VIETNAM NATIONAL UNIVERSITY OF HO CHI MINH CITY

UNIVERSITY OF SCIENCE

FACULTY OF INFORMATION TECHNOLOGY

Report
Lab 02: Logic

Course name: Fundamentals of Artificial Intelligence

CSC14003_22CLC09

Student:
Nguyen Ho Dang Duy
22127085
22CLC09

Lecturer:
Bui Duy Dang
Nguyen Thi Thu Hang
Pham Trong Nghia

Ho Chi Minh city, August 2024

Contents

1 Information page

- Full name: Nguyen Ho Dang Duy
- Student ID: 22127085
- Class: 22CLC09

2 Detailed Analysis of Functions

2.1 Class Clause

2.1.1 Clause.parse(clause_str: str)

Purpose: Parses a string representation of a clause into a `Clause` object. Each clause is a disjunction of literals.
Example:

```
Clause.parse("A OR -B OR C") = Clause({A, -B, C})
```

2.1.2 Clause.negate()

Purpose: Returns a list of `Clause` objects representing the negation of each literal in the original clause.
Example:

```
Clause({A OR -B}).negate() = List({Clause({-A}), Clause({B})})
```

2.1.3 Clause.negate_literal(literal: str)

Purpose: Returns the negation of a single literal.
Example:

```
Clause.negate_literal(A) = -A
Clause.negate_literal(-B) = B
```

2.1.4 Clause.resolve(other: Clause)

Purpose: Resolves the current clause with another clause to find new resolvents.
Example:

```
Clause({A, B}).resolve(Clause({-A, C})) = List({Clause({B, C})})
```

2.1.5 Clause.contains_tautology()

Purpose: Checks if the clause contains a tautology, i.e., a literal and its negation.
Example:

```
Clause({A, -A, B}).contains_tautology() = True
Clause({A, B}).contains_tautology() = False
```

2.2 Class KnowledgeBase

2.2.1 KnowledgeBase.add_clause(clause: Clause)

Purpose: Adds a clause to the knowledge base.
Example:

```
KnowledgeBase().add_clause(Clause({A, -B})) = KnowledgeBase({Clause({A, -B})})
```

2.2.2 KnowledgeBase.print_kb()

Purpose: Prints the current state of the knowledge base to command line.
Example:

```
KnowledgeBase().print_kb() = Prints all clauses in the knowledge base
```

2.2.3 KnowledgeBase.print_resolutions(resolutions)

Purpose: Prints the resolutions performed in the current loop to command line.
Example:

```
KnowledgeBase().print_resolutions(List of resolutions) = Prints all resolutions
```

2.3 Other functions

2.3.1 print_kb()

Purpose: To display the current state of the Knowledge Base (KB), showing all the clauses it contains.
Example: If the KB contains the clauses ['-A', 'B'] and ['A', '-B'], the output will be:

```
-A OR B
A OR -B
------
```

2.3.2 print_resolutions(self, resolutions)

Purpose: To print the results of the resolution process for each loop iteration, including the clauses being resolved and their resolvent. It also provides an explanation if an empty clause is derived.
Example: If resolving ['-A', 'B'] with ['A', '-B'] results in an empty clause, the output will be:

```
Loop 1:
Resolving: -A OR B with A OR -B
Result: {}
------
```

2.3.3 format_output(all_steps: List[List[Clause]], entails: bool)

Purpose: Formats the output into the required format, showing all resolved clauses and whether the query is entailed by the knowledge base.
Example:

```
format_output(List of all steps, True) = Formatted string with steps and "YES"
```

2.3.4 parse_input(file_path: str)

Purpose: Parses the input file to extract the query and knowledge base clauses.
Example:

```
parse_input("input.txt") = (Clause({A}), [Clause({B}), Clause({-B})])
```

2.3.5 main()

The main function to run the PL-Resolution program based on command-line arguments. You can run the program in command line with the following options:

- **-i** or –**input_file**: Path to the input file.
- **-o** or –**output_file**: Path to the output file
- **-all**: Run all input files in the Input folder

So you can run the program in 2 ways with these syntax:

```
python source_code.py -i <input_file> -o <output_file>
```

This syntax can run the program with input file path and write result to your output file path. Or you can use:

```
python source_code.py -all
```

To run all the input files in this lab.

3 PL Resolution Algorithm

3.1 Function Signature

```
FUNCTION pl_resolution(alpha: Clause) RETURNS (List of List of Clause, Boolean)
```

Purpose: Determine if the knowledge base entails a given query using propositional logic resolution.

3.2 Detailed Explanation

3.2.1 Negation of the Query

```
negated_alpha_clauses = NEGATE(alpha)
FOR EACH negated_clause IN negated_alpha_clauses:
    ADD_CLAUSE_TO_KB(negated_clause)
```

- **Action:** Negate the query α and add the negated clauses to the knowledge base.
- **Reason:** By adding $\neg\alpha$, we check if the conjunction of the knowledge base and negated query is unsatisfiable.

Example:
If the query α is $A \vee B$, then:

$$\neg\alpha = \neg A \wedge \neg B = \{\{-A\}, \{-B\}\}$$

After adding to the knowledge base:

$$KB = \{\{-A\}, \{-B\}, \text{original KB clauses}\}$$

3.2.2 Initialize Data Structures

```
all_clauses = COPY_OF(KB)
all_steps = EMPTY_LIST()
all_resolutions = EMPTY_LIST()
```

- **Action:** Create a copy of the clauses and initialize lists for resolution tracking.
- **Reason:** These structures track the resolution process and new clause generation.

3.2.3 Main Resolution Loop

```
WHILE TRUE:
    INCREMENT(loop_count)
    pairs = GENERATE_UNIQUE_PAIRS(all_clauses)
    step_clauses = EMPTY_LIST()

    FOR EACH (clause1, clause2) IN pairs:
        resolvents = RESOLVE(clause1, clause2)

```

```
        FOR EACH resolvent IN resolvents:
            IF resolvent IS EMPTY:
                ADD_TO(step_clauses, resolvent)
                ADD_TO(all_steps, COPY_OF(step_clauses))
                PRINT_RESOLUTIONS(all_resolutions)
                RETURN (all_steps, True)

            IF NOT EXISTS(resolvent, all_clauses) AND NOT EXISTS(resolvent,
  step_clauses):
                ADD_TO(step_clauses, resolvent)

            ADD_TO(all_resolutions, (clause1, clause2, resolvent))
```

Purpose: Continues resolving pairs of clauses until an empty clause is found or no new clauses can be generated.
Inner Loop Details:

1. **Pair Selection:**

```
pairs = GENERATE_UNIQUE_PAIRS(all_clauses)
```

   - **Action:** Select all unique pairs of clauses (ci, cj).
   - **Reason:** Find new resolvent clauses from these pairs.

2. **Resolvent Generation:**

```
resolvents = RESOLVE(clause1, clause2)
```

   - **Action:** Resolve pairs to generate new clauses.
   - **Reason:** Apply resolution rule to eliminate contradictory literals.

3. **Check for Empty Clause:**

```
IF resolvent IS EMPTY:
    ADD_TO(step_clauses, resolvent)
    ADD_TO(all_steps, COPY_OF(step_clauses))
    PRINT_RESOLUTIONS(all_resolutions)
    RETURN (all_steps, True)
```

   - **Action:** Check for an empty clause {}.
   - **Reason:** An empty clause indicates contradiction, proving entailment.

4. **Add Non-Tautological Clauses:**

```
IF NOT EXISTS(resolvent, all_clauses) AND NOT EXISTS(resolvent,
   step_clauses):
    ADD_TO(step_clauses, resolvent)

ADD_TO(all_resolutions, (clause1, clause2, resolvent))
```

   - **Action:** Add resolvent if non-duplicate and non-tautological.

- **Reason:** Avoid redundant clauses and ensure efficient resolution.

Example:
Resolve these clauses:

$$C1 = \{A, B\}$$
$$C2 = \{-A, C\}$$

Resolution:
Resolving $C1$ and $C2$ eliminates A and $-A$, yielding:

$$R1 = \{B, C\}$$

An empty clause $\{\}$ implies contradiction.

3.2.4 Check for Conclusion

```
IF step_clauses IS EMPTY:
    ADD_TO(all_steps, step_clauses)
    PRINT_RESOLUTIONS(all_resolutions)
    RETURN (all_steps, False)
```

- **Action:** If no new clauses were added, conclude no entailment.
- **Reason:** No contradictions mean $KB \wedge \neg\alpha$ is satisfiable.

3.2.5 Add New Clauses and Repeat

```
SORT_AND_DEDUPLICATE(step_clauses)
ADD_TO(all_steps, step_clauses)
EXTEND(all_clauses, step_clauses)
PRINT_RESOLUTIONS(all_resolutions)
```

- **Action:** Sort, deduplicate, and add new clauses to the next iteration.
- **Reason:** Expand knowledge base for subsequent resolutions.

4 Detailed explanation for test cases

Because the program uses a *while True* loop, there will be many repeated pairs of clauses and this report will become very long and difficult to read. So the following explanation only records the newly generated clause pairs in each iteration. If you want to view full result, you can run each test case and it will be printed in command line.

4.1 input01.txt

```
-A
4sult
-A OR B
B OR -C
A OR -B OR C
-B
```

Knowledge Base after adding negation of alpha:

- $-A$ OR B
- B OR $-C$
- A OR $-B$ OR C
- $-B$
- A

Loop 1:

- Resolving: $-A$ OR B with $-B \Rightarrow$ Get $-A$
- Resolving: $-A$ OR B with $A \Rightarrow$ Get B
- Resolving: B OR $-C$ with $-B \Rightarrow$ Get $-C$

Loop 2:

- Resolving: B with $-B \Rightarrow$ Get {}
- Resolving: A OR $-B$ OR C with $B \Rightarrow$ Get A OR C
- Resolving: A OR $-B$ OR C with $-C \Rightarrow$ Get A OR $-B$
- Resolving: A OR $-B$ OR C with $-A \Rightarrow$ Get $-B$ OR C

Output: We got $-A$ after run pl-resolution so KB entails α

```
3
-A
B
-C
4
{}
A OR C
A OR -B
-B OR C
YES
```

4.2 input02.txt

```
A
4
-A OR B
-C OR B
A OR C OR -B
-B
```

Knowledge Base after adding negation of alpha:

- $-A$ OR B
- B OR $-C$
- A OR $-B$ OR C
- $-B$
- $-A$

Loop 1:

- Resolving: B OR $-C$ with $-B \Rightarrow$ Get $-C$
- Resolving: A OR $-B$ OR C with $-A \Rightarrow$ Get $-B$ OR $-C$

Loop 2:

- Resolving: $-A$ OR B with $-B$ OR $C \Rightarrow$ Get $-A$ OR C
- Resolving: A OR $-B$ OR C with $-C \Rightarrow$ Get A OR $-B$

Loop 3:

- Resolving: B OR $-C$ with A OR $-B \Rightarrow$ Get A OR $-C$

Loop 4:

- No resolutions in this loop.

Output: We didn't get A after run pl-resolution so KB not entails α

```
2
-C
-B OR C
2
-A OR C
A OR -B
1
A OR -C
0
NO
```

4.3 input03.txt

```
-A
4
-C OR B
-D OR B
-B OR -A
-E OR -D OR B
```

Knowledge Base after adding negation of alpha:

- B OR $-C$
- B OR $-D$
- $-A$ OR $-B$
- B OR $-D$ OR $-E$
- A

Loop 1:

- Resolving: $-A$ OR $-B$ with $A \Rightarrow$ Get $-B$
- Resolving: B OR $-C$ with $-A$ OR $-B \Rightarrow$ Get $-A$ OR $-C$
- Resolving: B OR $-D$ with $-A$ OR $-B \Rightarrow$ Get $-A$ OR $-D$
- Resolving: $-A$ OR $-B$ with B OR $-D$ OR $-E \Rightarrow$ Get $-A$ OR $-D$ OR $-E$

Loop 2:

- Resolving: B OR $-C$ with $-B \Rightarrow$ Get $-C$
- Resolving: B OR $-D$ with $-B \Rightarrow$ Get $-D$
- Resolving: B OR $-D$ OR $-E$ with $-B \Rightarrow$ Get $-D$ OR $-E$

Loop 3:

- No resolutions in this loop.

Output: We didn't get $-A$ after run pl-resolution so KB not entails α

```
4
-B
-A OR -C
-A OR -D
-A OR -D OR -E
3
-C
-D
-D OR -E
0
NO
```

4.4 input04.txt

```
E
5
A
B
D
-A OR -B OR C
-C OR -D OR E
```

Knowledge Base after adding negation of alpha:

- A
- B
- D
- $-A$ OR $-B$ OR C
- $-C$ OR $-D$ OR E
- $-E$

Loop 1:

- Resolving: A with $-A$ OR $-B$ OR $C \Rightarrow$ Get $-B$ OR C
- Resolving: B with $-A$ OR $-B$ OR $C \Rightarrow$ Get $-A$ OR C
- Resolving: D with $-C$ OR $-D$ OR $E \Rightarrow$ Get $-C$ OR E
- Resolving: $-A$ OR $-B$ OR C with $-C$ OR $-D$ OR $E \Rightarrow$ Get $-A$ OR $-B$ OR $-D$ OR E
- Resolving: $-C$ OR $-D$ OR E with $-E \Rightarrow$ Get $-C$ OR $-D$

Loop 2:

- Resolving: A with $-A$ OR $-B$ OR $-D$ OR $E \Rightarrow$ Get $-B$ OR $-D$ OR E
- Resolving: A with $-A$ OR $C \Rightarrow$ Get C
- Resolving: B with $-A$ OR $-B$ OR $-D$ OR $E \Rightarrow$ Get $-A$ OR $-D$ OR E
- Resolving: D with $-A$ OR $-B$ OR $-D$ OR $E \Rightarrow$ Get $-A$ OR $-B$ OR E
- Resolving: D with $-C$ OR $-D \Rightarrow$ Get $-C$
- Resolving: $-A$ OR $-B$ OR C with $-C$ OR $-D \Rightarrow$ Get $-A$ OR $-B$ OR $-D$
- Resolving: $-A$ OR C with $-C$ OR $-D \Rightarrow$ Get $-A$ OR $-D$
- Resolving: $-A$ OR C with $-C$ OR $E \Rightarrow$ Get $-A$ OR E
- Resolving: $-B$ OR C with $-C$ OR $-D \Rightarrow$ Get $-B$ OR $-D$

- Resolving: $-B$ OR C with $-C$ OR $E \Rightarrow$ Get $-B$ OR E

Loop 3:

- Resolving: A with $-A$ OR $-D$ OR $E \Rightarrow$ Get $-D$ OR E
- Resolving: $-A$ OR $-B$ OR C with $-C \Rightarrow$ Get $-A$ OR $-B$
- Resolving: $-A$ OR C with $-C \Rightarrow$ Get $-A$
- Resolving: $-B$ OR C with $-C \Rightarrow$ Get $-B$
- Resolving: $-C$ OR $-D$ with $C \Rightarrow$ Get $-D$
- Resolving: $-C$ OR E with $C \Rightarrow$ Get E
- Resolving: $-C$ with $C \Rightarrow$ Get {}

Output: We got E after run pl-resolution so KB entails α

```
5
-A OR C
-B OR C
-C OR -D
-C OR E
-A OR -B OR -D OR E
10
-C
C
-A OR -D
-A OR E
-B OR -D
-B OR E
-A OR -B OR -D
-A OR -B OR E
-A OR -D OR E
-B OR -D OR E
7
{}
-A
-B
-D
E
-A OR -B
-D OR E
YES
```

4.5 input05.txt

```
A
6
-B OR C
B OR -C
B OR D
-C OR A
-D OR A
-A OR E
```

Knowledge Base after adding negation of alpha:

- $-B$ OR C
- B OR $-C$
- B OR D
- A OR $-C$
- A OR $-D$
- $-A$ OR E
- $-A$

Loop 1:

- Resolving: $-B$ OR C with B OR $D \Rightarrow$ Get C OR D
- Resolving: $-B$ OR C with A OR $-C \Rightarrow$ Get A OR $-B$
- Resolving: B OR D with A OR $-D \Rightarrow$ Get A OR B
- Resolving: A OR $-C$ with $-A$ OR $E \Rightarrow$ Get $-C$ OR E
- Resolving: A OR $-C$ with $-A \Rightarrow$ Get $-C$
- Resolving: $-A$ OR $-D$ with $-A$ OR $E \Rightarrow$ Get $-D$ OR E
- Resolving: A OR $-D$ with $-A \Rightarrow$ Get $-D$

Loop 2:

- Resolving: $-B$ OR C with $-C \Rightarrow$ Get $-B$
- Resolving: $-B$ OR C with $-C$ OR $E \Rightarrow$ Get $-B$ OR E
- Resolving: $-B$ OR C with A OR $B \Rightarrow$ Get A OR C
- Resolving: B OR D with $-D \Rightarrow$ Get B
- Resolving: B OR D with $-D$ OR $E \Rightarrow$ Get B OR E

- Resolving: B OR D with A OR $-B \Rightarrow$ Get A OR D
- Resolving: $-C$ with C OR $D \Rightarrow$ Get D
- Resolving: $-C$ OR E with C OR $D \Rightarrow$ Get D OR E
- Resolving: $-D$ with C OR $D \Rightarrow$ Get C
- Resolving: $-D$ OR E with C OR $D \Rightarrow$ Get C OR E
- Resolving: A OR $-B$ with A OR $B \Rightarrow$ Get A

Loop 3:

- Resolving: A OR $-C$ with C OR $E \Rightarrow$ Get A OR E
- Resolving: $-A$ OR E with $A \Rightarrow$ Get E
- Resolving: $-A$ with $A \Rightarrow$ Get {}

Output: We got A after run pl-resolution so KB entails α

```
7
-C
-D
A OR B
A OR -B
C OR D
-C OR E
-D OR E
11
A
B
-B
C
D
A OR C
A OR D
B OR E
-B OR E
C OR E
D OR E
3
{}
E
A OR E
YES
```

4.6 input06.txt

```
-R
6
P OR -Q
-P OR R OR S
Q
-P OR -R
Q OR P OR R
-Q OR S
```

Knowledge Base after adding negation of alpha:

- P OR $-Q$
- $-P$ OR R OR SR
- Q
- $-P$ OR $-R$
- P OR Q OR R
- $-Q$ OR S
- R

Loop 1:

- Resolving: P OR $-Q$ with $-P$ OR R OR $S \Rightarrow$ Get $-Q$ OR R OR S
- Resolving: P OR $-Q$ with $Q \Rightarrow$ Get P
- Resolving: P OR $-Q$ with $-P$ OR $-R \Rightarrow$ Get $-Q$ OR $-R$
- Resolving: P OR $-Q$ with P OR Q OR $R \Rightarrow$ Get P OR R
- Resolving: $-P$ OR R OR S with $-P$ OR $-R \Rightarrow$ Get $-P$ OR S
- Resolving: $-P$ OR R OR S with P OR Q OR $R \Rightarrow$ Get Q OR R OR S
- Resolving: Q with $-Q$ OR $S \Rightarrow$ Get S
- Resolving: $-P$ OR $-R$ with $R \Rightarrow$ Get $-P$
- Resolving: P OR Q OR R with $-Q$ OR $S \Rightarrow$ Get P OR R OR S

Loop 2:

- Resolving: $-P$ OR R OR S with $-Q$ OR $-R \Rightarrow$ Get $-P$ OR $-Q$ OR S
- Resolving: $-P$ OR R OR S with $P \Rightarrow$ Get R OR S
- Resolving: Q with $-Q$ OR $-R \Rightarrow$ Get $-R$

- Resolving: P OR $-Q$ with $-P \Rightarrow$ Get $-Q$
- Resolving: $-P$ OR $-R$ with Q OR R OR $S \Rightarrow$ Get $-P$ OR Q OR S
- Resolving: P OR Q OR R with $-P \Rightarrow$ Get Q OR R
- Resolving: $-P$ with $P \Rightarrow$ Get $\{\}$

Output: We got $-R$ after run pl-resolution so KB entails α

```
9
-P
P
S
P OR R
-P OR S
-Q OR -R
P OR R OR S
Q OR R OR S
-Q OR R OR S
7
{}
-Q
-R
Q OR R
R OR S
-P OR Q OR S
-P OR -Q OR S
YES
```

4.7 input07.txt

```
Q OR R
4
-P OR R
P OR S
R OR -S
Q OR -R
```

Knowledge Base after adding negation of alpha:

- $-P$ OR R
- P OR S
- R OR $-S$
- Q OR $-R$
- $-R$
- $-Q$

Loop 1:

- Resolving: $-P$ OR R with P OR $S \Rightarrow$ Get R OR S
- Resolving: $-P$ OR R with Q OR $-R \Rightarrow$ Get $-P$ OR Q
- Resolving: $-P$ OR R with $-R \Rightarrow$ Get $-P$
- Resolving: P OR S with R OR $-S \Rightarrow$ Get P OR R
- Resolving: R OR $-S$ with Q OR $-R \Rightarrow$ Get Q OR $-S$
- Resolving: R OR $-S$ with $-R \Rightarrow$ Get $-S$
- Resolving: Q OR $-R$ with $-Q \Rightarrow$ Get $-R$

Loop 2:

- Resolving: $-P$ OR R with P OR $R \Rightarrow$ Get R
- Resolving: P OR S with $-P \Rightarrow$ Get S
- Resolving: P OR S with $-P$ OR $Q \Rightarrow$ Get Q OR S
- Resolving: P OR S with $-S \Rightarrow$ Get P
- Resolving: P OR S with Q OR $-S \Rightarrow$ Get P OR Q
- Resolving: $-P$ OR Q with P OR $R \Rightarrow$ Get Q OR R

Loop 3:

- Resolving: Q OR $-R$ with Q OR $R \Rightarrow$ Get Q

- Resolving: $-R$ with $R \Rightarrow$ Get {}

Output: We got Q OR R after run pl-resolution so KB entails α

```
6
-P
-S
-P OR Q
P OR R
R OR S
Q OR -S
6
P
R
SR
P OR Q
Q OR R
Q OR S
2
{}
Q
YES
```

5 Evaluation of the Resolution Method for Propositional Logic

5.1 Advantages

- **Completeness:** The resolution method is complete for propositional logic, meaning that if a formula is unsatisfiable, the method will eventually derive an empty clause, confirming the unsatisfiability.
- **Simplicity:** The method is straightforward and based on a single inference rule, making it easier to implement compared to other logic inference methods.

5.2 Disadvantages

- **Exponential Growth:** The search space can grow exponentially as new resolvents are generated. This makes the method computationally expensive and can lead to inefficiency in practice, especially with large knowledge bases.
- **Redundancy**: The method may generate many redundant clauses that do not contribute to finding a solution, increasing the computational load without adding value.

6 Proposed Solution for Specific Problem

To address the challenges of inefficiency and redundancy in the resolution method, I propose the following enhancements:

- **Memoization:** Use memoization to store and reuse the results of previous resolutions, avoiding repeated calculations and speeding up the process.
- **Subsumption Checking:** Integrate subsumption checks to eliminate clauses that are subsumed by more general ones, reducing the number of clauses considered at each step.
- **Parallel Processing:** Leverage parallel processing to handle multiple clause pairs simultaneously, reducing overall computation time.

References

[1] Phungvhbui. (n.d.). GitHub - phungvhbui/AI-Lab-PL-Resolution: Implement PL-Resolution, Introduction to AI, HCMUS

[2] GeeksforGeeks. (2022, May 27). Resolution algorithm in artificial intelligence. GeeksforGeeks. https://www.geeksforgeeks.org/resolution-algorithm-in-artificial-intelligence/

[3] Fundamentals of Artificial Intelligence resources - Mr.Bui Duy Dang