THE UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY

Nguyễn Phương Nam - 21120504

LAB 02
LOGIC

AI FUNDAMENTALS DEPARTMENT
GENERAL PROGRAM

INSTRUCTOR

Nguyễn Ngọc Đức

Tp. Hồ Chí Minh, 11/2023

Reassurances

I declare that this is my own work. I have personally written the code sections. All references materials I have used are explicitly mentioned.

Thanks

I would like to thank Mr.Nguyễn Ngọc Đức for providing me the opportunity to delve into the details of the resolution algorithm and allowing me to design a simple Knowledge Base for my self.

Table of contents

Chapter 1

Resolution algorithm

1.1 Resolution rule (PL-Resolve) [1]

The resolution rule in propositional logic is a single valid inference rule that produces a new clause implied by two clauses containing complementary literals. A literal is a propositional variable or the negation of a propositional variable. Two literals are said to be complements if one is the negation of the other (in the following, $\neg c$ is taken to be the complement to c). The resulting clause contains all the literals that do not have complements. Formally:

$$\frac{p_1 \vee \cdots \vee p_n \vee c, \quad q_1 \vee \cdots \vee q_m \vee \neg c}{p_1 \vee \cdots \vee p_n \vee q_1 \vee \cdots \vee q_m}$$

1.2 Resolution algorithm [2]

To show that $KB \models \alpha$, we show that $(KB \wedge \neg\alpha)$ is unsatisfiable. First, $(KB \wedge \neg\alpha)$ is converted into CNF. Then, the resolution rule is applied to the resulting clauses. Each pair that contains complementary literals is resolved to produce a new clause, which is added to the set if it is not already present. The process continues until one of two things happens:

- There are no new clauses that can be added, in which case KB does not entail α; or
- Two clauses resolve to yield the empty clause, in which case KB entails α.

1.3 Pseudocode [2]

function PL-RESOLUTION(KB, α) **return** true of false

  inputs: KB, the knowledge base, a sentence in propositional logic

    α, the query, a sentence in propositional logic

  clauses $\leftarrow$ the set of clauses in the CNF representation of $(KB \wedge \neg\alpha)$

  new $\leftarrow \{\}$

  while true **do**

    for each pair of clauses C_i, C_j **in** *clauses* **do**

      resolvents $\leftarrow$ PL-RESOLVE(C_i, C_j)

      if *resolvents* contains the empty clause **then return** true

      new $\leftarrow$ *new* $\cup$ *resolvents*

    if *new* $\subseteq$ *clauses* **then return** false

    clauses $\leftarrow$ *clauses* $\cup$ *new*

Chapter 2

The advantages and disadvantages of resolution method

2.1 The advantages

- Completeness: The resolution method can be used to prove any valid theorem of propositional logic.
- Soundness: The resolution method can never prove a theorem that is not valid.
- Automation: The resolution method can be implemented in a computer program. This makes it a powerful tool for automated theorem proving.

2.2 The disadvantages

- Efficiency: The resolution method can be inefficient for some problems. This is because it may explore a large number of search paths before finding a proof.
- Explanations: The resolution method does not always provide explanations for its proofs. This can make it difficult to understand why a proof is valid.
- One thing I noticed is that the resolution algorithm repeats resolving previously resolved clauses again and again. This problem can easily be solved by only choosing Ci in clauses and choosing Cj in new (new in previous iteration)

2.3 Some improvements

- One heuristic is to focus on resolving clauses that contain literals that occur in many other clauses. This is because these clauses are more likely to be part of a proof.

- Another heuristic is to prefer resolving clauses that are smaller. This is because smaller clauses are easier to process and are more likely to lead to a shorter proof.

Chapter 3

Implementation

3.1 The data structures used in the program

- Each clause is a list of sorted literals. For example, the clause A OR B OR -C is represented as ["A", "B", "-C"].
- Knowledge base is a list of clauses. For example, the knowledge base (A OR C OR -B) AND (-A OR -C OR B) is represented as [["A", "-B", "C"], ["-A", "B", "-C"]].
- *output* is a 3d list. *output[i]* is a list of clauses generated in the i-th iteration.

3.2 The functions used in the program

- *read_input(filename)*: Reads the input file and returns the α statement and the knowledge base.
- *write_output(filename, output)*: uses *output* to write the output file. In each iteration, first it writes *len(output[i])* to indicate the number of clauses generated in the i-th iteration. Then, it writes each clause in output[i] in a new line.
- *PL_Resolve(clause1, clause2)*: using the resolution rule, it returns the resolvent of clause1 and clause2. If applying the resolution rule is failed, it returns None. If two clauses yield an empty clause, it returns ["{}"] to indicate α is entailed by the knowledge base.
- *PL_Resolution(α, KB)*:
  - First, it adds $\neg\alpha$ to the knowledge base.

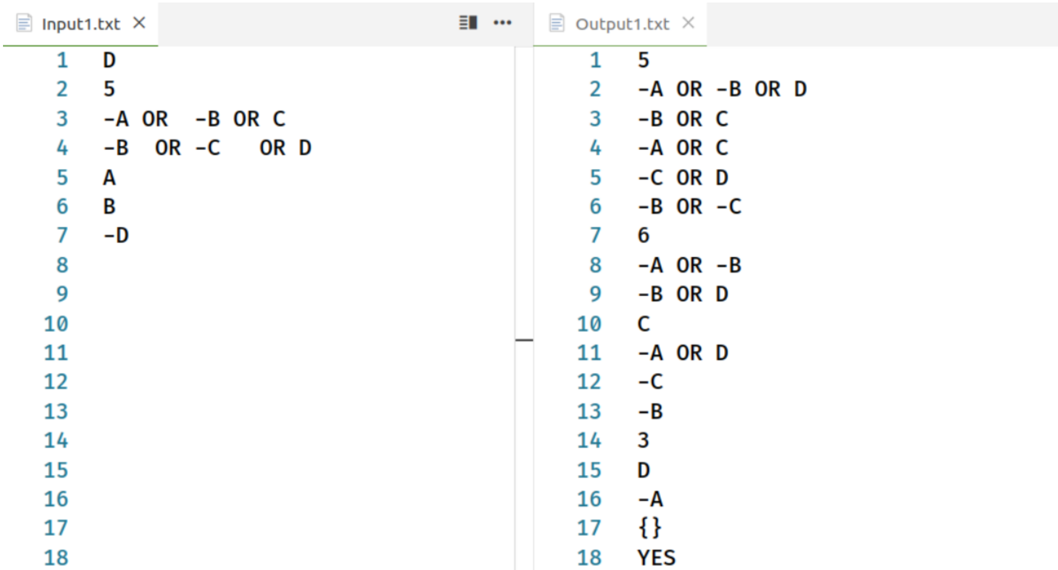
- Then, it iterates until it finds an empty clause or it cannot generate any new clauses.
- In each iteration, it generates new clauses by applying the resolution rule to all pairs of clauses in the knowledge base. All the new clauses are added to the knowledge base and captured in *output*.
- If it finds an empty clause, it returns True and *output*.
- If it cannot generate any new clauses, it returns False and *output*.

Chapter 4

Test cases

4.1 Test case 1

- Output:



Input1.txt

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

Output1.txt

```
5
-A OR -B OR D
-B OR C
-A OR C
-C OR D
-B OR -C
6
-A OR -B
-B OR D
C
-A OR D
-C
-B
3
D
-A
{}
YES
```

- Explanation:

```
1   -A OR  -B OR C        |
2   -B  OR -C   OR D      |
3   A                     |
4   B                     |
5   -D (negation of α)    |
6   ──────────────────────┼──────────────────────
7   -A OR -B OR D         | (1 resolves with 2)
8   -B OR C               | (1 resolves with 3)
9   -A OR C               | (1 resolves with 4)
10  -C OR D               | (2 resolves with 4)
11  -B OR -C              | (2 resolves with 5)
12  ──────────────────────┼──────────────────────
13  -A OR -B              | (1 resolves with 11)
14  -B OR D               | (2 resolves with 8)
15  C                     | (3 resolves with 9)
16  -A OR D               | (4 resolves with 7)
17  -C                    | (4 resolves with 11)
18  -B                    | (8 resolves with 11)
19  ──────────────────────┼──────────────────────
20  D                     | (3 resolves with 16)
21  -A                    | (4 resolves with 13)
22  {}                    | (4 resolves with 18)
23  ──────────────────────┼──────────────────────
24  YES                   |
```

4.2 Test case 2

- Output:

Input2.txt

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

Output2.txt

```
3
-P OR Q
-Q
P
5
-P OR -R
Q OR -R
-P OR R
Q OR R
{}
YES
```

- Explanation:

```
1   -Q OR -R       |
2   P OR -R        |
3   Q OR -P OR R   |
4   -Q OR P        |
5   -P             | (negation of α)
6   Q              | (negation of α)
7   -R             | (negation of α)
8   ---------------+---------------
9   -P OR Q        | (3 resolves with 7)
10  -Q             | (4 resolves with 5)
11  P              | (4 resolves with 6)
12  ---------------+---------------
13  -P OR -R       | (1 resolves with 9)
14  Q OR -R        | (2 resolves with 9)
15  -P OR R        | (3 resolves with 10)
16  Q OR R         | (3 resolves with 11)
17  {}             | (5 resolves with 11)
18  YES
```

4.3 Test case 3

- Output:

Input3.txt

```
D OR E
3
-A OR B OR -D
-C OR D OR E
-C OR A
```

Output3.txt

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

- Explanation:

```
1   -A OR B OR -D          |
2   -C OR D OR E           |
3   -C OR A                |
4   -D                     | (negation of α)
5   -E                     | (negation of α)
6   -----------------------+-------------------
7   -A OR B OR -C OR E     | (1 resolves with 2)
8   B OR -C OR -D          | (1 resolves with 3)
9   -C OR E                | (2 resolves with 4)
10  -C OR D                | (2 resolves with 5)
11  -----------------------+-------------------
12  -A OR B OR -C          | (1 resolves with 10)
13  B OR -C OR E           | (2 resolves with 8)
14  -C                     | (4 resolves with 10)
15  B OR -C                | (8 resolves with 10)
16  -----------------------+-------------------
17  NO                     |
```

4.4 Test case 4

- Output:

Input4.txt

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

Output4.txt

```
5
-A OR -C
-A OR B
-A OR C OR -D
-A OR -D
-B OR -D
3
-A OR -C OR -D
-A OR C
-A OR -B
1
-A OR -B OR -D
0
NO
```

- Explanation:

```
1   -A OR B OR -C     |
2   -D OR -A OR B     |
3   -C OR -D          |
4   -A OR D           |
5   -B OR C           |
6   -A                | (negation of α)
7   -B                | (negation of α)
8   -C                | (negation of α)
9   -D                | (negation of α)
10  ------------------+--------------------
11  -A OR -C          | (1 resolves with 7)
12  -A OR B           | (2 resolves with 4)
13  -A OR C OR -D     | (2 resolves with 5)
14  -A OR -D          | (2 resolves with 7)
15  -B OR -D          | (3 resolves with 5)
16  ------------------+--------------------
17  -A OR -C OR -D    | (1 resolves with 15)
18  -A OR C           | (4 resolves with 13)
19  -A OR -B          | (4 resolves with 15)
20  ------------------+--------------------
21  -A OR -B OR -D    | (5 resolves with 17)
22  ------------------+--------------------
23  NO                |
```

4.5 Test case 5

- Output:

Input5.txt

```
P OR S
5
-P OR Q OR R
-R OR S OR -P
-Q OR -S OR P
-P OR -Q OR -R
```

Output5.txt

```
4
-P OR Q OR S
-P OR -R
-Q OR -R OR -S
-Q OR -S
2
-P OR Q
-P OR R OR -S
4
-Q OR R OR -S
-P OR -Q OR -S
-P OR -S
-P OR -R OR -S
2
-P OR Q OR -S
-P OR Q OR -R
0
NO
```

- Explanation:

```
1   -P OR Q OR R     |
2   -R OR S OR -P    |
3   -Q OR -S OR P    |
4   -P OR -Q OR -R   |
5   -P               | (negation of α)
6   -S               | (negation of α)
7   -----------------+---------------------
8   -P OR Q OR S     | (1 resolves with 2)
9   -P OR -R         | (2 resolves with 6)
10  -Q OR -R OR -S   | (3 resolves with 4)
11  -Q OR -S         | (3 resolves with 5)
12  -----------------+---------------------
13  -P OR Q           | (1 resolves with 9)
14  -P OR R OR -S    | (1 resolves with 11)
15  -----------------+---------------------
16  -Q OR R OR -S    | (3 resolves with 14)
17  -P OR -Q OR -S   | (4 resolves with 14)
18  -P OR -S         | (9 resolves with 14)
19  -P OR -R OR -S   | (10 resolves with 13)
20  -----------------+---------------------
21  -P OR Q OR -S    | (1 resolves with 19)
22  -P OR Q OR -R    | (8 resolves with 19)
23  -----------------+---------------------
24  NO               |
```

References

Bibliography

[1] *Resolution (logic)*. URL: `https://en.wikipedia.org/wiki/Resolution_(logic)` (visited on 11/22/2023).

[2] Stuart J. Russell and Peter Norvig. *Artificial Intelligence A Modern Approach*. Pearson, 2020.