example Output" section to see what is expected from your system using this test case.
2. A directory called "TestCase" that contains text files of the instance you created for testing. File names should be straightforward, similar to those in ExampleTestCase.
3. A directory called "src" that contains all the source codes.
4. A README file that contains instructions to build and run your system.
5. A PDF report that describes how your system works and shows the testing results using the test instance created by you (e.g., various screen shots of various steps).

File Formats:

Attributes File

Each row in this file describes one attribute and its two values. Attributes and their values are lower-case strings. Each attribute is followed by a colon, which is followed by two values separated by a comma. Use different strings for values to avoid confusion. An example format is as follows.

```
dissert:  cake, ice-cream
drink:  wine, beer
entree:  fish, beef
appetizer:  soup, salad
...
```

Hard Constraints File

Hard constraints essentially are a long CNF formula, so in this file each row is a clause, which is a disjunction of literals that could be positive or negative. A literal is positive if it is a value of some attribute, and negative if it is the negation of a value of some attribute. Inside a clause, negation (NOT) and disjunction (OR) are all upper cases, and values are again lower cases. An example format is as follows.

```
NOT soup OR NOT ice-cream
NOT beef OR salad
...
```

Preferences File

Penalty Logic

In this file each row describes one penalty logic rule which is a comma-separated pair. The first part is a CNF propositional formula using values and connectives (AND, OR, and NOT), and the second part is the penalty value. An example format is as follows.

```
fish AND wine, 10
wine OR cake, 6
beer AND beer OR beef AND NOT soup, 7
...
```

Qualitative Choice Logic

In this file each row describes one qualitative choice logic rule which is of format "ϕ1 BT . . . ϕn IF ψ," where BT reads better than (>) and IF is follwed by the condition formula ψ. Formulas here also are CNF. An example format is as follows, where "BT" stands for "better than".

```
fish BT beef IF
wine BT beer IF fish
beef AND beer BT fish AND beer BT beef AND wine IF
cake BT ice-cream IF soup
...
```

Example Output:

```
Welcome to PrefAgent!

Enter Attributes File Name: attributes.txt

Enter Hard Constraints File Name: constraints.txt

Choose the preference logic to use:
1. Penalty Logic
2. Qualitative Choice Logic
3. Exit

Your Choice: 1

You picked Penalty Logic
Enter Preferences File Name: penaltylogic.txt

Choose the reasoning task to perform:
1. Encoding
2. Feasibility Checking
3. Show the Table
4. Exemplification
5. Omni-optimization
6. Back to previous menu

Your Choice: 1
o0 – ice-cream, beer, beef
o1 – ice-cream, beer, fish
o2 – ice-cream, wine, beef
o3 – ice-cream, wine, fish
o4 – cake, beer, beef
o5 – cake, beer, fish
o6 – cake, wine, beef
o7 – cake, wine, fish

Choose the reasoning task to perform:
1. Encoding
2. Feasibility Checking
3. Show the Table
4. Exemplification
5. Omni-optimization
6. Back to previous menu

Your Choice: 2
Yes, there are 6 feasible objects.

Choose the reasoning task to perform:
1. Encoding
```

```
2. Feasibility Checking
3. Show the Table
4. Exemplification
5. Omni-optimization
6. Back to previous menu

Your Choice: 3
+----------+--------------+-------------+--------------+
| encoding | fish AND wine | wine OR cake | total penalty |
+----------+--------------+-------------+--------------|
|       o0 |           10 |           6 |           16 |
|       o1 |           10 |           6 |           16 |
|       o4 |           10 |           0 |           10 |
|       o5 |           10 |           0 |           10 |
|       o6 |           10 |           0 |           10 |
|       o7 |            0 |           0 |            0 |
+----------+--------------+-------------+--------------+

Choose the reasoning task to perform:
1. Encoding
2. Feasibility Checking
3. Show the Table
4. Exemplification
5. Omni-optimization
6. Back to previous menu

Your Choice: 4
Two randomly selected feasible objects are o7 and o5,
and o7 is strictly preferred over o5.

Choose the reasoning task to perform:
1. Encoding
2. Feasibility Checking
3. Show the Table
4. Exemplification
5. Omni-optimization
6. Back to previous menu

Your Choice: 5
All optimal objects: o7

Choose the reasoning task to perform:
1. Encoding
2. Feasibility Checking
3. Show the Table
4. Exemplification
5. Omni-optimization
6. Back to previous menu

Your Choice: 6
```

```
Choose the preference logic to use:
1. Penalty Logic
2. Qualitative Choice Logic
3. Exit

Your Choice: 2

You picked Qualitative Choice Logic
Enter Preferences File Name: qualitativechoicelogic.txt

Choose the reasoning task to perform:
1. Encoding
2. Feasibility Checking
3. Show the Table
4. Exemplification
5. Omni-optimization
6. Back to previous menu

Your Choice: 1
o0 – ice-cream, beer, beef
o1 – ice-cream, beer, fish
o2 – ice-cream, wine, beef
o3 – ice-cream, wine, fish
o4 – cake, beer, beef
o5 – cake, beer, fish
o6 – cake, wine, beef
o7 – cake, wine, fish

Choose the reasoning task to perform:
1. Encoding
2. Feasibility Checking
3. Show the Table
4. Exemplification
5. Omni-optimization
6. Back to previous menu

Your Choice: 2
Yes, there are 6 feasible objects.

Choose the reasoning task to perform:
1. Encoding
2. Feasibility Checking
3. Show the Table
4. Exemplification
5. Omni-optimization
6. Back to previous menu

Your Choice: 4
```

```
Two randomly selected feasible objects are o0 and o1,
and o0 and o1 are incomparable.

Choose the reasoning task to perform:
1. Encoding
2. Feasibility Checking
3. Show the Table
4. Exemplification
5. Omni-optimization
6. Back to previous menu

Your Choice: 4
Two randomly selected feasible objects are o4 and o0,
and o0 is strictly preferred over o4.

Choose the reasoning task to perform:
1. Encoding
2. Feasibility Checking
3. Show the Table
4. Exemplification
5. Omni-optimization
6. Back to previous menu

Your Choice: 5
All optimal objects: o0, o1, o7

Choose the reasoning task to perform:
1. Encoding
2. Feasibility Checking
3. Show the Table
4. Exemplification
5. Omni-optimization
6. Back to previous menu

Your Choice: 55
Wrong Choice! Enter your choice: 6

Choose the preference logic to use:
1. Penalty Logic
2. Qualitative Choice Logic
3. Exit

Your Choice: ABC
Wrong Choice! Enter your choice: 3

Bye!
```