VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY

UNIVERSITY OF SCIENCE

FACULTY OF INFORMATION TECHNOLOGY

LÊ CÔNG ĐẮT

LAB 02

LOGIC

Module: Introduction to Artificial Intelligence

Ho Chi Minh City – 2023

VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY

UNIVERSITY OF SCIENCE

FACULTY OF INFORMATION TECHNOLOGY

20120454 - LÊ CÔNG ĐẮT

LAB 02

| Topic |

LOGIC

| Instructor |

Nguyen Ngoc Duc

Module: Introduction to Artificial Intelligence

Ho Chi Minh City - 2023

TABLE OF CONTENTS

1. Propositional logic [1]

1.1. Syntax

The syntax of propositional logic defines the allowable sentences. The atomic sentences consist of a single proposition symbol. Each such symbol stands for a proposition that can be true or false. We use symbols that start with an uppercase letter and may contain other letters or subscripts, for example: P, Q, R, $W_{1,3}$ and FacingEast. The names are arbitrary but are often chosen to have some mnemonic value—we use $W_{1,3}$ to stand for the proposition that the wumpus is in [1,3]. (Remember that symbols such as $W_{1,3}$ are atomic, i.e., W, 1, and 3 are not meaningful parts of the symbol). There are two proposition symbols with fixed meanings: True is the always-true proposition and False is the always-false proposition. Complex sentences are constructed from simpler sentences, using parentheses and operators called logical connectives. There are five connectives in common use:

¬ (not). A sentence such as $\neg W_{1,3}$ is called the negation of $W_{1,3}$. A literal is either an atomic sentence (a positive literal) or a negated atomic sentence (a negative literal).

∧ (and). A sentence whose main connective is ∧, such as $W_{1,3} \wedge P_{3,1}$, is called a conjunction; its parts are the conjuncts. (The ∧ looks like an “A” for “And.”)

∨ (or). A sentence whose main connective is ∨, such as $(W_{1,3} \wedge P_{3,1}) \vee W_{2,2}$, is a disjunction; its parts are disjuncts—in this example, $(W_{1,3} \wedge P_{3,1})$ and $W_{2,2}$.

⇒ (implies). A sentence such as $(W_{1,3} \wedge P_{3,1}) \Rightarrow \neg W_{2,2}$ is called an implication (or conditional). Its premise or antecedent is $(W_{1,3} \wedge P_{3,1})$ and its conclusion or consequentis $\neg W_{2,2}$. Implications are also known as rules or if–then statements. The implication symbol is sometimes written in other books as ⊃ or →.

⇔ (if and only if). The sentence $W_{1,3} \Leftrightarrow \neg W_{2,2}$ is a biconditional.

$$
\begin{aligned}
\textit{Sentence} &\rightarrow \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} &\rightarrow \textit{True} \mid \textit{False} \mid P \mid Q \mid R \mid \ldots \\
\textit{ComplexSentence} &\rightarrow (\textit{Sentence}) \\
&\mid \neg \textit{Sentence} \\
&\mid \textit{Sentence} \wedge \textit{Sentence} \\
&\mid \textit{Sentence} \vee \textit{Sentence} \\
&\mid \textit{Sentence} \Rightarrow \textit{Sentence} \\
&\mid \textit{Sentence} \Leftrightarrow \textit{Sentence} \\
\text{OPERATOR PRECEDENCE} &: \neg, \wedge, \vee, \Rightarrow, \Leftrightarrow
\end{aligned}
$$

Figure 7.7 A BNF (Backus–Naur Form) grammar of sentences in propositional logic, along with operator precedences, from highest to lowest.

1.2. Semantics

Having specified the syntax of propositional logic, we now specify its semantics. The semantics defines the rules for determining the truth of a sentence with respect to a particular model. In propositional logic, a model simply sets the truth value—true or false—for every proposition symbol. For example, if the sentences in the knowledge base make use of the proposition symbols $P_{1,2}$, $P_{2,2}$, and $P_{3,1}$, then one possible model is m_1 = {$P_{1,2}$ = false, $P_{2,2}$ = false, $P_{3,1}$ = true}.

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

Figure 7.8 Truth tables for the five logical connectives. To use the table to compute, for example, the value of $P \vee Q$ when P is true and Q is false, first look on the left for the row where P is *true* and Q is *false* (the third row). Then look in that row under the $P \vee Q$ column to see the result: *true*.

1.3. Conjunctive normal form (CNF)

The resolution rule applies only to clauses (that is, disjunctions of literals), so it would seem to be relevant only to knowledge bases and queries consisting of clauses. How, then, can it lead to a complete inference procedure for all of propositional logic? The answer is that every sentence of propositional logic is logically equivalent to a conjunction of clauses.

A sentence expressed as a conjunction of clauses is said to be in conjunctive normal form or CNF.

$$
\begin{aligned}
CNFSentence &\rightarrow Clause_1 \wedge \cdots \wedge Clause_n \\
Clause &\rightarrow Literal_1 \vee \cdots \vee Literal_m \\
Fact &\rightarrow Symbol \\
Literal &\rightarrow Symbol \mid \neg Symbol \\
Symbol &\rightarrow P \mid Q \mid R \mid \ldots \\
HornClauseForm &\rightarrow DefiniteClauseForm \mid GoalClauseForm \\
DefiniteClauseForm &\rightarrow Fact \mid (Symbol_1 \wedge \cdots \wedge Symbol_l) \Rightarrow Symbol \\
GoalClauseForm &\rightarrow (Symbol_1 \wedge \cdots \wedge Symbol_l) \Rightarrow False
\end{aligned}
$$

Figure 7.12 A grammar for conjunctive normal form, Horn clauses, and definite clauses. A CNF clause such as $\neg A \vee \neg B \vee C$ can be written in definite clause form as $A \wedge B \Rightarrow C$.

1.4. A resolution algorithm

To show that KB |= α, we show that (KB ∧ ¬α) is unsatisfiable. We do this by proving a contradiction. A resolution algorithm is shown in Figure 7.13. First, (KB∧ ¬α) 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.

```
function PL-RESOLUTION(KB, α) returns true or false
  inputs: KB, the knowledge base, a sentence in propositional logic
          α, the query, a sentence in propositional logic

  clauses ← the set of clauses in the CNF representation of KB ∧ ¬α
  new ← { }
  while true do
    for each pair of clauses Ci, Cj in clauses do
      resolvents ← PL-RESOLVE(Ci, Cj)
      if resolvents contains the empty clause then return true
      new ← new ∪ resolvents
    if new ⊆ clauses then return false
    clauses ← clauses ∪ new
```

Figure 7.13 A simple resolution algorithm for propositional logic. PL-RESOLVE returns the set of all possible clauses obtained by resolving its two inputs.

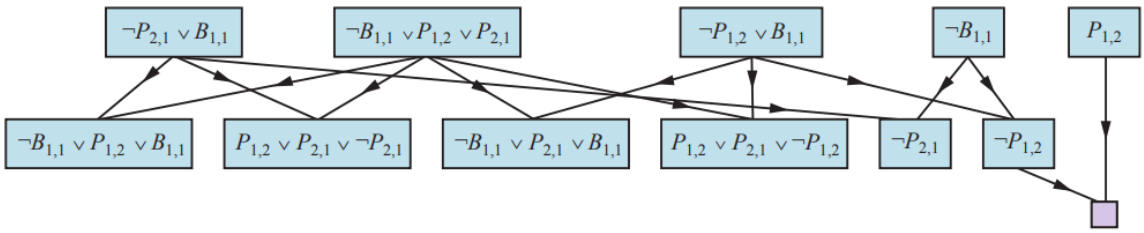

Figure 7.14 Partial application of PL-RESOLUTION to a simple inference in the wumpus world to prove the query $\neg P_{1,2}$. Each of the leftmost four clauses in the top row is paired with each of the other three, and the resolution rule is applied to yield the clauses on the bottom row. We see that the third and fourth clauses on the top row combine to yield the clause $\neg P_{1,2}$, which is then resolved with $P_{1,2}$ to yield the empty clause, meaning that the query is proven.

2. Evaluation on resolution method for propositional logic

2.1. Advantages

- Completeness: always terminates and returns result
  - + There are no new clauses that can be added, in which case knowledge base KB does not entail the statement α.
  - + Two clauses resolve to yield the empty clause, in which case knowledge base KB entails the statement α.
- Rightness: The returned result is always a consequence of the initial knowledge base

2.2. Disadvantages

- Generating all possible pairs of propositions in the knowledge base leads to a large number of pairs of propositions that need to be resolved.
- The system generates unreasonable pairs of propositions, causing the proposition after resolution to have more literals than the original proposition.
- Lack of orientation mechanism: the generated propositions may not be related to the proposition that needs to be denied.

2.3. Improved solution

- Prioritizing the resolution of propositions with fewer literals helps increase the likelihood of obtaining shorter post-resolution propositions.
- Prioritizing the resolution of pairs of propositions related to the proposition that needs to be denied or propositions inferred from the proposition that needs to be denied.

3. Test cases

3.1. Test case 1

input1.txt	output1.txt	Note
-A 3 -B OR C -A OR -C -A OR B	4 B -C -A OR -B -A OR C 3 -A -B C 1 {} YES	 (-A OR B) resolve with (negative of -A) (-A OR -C) resolve with (negative of -A) (-A OR -C) resolve with (-B OR C) (-A OR B) resolve with (-B OR C) (B) resolve with (-A OR -B) (-A OR -B) resolve with (negative of -A) (-A OR C) resolve with (negative of -A) (-A) resolve with (negative of -A) KB entails α because two clauses resolve to yield the empty clause

3.2. Test case 2

input2.txt	output2.txt	Note
C 4 -A -B OR C A OR B OR -D A OR B OR C OR D	7 -B B OR -D A OR B OR C A OR B OR D A OR C OR D A OR C OR -D B OR C OR D 9 -D A OR B A OR C A OR D A OR -D B OR C B OR D C OR D C OR -D 4 A B C D 1 {} YES	 (-B OR C) resolve with (negative of C) (A OR B OR -D) resolve with (-A) (A OR B OR -D) resolve with (A OR B OR C OR D) (A OR B OR C OR D) resolve with (-C) (A OR B OR C OR D) resolve with (-B OR C) (A OR B OR -D) resolve with (-B OR C) (A OR B OR C OR D) resolve with (-A) (-B) resolve with (B OR -D) (A OR B OR C) resolve with (negative of C) (A OR B OR C) resolve with (-B OR C) (A OR C OR D) resolve with (-C) (A OR B OR -D) resolve with (-B) (A OR B OR C) resolve with (-A) (A OR B OR D) resolve with (-A) (A OR C OR D) resolve with (-A) (-B OR C) resolve with (B OR -D) (A OR B) resolve with (-A) (A OR C) resolve with (-A) (A OR C) resolve with (negative of C) (A OR D) resolve with (-A) (A) resolve with (-A) KB entails α because two clauses resolve to yield

		the empty clause

3.3. Test case 3

input3.txt	output3.txt	Note
D 4 -A OR C A -B OR -C OR D B	4 C -B OR -C -C OR D -A OR -B OR D 6 -B -C D -A OR -B -A OR D -B OR D 2 {} -A YES	 (-A OR C) resolve with (A) (-B OR -C OR D) resolve with (negative of D) (-B OR -C OR D) resolve with (B) (-A OR C) resolve with (-B OR -C OR D) (C) resolve with (-B OR -C) (B) resolve with (-B OR -C) (C) resolve with (-C OR D) (-A OR C) resolve with (-B OR -C) (B) resolve with (-A OR -B OR D) (A) resolve with (-A OR -B OR D) (B) resolve with (-B) (C) resolve with (-A OR C) KB entails $\boldsymbol{\alpha}$ because two clauses resolve to yield the empty clause

3.4. Test case 4

input4.txt	output4.txt	Note
-C 5 A -A OR B -B OR C OR D -D OR E -E	4 B -D -A OR C OR D -B OR C OR E 5 -A OR C -B OR C C OR D C OR E -A OR C OR E 0 NO	 (A) resolve with (-A OR B) (-D OR E) resolve with (-E) (-A OR B) resolve with (-B OR C OR D) (-B OR C OR D) resolve with (-D OR E) (-D) resolve with (-A OR C OR D) (-B OR C OR E) resolve with (-E) (-A OR C OR D) resolve with (A) (-B OR C OR E) resolve with (B) (-A OR C OR D) resolve with (-D OR E) KB DOES NOT entail α because there are no new clauses that can be added and there are no empty clauses found

3.5. Test case 5

input5.txt	output5.txt	Note
B 4	5 A OR D	 (A OR B OR D) resolve with (negative of B)

A OR B OR C OR D A OR B OR D -B OR -C -A	B OR D A OR C OR D A OR -C OR D B OR C OR D 4 D C OR D -C OR D A OR -B OR D 1 -B OR D 0 NO	(A OR B OR D) resolve with (-A) (A OR B OR C OR D) resolve with (negative of B) (A OR B OR D) resolve with (-B OR -C) (A OR B OR C OR D) resolve with (-A) (A OR D) resolve with (-A) (A OR C OR D) resolve with (-A) (A OR -C OR D) resolve with (-A) (A OR C OR D) resolve with (-B OR -C) (A OR -B OR D) resolve with (-A) KB DOES NOT entail α because there are no new clauses that can be added and there are no empty clauses found

4. Evalutions

STT	Description	Ratio	Evalution
1	Read input data and store in suitable data structure	0.5	100%
2	Implementation of resolution method	1.0	100%
3	Inference process and results	2.5	100%
5	Testcases, report, evaluations,	1.0	100%

REFERENCES

[1] Russell, S. J., & Norvig, P. (2020). Artificial intelligence: A modern approach (4th ed.). Pearson Education Limited.

[2] Berkeley University of California. (2023, April 12). "*Artificial Intelligence: A Modern Approach, 4th US ed. by Stuart Russell and Peter Norvig*". Berkeley University of California. http://aima.cs.berkeley.edu/

[3] Berkeley University of California. (2023, April 12). "*AIMA Python file: logic.py*". Berkeley University of California. http://aima.cs.berkeley.edu/python/logic.html