Propositional Theorem Proving will be discussed under the following headings:

1. Inferences and Proofs
2. Proof by Resolution
3. Horn Clauses and definite Clauses
4. Forward and backward chaining

Resolving Clauses

Clause 1: (PVQVR)
Clause 2:(¬PV¬QVS)
To resolve these clauses, we look for complementary literals. In this case, **P and ¬P** are complementary.
So, we can resolve these two clauses by removing the complementary literals and
combining the remaining literals: **(PVQVR) and ((¬PV¬QVS)**

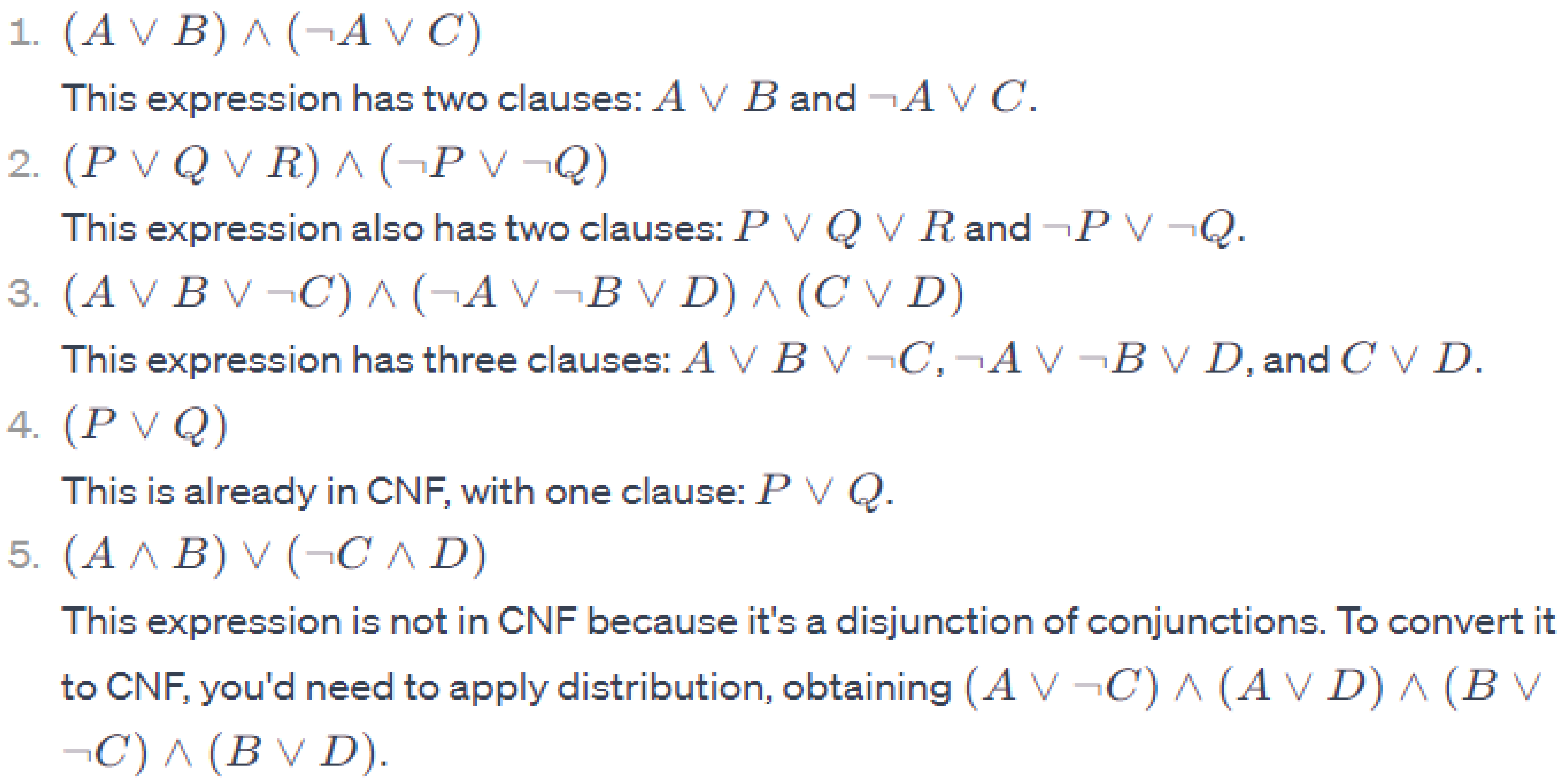
Resolving P and ¬P gives**: (QVR)V(¬QVS)**

This is the resolvent.

Conjunctive Normal Form

A formula is in CNF if it is a conjunction (AND) of clauses, where each clause is a disjunction (OR) of literals.

CNF Examples



1. $(A \vee B) \wedge (\neg A \vee C)$

   This expression has two clauses: $A \vee B$ and $\neg A \vee C$.
2. $(P \vee Q \vee R) \wedge (\neg P \vee \neg Q)$

   This expression also has two clauses: $P \vee Q \vee R$ and $\neg P \vee \neg Q$.
3. $(A \vee B \vee \neg C) \wedge (\neg A \vee \neg B \vee D) \wedge (C \vee D)$

   This expression has three clauses: $A \vee B \vee \neg C$, $\neg A \vee \neg B \vee D$, and $C \vee D$.
4. $(P \vee Q)$

   This is already in CNF, with one clause: $P \vee Q$.
5. $(A \wedge B) \vee (\neg C \wedge D)$

   This expression is not in CNF because it's a disjunction of conjunctions. To convert it to CNF, you'd need to apply distribution, obtaining $(A \vee \neg C) \wedge (A \vee D) \wedge (B \vee \neg C) \wedge (B \vee D)$.

Steps to Convert a Formula to CNF:

1. **Eliminate Biconditionals (⇔):** Replace each biconditional (↔) with an equivalent expression in terms of conjunction (∧), disjunction (∨), and negation (¬).
    - Example: Replace A ⇔ B with (A⇒B) ∧(B⇒A)

2. **Eliminate Implications** (⇒): Replace each implication (⇒) with an equivalent expression using conjunction and negation.
    - Example: Replace A⇒B with ¬A V B

3. **Move Negations Inward** (¬): Apply De Morgan's laws and distribute negations inward to literals.
    - ¬(¬A) ≡ A
    - ¬(A V B) ≡(¬A ∧ ¬B)
    - ¬(A∧ B) ≡ (¬A V ¬B)

4. **Distribute Disjunctions Over Conjunctions**: Apply the distributive law to ensure that disjunctions are only over literals or conjunctions of literals. Suppose we have the following formula: **H=(P∧Q)V(R∧S∧T)** .
Now, let's distribute disjunctions over conjunctions:
H=(PV(R∧S∧T))∧(QV(R∧S∧T))

Proof by Resolution

Unit Resolution and Complete Resolution

Mathematically, the Unit Resolution rule can be expressed as:

$$\frac{P \vee Q, \neg P \vee R}{Q \vee R} \text{ (Unit Resolution)}$$

Here PVQ and ¬PVR are clauses, and after applying Unit Resolution, QVR is the simplified clause.

Complete Resolution

$$\frac{C_1\colon P_1 \vee Q_1 \vee \ldots \vee L,\ C_2\colon \neg P_1 \vee R_1 \vee \ldots \vee M}{C\colon Q_1 \vee \ldots \vee L \vee R_1 \vee \ldots \vee M}$$

Proof By Resolution Process

1. **Initial Set of Clauses (Knowledge Base**)
2. **Negate the Conclusion**:
3. **Apply Resolution**
4. **Continue Resolving**
5. **Conclusion**
6. **Termination**

1. Initial Set of Clauses (Knowledge Base)

- Begin with a set of clauses representing the knowledge base in CNF. These clauses are typically obtained from logical statements or axioms.

2. Negate the Conclusion

- To prove a statement (conclusion), negate it. This negation is added to the set of clauses in CNF.

3. Apply Resolution

- Apply the resolution rule iteratively to the set of clauses. The resolution rule involves selecting two clauses that contain complementary literals (a literal and its negation). By resolving these clauses, a new clause is generated.

4.Continue Resolving

- Repeat the resolution process until either:
- The empty clause (□) is derived, indicating unsatisfiability.
- No further resolutions are possible, and the set of clauses remains unchanged, indicating satisfiability.

5. Conclusion

- If the empty clause is derived, the original set of clauses is unsatisfiable, and the negated statement is proven.
- If no further resolutions are possible and the set of clauses remains, then the original set of clauses is satisfiable, and the negated statement is not proven.

6. Termination

- The proof by resolution terminates when either the unsatisfiability is established, or it is determined that no further resolutions can lead to unsatisfiability.

Example : Let's consider a simplified example of a knowledge base for the Wumpus World scenario and demonstrate proof by resolution to establish the unsatisfiability of a certain statement.
In Wumpus World**, an agent explores a grid containing** a Wumpus (a monster), pits, and gold. Apply the resolution to prove P[1,2].

Knowledge Base (KB)

1. **$W[1,1] \lor P[1,2]$**
2. **$\neg W[1,1] \lor \neg P[1,2]$**
3. **$B[1,2] \Rightarrow P[1,2]$**
4. **$\neg B[1,2] \Rightarrow \neg P[1,2]$**

Convert the Knowledge Base (KB) into CNF

1. **W[1,1] ∨ P[1,2]**	1. **W[1,1] ∨ P[1,2]**
2. **¬W[1,1]∨¬P[1,2]**	2. **¬W[1,1]∨¬P[1,2]**
3. **B[1,2]⇒P[1,2]**	3. **¬ B[1,2] ∨ P[1,2]**
4. **¬B[1,2]⇒¬P[1,2]**	4. **B[1,2] ∨ ¬P[1,2]**

Negated Conclusion: Let's say we want to prove the negation of the statement: **¬PitIn[1,2]**

Apply Resolution:

1. **W[1,1] V P[1,2] , ¬P[1,2] resolves into W[1,1]**
2. **¬W[1,1]V¬P[1,2], W[1,1] resolves into ¬P[1,2]**
3. **¬B[1,2] V P[1,2] , ¬P[1,2] resolves into ¬B[1,2]**
4. **B[1,2] V¬P[1,2], ¬B[1,2] resolves into ¬P[1,2]**

Applying resolution, we end up with: **¬P[1,2]**

Which is not empty and also there is not further any clauses to continue.

This gives conclusion that our negation conclusion is **False** and **P[1,2]**

is true for the given knowledge base.

Premises:

p
$(p \wedge q) \Rightarrow r$
$(s \vee t) \Rightarrow q$

t

$\longrightarrow$

p
$\neg p \vee \neg q \vee r$
$\neg s \vee q$
$\neg t \vee q$
t

CNF

A resolution proof of r:

$$\begin{tikzcd}
\boxed{\neg p \vee \neg q \vee r} \arrow[dr] & & \boxed{\neg r} \arrow[dl] & & & \\
& \boxed{\neg p \vee \neg q} \arrow[dr] & & \boxed{p} \arrow[dl] & & \\
& & \boxed{\neg q} \arrow[dr] & & \boxed{\neg t \vee q} \arrow[dl] & \\
& & & \boxed{\neg t} \arrow[dr] & & \boxed{t} \arrow[dl] \\
& & & & \boxed{} &
\end{tikzcd}$$

3. Horn Clauses and definite Clauses

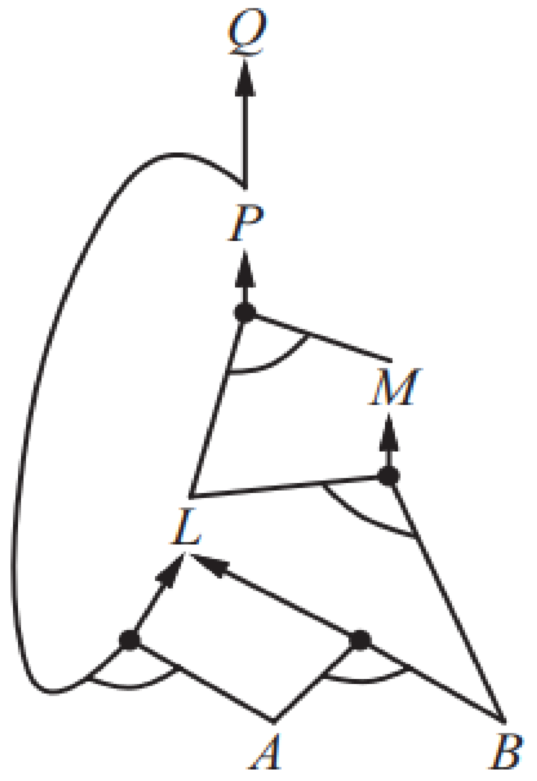
- **Definite Clause**: A definite clause is a specific form of a Horn clause where there is **exactly one positive literal in the head**. The general form of a definite clause is H←B1, B2 ,…,Bn , where H is the positive literal (head), and B1 ,B2 ,…,Bn are the negative literals or atoms (body).
- **Horn Clause:A Horn clause is a special type of logical clause that is a disjunction of literals, with at most one positive (non-negated) literal. In other words, a Horn clause is of the form H←B1 ,B2 ,…,Bn , where H is the positive literal (head), and B1 ,B2 ,…,Bn are the negative literals or atoms (body).**

A grammar for conjunctive normal form

$$\begin{array}{rcl}
\mathit{CNFSentence} & \rightarrow & \mathit{Clause}_1 \wedge \cdots \wedge \mathit{Clause}_n \\
\mathit{Clause} & \rightarrow & \mathit{Literal}_1 \vee \cdots \vee \mathit{Literal}_m \\
\mathit{Literal} & \rightarrow & \mathit{Symbol} \mid \neg \mathit{Symbol} \\
\mathit{Symbol} & \rightarrow & P \mid Q \mid R \mid \ldots \\
\mathit{HornClauseForm} & \rightarrow & \mathit{DefiniteClauseForm} \mid \mathit{GoalClauseForm} \\
\mathit{DefiniteClauseForm} & \rightarrow & (\mathit{Symbol}_1 \wedge \cdots \wedge \mathit{Symbol}_l) \Rightarrow \mathit{Symbol} \\
\mathit{GoalClauseForm} & \rightarrow & (\mathit{Symbol}_1 \wedge \cdots \wedge \mathit{Symbol}_l) \Rightarrow \mathit{False}
\end{array}$$

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

(a)



(b)

Figure 7.16 (a) A set of Horn clauses. (b) The corresponding AND–OR graph.

4. Forward and Backward Chaining

- **Forward chaining** is a reasoning strategy that starts with known facts in the knowledge base and propagates inferences forward until the desired query or goal is reached.
- **Backward chaining** is a reasoning strategy that works backward from the query or goal. It finds implications in the knowledge base whose conclusion is the query and then recursively checks if the premises of those implications can be proved true.

End of Module3