ID	**20126041**
Full name	Nguyễn Huỳnh Mẫn
Class	20VP

Artificial Intelligence Foundation
CSC14003_20VP

Lab02: PL Resolution

REPORT

I. Evaluation Criteria:

No.	Criteria	Completion
1	Reads the input data and stores it in the appropriate data structure	100%
2	Install the fusion algorithm on propositional logic	100%
3	The inference steps generate enough propositions and correct conclusions	100%
4	Follow the format description of the question	100%
5	Five test case reports and reviews	100%
6	Discussion on the algorithm's efffiency and suggestions.	100%

II. Brief description of main functions and it's utilities:

1. main.py

a. main

This function performs all of the following basic action with each test of 5 test cases:

- Read the input data and store it in appropriate data strutures.
- Call the function pl_resolution, which implements the PL Resolution algorithm.
- Write the output data to the output file in valid format.

2. MyAlgo.py

Class MyAlgo has 5 attributes:

- *alpha*: stores information of the alpha query (list).
- *KB*: stores information of the Knowledge Base (list).
- *new_clauses_list*: stores new clauses after each loop of the PL Resolution algorithm (list).
- *solution:* represent the result of the query (bool).
- *Util:* An instance of the 'Util' class from "Util.py", which is used for utility purposes.

a. ***read_file***

This function helps you read an input data from an input file into the Knowledge Base (*KB*) and Alpha (*alpha*).

All of clauses are standardized:

- Get rid of all of duplicates.
- Literals within a clause are sorted base on alphabetical prioritized order.

KB and *alpha* are standardized also:

- Get rid of all of duplicates.
- Get rid of all of clauses in which two complementary literals appear.

b. ***pl_resolution***

This function helps you query *alpha* based on the *KB* by the PL Resolution algorithm, the result of the query *alpha* is stored in *solution*, all of new clauses of each loop are stored in *new_clauses_list*.

c. ***wirte_file***

This function write an ouput data to an output file in valid format.

3. *Util.py*

a. *standard_cnf_sentence*

This function helps you to standardized a CNF sentence such as *KB* or *alpha*:

- Standardize all of clauses.
  - Get rid of all of duplicates.
  - Literals within a clause are sorted following the alphabetical order.
- Discard all of clauses in which 2 complementary literals appears.

b. *negation_of_cnf_sentence*

This function helps you generate a negation of a CNF sentence. I choose to implement the distribution algorithm on this function.

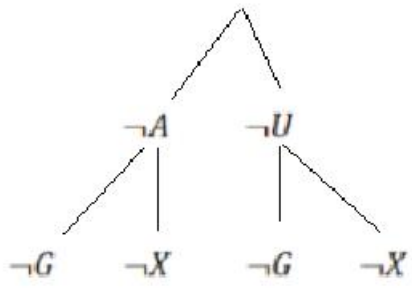
Idea:

$$alpha: (A \vee U) \wedge (G \vee X)$$

$$negation\ of\ alpha: (\neg A \wedge \neg U) \vee (\neg G \wedge \neg X)$$
$$\equiv (\neg A \vee \neg G) \wedge (\neg A \vee \neg X) \wedge (\neg U \vee \neg G) \wedge (\neg U \vee \neg X)\ (distribution)$$

To implement the distribution, I create a tree recursive likes the below one. The result of the distribution is conjunction of clauses, with each clause is a disjunction of all literals on each branch of the tree.

For further understading, please read 2 functions generate_combinations and generate_combinations_recursive.



c. ***resolve***

This function helps you to resolve 2 clauses then return a list of resolvents (list of clauses).

d. **generate_combinations_recursive**

- A helper method for generating combinations from multiple sets recursively.

e. **generate_combinations**

- Generates a combination list from multiple sets.
- Takes a list of sets and returns all possible combinations of elements from these sets.

EX: GENERATE A COMBINATION LIST:

- ***set_1 (x,y)***
- ***set_2 (a,b)***

⇨ ***COMBINATION SET: (x, a), (x, b), (y, a), (y, b)***

4. Defi.py

Defines an enumeration class named Definition using the Enum module from the Python standard library:

- EMPTY_CLAUSE: Represents a symbol used to denote an empty clause. In this case, it is defined as the string '{}'.
- NOT_OPERATOR: Represents a symbol used as the prefix for the negation (NOT) operator. In this case, it is defined as the string '-'.
- OR_OPERATOR: Represents a symbol used as a delimiter for the OR operator. In this case, it is defined as the string ' OR '.

5. *StaticMethod.py*

This file defines a class named Stt_Med that contains several static methods related to manipulating and checking properties of logical clauses, particularly in the context of some logical reasoning or resolution algorithm. Here's a summary of what each method does:

a. *is_empty_clause -> bool:*

- Checks whether a given clause is empty.
- Returns True if the length of the clause is zero, indicating that it is an empty clause.

b. *standard_clause -> list:*

- Returns a standardized version of a logical clause.
- Removes duplicate literals from the clause.
- Sorts the literals within the clause based on an alphabetical prioritized order.

c. *is_complementary_literal -> bool:*

- Checks whether two literals are complementary.
- Returns True if the literals have different lengths and the last character of each literal is the same.

d. *negation_of_literal -> str:*

- Returns the negation of a given literal.
- If the literal starts with the negation symbol (defined in the Defi.py file), it removes the negation; otherwise, it adds the negation symbol to the beginning of the literal.

III. Discussion on the algorithm's efficiency and suggestions to improve:

a. Advantages:

- Easy to read and understand: The source code is clear and has comments to help readers easily understand the logic and decoding process.
- Organized functions: Source code is organized into separate functions and files such as reading file, logical solutions and write_file, helping to manage source code results.
- Use Enum: Using Enum helps define constants clearly and makes maintenance easy.
- Use deepcopy library: Using deepcopy from a secured copy of the library works with an independent copy of the data, helping to avoid reference issues.

b. Disadvantages:

- Algorithm performance: One can consider how to optimize the fusion algorithm to improve performance, especially when dealing with large data sets. Improved pruning strategies or algorithms can be used. Despite the meticulous and efficient organization and division of code, there is always room for improvement.
- Memory management: Use memory management strategies effectively to avoid wasting resources when working with large data sets. Using deepcopy can result in wasting memory resources. There needs to be a better way to handle data to avoid wasting memory space.

c. Proposal:

- Optimization Strategy: The proposed optimization for the PL Resolution algorithm seeks to improve efficiency by minimizing redundant resolutions during iterative clause resolution. In the conventional approach, each iteration involves resolving clauses from both the Knowledge Base (KB) and the negation of the goal (not

alpha), leading to potentially redundant resolutions.However, this process may redundantly readdress resolutions that have already been established in prior iterations.

- Focused Resolution: The optimization strategy introduces a focused resolution approach during each iteration. Specifically, at iteration n, the algorithm selectively targets resolutions between a clause Ci from the KB and a clause Cj from the new clauses generated in the preceding iteration (n - 1). This selection is based on the condition that the index i is less than j. The rationale is that resolutions within the same set (KB or new clauses) have likely been covered in earlier iterations.

- Illustration:

KB and not alpha	Loop 1	Loop 2	Loop 3
(1) …	(5) …	(8) …	(12) …
(2) …	(6) …	(9) …	
(3) …	(7) …	(10) …	
(4) …		(11) …	

To illustrate, consider the resolutions in a tabular format:

- At Loop 1, resolutions involve clauses (1) with (2), (1) with (3), ..., (3) with (4).
- At Loop 2, resolutions involve clauses (5) with (6), (5) with (7), ..., (7) with (8).

- Targeted Approach: The optimization ensures that resolutions within the same set are bypassed, focusing on resolving clauses that have not been previously addressed within that set. This targeted approach streamlines the resolution process, contributing to overall algorithm efficiency.

THE END