Proof Using Resolution

Outline

I. Rule of resolution

II. Conjunctive normal forms

III. Resolution refutation

IV. Horn clauses

* Figures are from the textbook site unless the source is specifically cited.
An inference algorithm $i$ is

- **sound** if $\text{KB} \models \alpha$ whenever $\text{KB} \vdash_i \alpha$
- **complete** if $\text{KB} \vdash_i \alpha$ whenever $\text{KB} \models \alpha$
I. Resolution

An inference algorithm $i$ is

- **sound** if $KB \models \alpha$ whenever $KB \vdash_i \alpha$
- **complete** if $KB \vdash_i \alpha$ whenever $KB \models \alpha$

- Inference rules covered so far are sound.
- The inference algorithms using them may not be complete.
I. Resolution

An inference algorithm $i$ is

- **sound** if $KB \models \alpha$ whenever $KB \vdash_i \alpha$
- **complete** if $KB \vdash_i \alpha$ whenever $KB \models \alpha$

- Inference rules covered so far are sound.
- The inference algorithms using them may not be complete.

**resolution** + a complete search algorithm = a complete inference algorithm
I. Resolution

An inference algorithm $i$ is

- **sound** if $KB \models \alpha$ whenever $KB \vdash_i \alpha$

- **complete** if $KB \vdash_i \alpha$ whenever $KB \models \alpha$

- Inference rules covered so far are sound.

- The inference algorithms using them may not be complete.

\[ \text{resolution} + \text{a complete search algorithm} = \text{a complete inference algorithm} \]

| single inference rule |
Wumpus World Revisited

Rules

KB:

\( R_1: \neg P_{1,1} \)

\( R_2: B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \)

\( R_3: B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1}) \)

\( R_4: \neg B_{1,1} \)

\( R_5: B_{2,1} \)

\[
\begin{align*}
R_2: B_{1,1} & \iff (P_{1,2} \lor P_{2,1}) \\
R_6: (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) & \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}) \\
R_7: (P_{1,2} \lor P_{2,1}) & \Rightarrow B_{1,1} \\
R_4: \neg B_{1,1} & \\
R_9: \neg B_{1,1} & \Rightarrow \neg (P_{1,2} \lor P_{2,1}) \\
R_9: \neg (P_{1,2} \lor P_{2,1}) & \\
R_{10}: \neg P_{1,2} \land \neg P_{2,1} & \\
R_{11}: \neg P_{1,2} & \\
\end{align*}
\]

- biconditional elimination
- and-elimination
- logical equivalence
- modus ponens
- De Morgan's rule
- and-elimination
Wumpus World Revisited

Agent: [1,1] → [2,1] → [1,1]

KB:

\[
R_1: \neg P_{1,1} \\
R_2: B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \\
R_3: B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1}) \\
R_4: \neg B_{1,1} \\
R_5: B_{2,1}
\]

Rules

\[
R_2: B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \\
R_6: (B_{1,1} \implies (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \implies B_{1,1}) \\
R_7: (P_{1,2} \lor P_{2,1}) \implies B_{1,1} \\
R_4: \neg B_{1,1} \\
R_9: \neg B_{1,1} \implies \neg (P_{1,2} \lor P_{2,1}) \\
R_8: \neg (P_{1,2} \lor P_{2,1}) \\
R_{10}: \neg P_{1,2} \land \neg P_{2,1} \\
R_{11}: \neg P_{1,2}
\]

- biconditional elimination
- and-elimination
- logical equivalence
- modus ponens
- De Morgan's rule
- and-elimination
Wumpus World Revisited

KB:

\[ R_1: \neg P_{1,1} \]
\[ R_2: B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \]
\[ R_3: B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1}) \]
\[ R_4: \neg B_{1,1} \]
\[ R_5: B_{2,1} \]

Rules

\[ R_6: (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}) \]  
\text{biconditional elimination}

\[ R_7: (P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1} \]  
\text{and-elimination}

\[ R_8: \neg B_{1,1} \Rightarrow \neg(P_{1,2} \lor P_{2,1}) \]  
\[ R_9: \neg(P_{1,2} \lor P_{2,1}) \]  
\text{logical equivalence}

\[ R_{10}: \neg P_{1,2} \land \neg P_{2,1} \]  
\text{modus ponens}

\[ R_{11}: \neg P_{1,2} \]  
\text{De Morgan's rule}

\[ R_{12}: \neg P_{1,2} \land \neg P_{2,1} \]  
\text{and-elimination}

Added to KB via inferences
<table>
<thead>
<tr>
<th></th>
<th>1,1</th>
<th>2,1</th>
<th>3,1</th>
<th>4,1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,1</td>
<td>V OK</td>
<td>B V</td>
<td>P!</td>
<td>4,1</td>
</tr>
<tr>
<td>1,2</td>
<td>A S OK</td>
<td>2,2</td>
<td>3,2</td>
<td>4,2</td>
</tr>
<tr>
<td>1,3</td>
<td>W!</td>
<td>2,3</td>
<td>3,3</td>
<td>4,3</td>
</tr>
<tr>
<td>1,4</td>
<td>2,4</td>
<td>3,4</td>
<td>4,4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>1,4</td>
<td>2,4</td>
<td>3,4</td>
<td>4,4</td>
<td></td>
</tr>
<tr>
<td>1,3</td>
<td>2,3</td>
<td>3,3</td>
<td>4,3</td>
<td></td>
</tr>
<tr>
<td>1,2</td>
<td>2,2</td>
<td>3,2</td>
<td>4,2</td>
<td></td>
</tr>
<tr>
<td>1,1</td>
<td>2,1</td>
<td>3,1</td>
<td>4,1</td>
<td></td>
</tr>
</tbody>
</table>

The image contains a table with some annotations. The table is likely related to a game or a puzzle, with symbols and arrows indicating specific moves or outcomes. The annotations include 'W!', 'OK', 'B', and 'P!', which might represent different actions or states in the game. The table is not fully labeled, but it seems to be part of a larger context that is not provided in the image.
[1, 1] → [1, 2]: stench but no breeze
[1,1] → [1,2]: stench but no breeze

Add to KB:

\[ R_{11}: \neg B_{1,2} \]
\[ R_{12}: B_{1,2} \iff (P_{1,1} \lor P_{2,2} \lor P_{1,3}) \]
[1,1] → [1,2]: stench but no breeze

Add to KB:

\[
\begin{align*}
R_{11}: & \quad \neg B_{1,2} \\
R_{12}: & \quad B_{1,2} \iff (P_{1,1} \lor P_{2,2} \lor P_{1,3}) \\
R_{13}: & \quad \neg P_{2,2} \\
R_{14}: & \quad \neg P_{1,3}
\end{align*}
\]

Similarly, as in deriving \( R_{10} \)
[1,1] → [1,2]: stench but no breeze

Add to KB:

\[ R_{11}: \neg B_{1,2} \]
\[ R_{12}: B_{1,2} \iff (P_{1,1} \lor P_{2,2} \lor P_{1,3}) \]

Similarly, as in deriving \( R_{10} \)

\[ R_{13}: \neg P_{2,2} \]
\[ R_{14}: \neg P_{1,3} \]

\[ R_{3}: B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1}) \]
[1,1] → [1,2]: stench but no breeze

Add to KB:

- \( R_{11} \): \( \neg B_{1,2} \)
- \( R_{12} \): \( B_{1,2} \iff (P_{1,1} \lor P_{2,2} \lor P_{1,3}) \)

Similarly, as in deriving \( R_{10} \)

- \( R_{13} \): \( \neg P_{2,2} \)
- \( R_{14} \): \( \neg P_{1,3} \)

\( R_3 \): \( B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1}) \)

biconditional elimination

\( R_5 \): \( B_{2,1} \)

\( R_{15} \): \( P_{1,1} \lor P_{2,2} \lor P_{3,1} \)
Resolvent

\[ R_{13}: \neg P_{2,2} \]

\[ R_{15}: P_{1,1} \lor P_{2,2} \lor P_{3,1} \]
Resolvent

\[ R_{13}: \neg P_{2,2} \]

\[ R_{15}: P_{1,1} \lor P_{2,2} \lor P_{3,1} \]

resolving the two literals that are negations of each other
Resolvent

\( R_{13}: \ \neg P_{2,2} \)

\( R_{15}: P_{1,1} \lor P_{2,2} \lor P_{3,1} \)

Resolving the two literals that are negations of each other

\( R_{16}: P_{1,1} \lor P_{3,1} \) (resolvent)
If there’s a pit in one of [1,1], [2,2], and [3,1] and it’s not in [2,2], then it’s in [1,1] or [3,1].
Resolvent

\[ R_{13}: \neg P_{2,2} \quad R_{15}: P_{1,1} \lor P_{2,2} \lor P_{3,1} \]

resolving the two literals that are negations of each other

\[ R_{16}: P_{1,1} \lor P_{3,1} \quad \text{(resolvent)} \]

If there’s a pit in one of [1,1], [2,2], and [3,1] and it’s not in [2,2], then it’s in [1,1] or [3,1].

\[ R_1: \neg P_{1,1} \quad R_{16}: P_{1,1} \lor P_{3,1} \]
Resolvent

\[ R_{13}: \neg P_{2,2} \quad R_{15}: P_{1,1} \lor P_{2,2} \lor P_{3,1} \]

resolving the two literals that are negations of each other

\[ R_{16}: P_{1,1} \lor P_{3,1} \quad (\text{resolvent}) \]

If there’s a pit in one of [1,1], [2,2], and [3,1] and it’s not in [2,2], then it’s in [1,1] or [3,1].

\[ R_{1}: \neg P_{1,1} \quad R_{16}: P_{1,1} \lor P_{3,1} \]
Resolvent

\[ R_{13}: \neg P_{2,2} \quad R_{15}: P_{1,1} \lor P_{2,2} \lor P_{3,1} \]

resolving the two literals that are negations of each other

\[ R_{16}: P_{1,1} \lor P_{3,1} \] (resolvent)

If there’s a pit in one of [1,1], [2,2], and [3,1] and it’s not in [2,2], then it’s in [1,1] or [3,1].

\[ R_1: \neg P_{1,1} \quad R_{16}: P_{1,1} \lor P_{3,1} \]

\[ R_{17}: P_{3,1} \]
Simple Resolution Rule

\[
\begin{array}{c}
l_1 \lor \cdots \lor l_i \lor \cdots \lor l_k, \quad m \\
l_1 \lor \cdots \lor l_{i-1} \lor l_{i+1} \lor \cdots \lor l_k
\end{array}
\]

\( (l_i \text{ and } m \text{ are complementary literals, i.e., } l_i = \neg m \text{ or } m = \neg l_i. ) \)
Simple Resolution Rule

\[
\frac{l_1 \lor \cdots \lor l_i \lor \cdots \lor l_k, \quad m}{l_1 \lor \cdots \lor l_{i-1} \lor l_{i+1} \lor \cdots \lor l_k}
\]

\(l_i\) and \(m\) are complementary literals, i.e., \(l_i = \neg m\) or \(m = \neg l_i\).

Since \(m\) is true, then \(l_i\) must be false. But one of \(l_1, \ldots, l_k\) must be true. Therefore, we can exclude \(l_i\) and assert that one of the remaining \(k - 1\) literals must be true.
Simple Resolution Rule

\[ l_1 \lor \cdots \lor l_i \lor \cdots \lor l_k, \quad m \]
\[ l_1 \lor \cdots \lor l_{i-1} \lor l_{i+1} \lor \cdots \lor l_k \]

(l_i and m are complementary literals, i.e., \( l_i = \neg m \) or \( m = \neg l_i \).)

Since \( m \) is true, then \( l_i \) must be false. But one of \( l_1, \ldots, l_k \) must be true. Therefore, we can exclude \( l_i \) and assert that one of the remaining \( k - 1 \) literals must be true.

Clause: a disjunction of literals.

\[ R_{15}: P_{1,1} \lor P_{2,2} \lor P_{3,1} \]
Simple Resolution Rule

\[
\frac{l_1 \lor \cdots \lor l_i \lor \cdots \lor l_k, \quad m}{l_1 \lor \cdots \lor l_{i-1} \lor l_{i+1} \lor \cdots \lor l_k}
\]  
(l_i and m are complementary literals, i.e., \(l_i = \neg m\) or \(m = \neg l_i\).)

Since \(m\) is true, then \(l_i\) must be false. But one of \(l_1, \ldots, l_k\) must be true. Therefore, we can exclude \(l_i\) and assert that one of the remaining \(k - 1\) literals must be true.

**Clause:** a disjunction of literals.

\[R_{15}: P_{1,1} \lor P_{2,2} \lor P_{3,1}\]

**Unit clause:** a single literal.
Simple Resolution Rule

\[
\frac{l_1 \lor \cdots \lor l_i \lor \cdots \lor l_k, \quad m}{l_1 \lor \cdots \lor l_{i-1} \lor l_{i+1} \lor \cdots \lor l_k}
\]

\(l_i\) and \(m\) are complementary literals, i.e., \(l_i = \neg m\) or \(m = \neg l_i\).

Since \(m\) is true, then \(l_i\) must be false. But one of \(l_1, \ldots, l_k\) must be true. Therefore, we can exclude \(l_i\) and assert that one of the remaining \(k - 1\) literals must be true.

**Clause**: a disjunction of literals.

\[R_{15}: P_{1,1} \lor P_{2,2} \lor P_{3,1}\]

**Unit clause**: a single literal.

\[R_1: \neg P_{2,2} \quad R_5: B_{2,1}\]
Simple Resolution Rule

\[ l_1 \lor \cdots \lor l_i \lor \cdots \lor l_k, \quad m \]
\[ \frac{}{l_1 \lor \cdots \lor l_{i-1} \lor l_{i+1} \lor \cdots \lor l_k} \]

(l_i and m are complementary literals, i.e., \(l_i = \neg m\) or \(m = \neg l_i\).)

Since \(m\) is true, then \(l_i\) must be false. But one of \(l_1, \ldots, l_k\) must be true. Therefore, we can exclude \(l_i\) and assert that one of the remaining \(k - 1\) literals must be true.

**Clause**: a disjunction of literals.

\[ R_{15}: \ P_{1,1} \lor P_{2,2} \lor P_{3,1} \]

**Unit clause**: a single literal.

\[ R_1: \ \neg P_{2,2} \]

\[ R_5: \ B_{2,1} \]

\[ P_{1,1} \lor P_{2,2} \lor P_{3,1} \]

\[ \neg P_{2,2} \]

\[ P_{1,1} \lor P_{3,1} \]
Full Resolution Rule

$l_i$ and $m_j$ are complementary literals:

\[
\begin{align*}
    l_1 \lor \cdots \lor l_i \lor \cdots \lor l_k, & \quad m_1 \lor \cdots \lor m_j \lor \cdots \lor m_k \\
\end{align*}
\]

\[
\begin{align*}
    l_1 \lor \cdots \lor l_{i-1} \lor l_{i+1} \lor \cdots \lor l_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n
\end{align*}
\]
Full Resolution Rule

$l_i$ and $m_j$ are complementary literals:

\[
l_1 \lor \cdots \lor l_i \lor \cdots \lor l_k, \quad m_1 \lor \cdots \lor m_j \lor \cdots \lor m_k
\]

\[
l_1 \lor \cdots \lor l_{i-1} \lor l_{i+1} \lor \cdots \lor l_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n
\]

If $l_i$ is true, then $m_j$ is false. Hence $m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n$ must be true.

If $l_i$ is false, then $l_1 \lor \cdots \lor l_{i-1} \lor l_{i+1} \lor \cdots \lor l_k$ must be true.
Full Resolution Rule

$l_i$ and $m_j$ are complementary literals:

$$l_1 \lor \ldots \lor l_i \lor \ldots \lor l_k, \quad m_1 \lor \ldots \lor m_j \lor \ldots \lor m_k$$

$$l_1 \lor \ldots \lor l_{i-1} \lor l_{i+1} \lor \ldots \lor l_k \lor m_1 \lor \ldots \lor m_{j-1} \lor m_{j+1} \lor \ldots \lor m_n$$

If $l_i$ is true, then $m_j$ is false. Hence $m_1 \lor \ldots \lor m_{j-1} \lor m_{j+1} \lor \ldots \lor m_n$ must be true.

If $l_i$ is false, then $l_1 \lor \ldots \lor l_{i-1} \lor l_{i+1} \lor \ldots \lor l_k$ must be true.

$$P_{1,1} \lor P_{3,1}, \quad \neg P_{1,1} \lor \neg P_{2,2}$$

$$P_{3,1} \lor \neg P_{2,2}$$
One Pair at a Time

Only one pair of complementary literals can be resolved at each step.

\[ P \lor \neg Q \lor R, \quad \neg P \lor Q \]
One Pair at a Time

Only one pair of complementary literals can be resolved at each step.

\[ P \lor \neg Q \lor R, \quad \neg P \lor Q \]

\[ \neg Q \lor R \lor Q \]
One Pair at a Time

Only one pair of complementary literals can be resolved at each step.

\[ P \lor \neg Q \lor R, \quad \neg P \lor Q \]

\[ \therefore \neg Q \lor R \lor Q \equiv true \]
Only one pair of complementary literals can be resolved at each step.

\[ P \lor \neg Q \lor R, \quad \neg P \lor Q \]

\[ \neg Q \lor R \lor Q \equiv true \]
One Pair at a Time

Only one pair of complementary literals can be resolved at each step.

\[
P \lor \lnot Q \lor R, \quad \lnot P \lor Q
\]

\[
\lnot Q \lor R \lor Q \equiv true
\]

\[
P \lor \lnot Q \lor R, \quad \lnot P \lor Q
\]

\[
R
\]
One Pair at a Time

Only one pair of complementary literals can be resolved at each step.

\[ P \lor \neg Q \lor R, \quad \neg P \lor Q \]

\[ \neg Q \lor R \lor Q \equiv true \]

Incorrect conclusion!
One Pair at a Time

Only one pair of complementary literals can be resolved at each step.

\[
\begin{align*}
P \lor \neg Q \lor R, & \quad \neg P \lor Q \\
\hline
\neg Q \lor R \lor Q & \equiv \text{true}
\end{align*}
\]

Incorrect conclusion!
II. Conjunctive Normal Form

The resolution rule applies to clauses only.

*Conjunctive normal form* (CNF): a conjunction of clauses
The resolution rule applies to clauses only.

**Conjunctive normal form** (CNF): a conjunction of clauses

\[
\begin{align*}
CNFSentence & \rightarrow \ Clause_1 \land \cdots \land \ Clause_n \\
Clause & \rightarrow \ Literal_1 \lor \cdots \lor \ Literal_m \\
Fact & \rightarrow \ Symbol \\
Literal & \rightarrow \ Symbol \mid \neg Symbol \\
Symbol & \rightarrow \ P \mid Q \mid R \mid \ldots
\end{align*}
\]
II. Conjunctive Normal Form

The resolution rule applies to clauses only.

*Conjunctive normal form (CNF)*: a conjunction of clauses

\[
\begin{align*}
CNFSentence & \rightarrow \ Clause_1 \land \cdots \land \ Clause_n \\
Clause & \rightarrow \ Literal_1 \lor \cdots \lor \ Literal_m \\
Fact & \rightarrow \ Symbol \\
Literal & \rightarrow \ Symbol \mid \neg Symbol \\
Symbol & \rightarrow \ P \mid Q \mid R \mid \ldots
\end{align*}
\]

Every sentence of propositional logic is equivalent to a CNF.
Converting to CNF

1. Eliminate $\iff$.
   
   $\alpha \iff \beta$
1. Eliminate $\iff$.

$\alpha \iff \beta$

replaced with

$\downarrow$

$(\alpha \implies \beta) \land (\beta \implies \alpha)$
1. Eliminate $\iff$.

$$\alpha \iff \beta$$

replaced with

$$\neg \alpha \lor (\beta \lor \beta) \land (\alpha \lor \neg \beta)$$

$$B_{1,1} \iff (P_{2,1} \lor P_{1,1})$$

$$\land \land (B_{1,1} \lor (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$$
Converting to CNF

1. Eliminate $\iff$.
   \[
   \alpha \iff \beta \\
   \text{replaced with} \\
   (\alpha \implies \beta) \land (\beta \implies \alpha)
   \]

2. Eliminate $\implies$.

\[
B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \\
(B_{1,1} \implies (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \implies B_{1,1})
\]
Converting to CNF

1. Eliminate $\iff$.
   \[ \alpha \iff \beta \]
   replaced with
   \[ (\alpha \implies \beta) \land (\beta \implies \alpha) \]

2. Eliminate $\implies$.
   \[ \alpha \implies \beta \]
   \[ \neg \alpha \lor \beta \]

\[ B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \]
\[ (B_{1,1} \implies (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \implies B_{1,1}) \]
Converting to CNF

1. Eliminate $\iff$.
   \[ \alpha \iff \beta \]
   replaced with
   \[ (\alpha \implies \beta) \land (\beta \implies \alpha) \]

2. Eliminate $\implies$.
   \[ \alpha \implies \beta \]
   \[ \neg \alpha \lor \beta \]
   \[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg (P_{1,2} \lor P_{2,1}) \lor B_{1,1}) \]
Converting to CNF

1. Eliminate $\iff$.
   
   $\alpha \iff \beta$
   
   replaced with
   
   $(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)$

2. Eliminate $\Rightarrow$.
   
   $\alpha \Rightarrow \beta$
   
   $\neg \alpha \lor \beta$

3. Move $\neg$ inwards, repeatedly applying
   
   $\neg(\neg \alpha) \equiv \alpha$
   
   $\neg(\alpha \lor \beta) \equiv (\neg \alpha \land \neg \beta)$
   
   $\neg(\alpha \land \beta) \equiv (\neg \alpha \lor \neg \beta)$
Converting to CNF

1. Eliminate $\iff$.
   \[
   \alpha \iff \beta
   \]
   replaced with
   \[
   (\alpha \implies \beta) \land (\beta \implies \alpha)
   \]

2. Eliminate $\implies$.
   \[
   \alpha \implies \beta
   \]
   \[
   \neg \alpha \lor \beta
   \]

3. Move $\neg$ inwards, repeatedly applying
   \[
   \neg(\neg \alpha) \equiv \alpha
   \]
   \[
   \neg(\alpha \lor \beta) \equiv (\neg \alpha \land \neg \beta)
   \]
   \[
   \neg(\alpha \land \beta) \equiv (\neg \alpha \lor \neg \beta)
   \]

   \[
   B_{1,1} \iff (P_{1,2} \lor P_{2,1})
   \]
   \[
   B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \land ((P_{1,2} \lor P_{2,1}) \implies B_{1,1})
   \]
   \[
   (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg(P_{1,2} \lor P_{2,1}) \lor B_{1,1})
   \]
   \[
   (\neg B_{1,1} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1})
   \]
Converting to CNF

1. Eliminate $\iff$.
   
   $\alpha \iff \beta$
   
   replaced with
   
   $$(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)$$
   
   $B_{1,1} \iff (P_{1,2} \lor P_{2,1})$
   
   $$\Downarrow$$
   
   $$(B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$$

2. Eliminate $\Rightarrow$.
   
   $\alpha \Rightarrow \beta$
   
   $$\Downarrow$$
   
   $$\neg \alpha \lor \beta$$
   
   $$(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg (P_{1,2} \lor P_{2,1}) \lor B_{1,1})$$

3. Move $\neg$ inwards, repeatedly applying
   
   $$\neg (\neg \alpha) \equiv \alpha$$
   
   $$\neg (\alpha \lor \beta) \equiv (\neg \alpha \land \neg \beta)$$
   
   $$\neg (\alpha \land \beta) \equiv (\neg \alpha \lor \neg \beta)$$
   
   $$(\neg B_{1,1} \lor \neg P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \land \neg B_{1,1} \lor P_{1,2} \lor P_{2,1} \lor B_{1,1}) \land (\neg P_{2,1} \lor P_{1,2} \lor B_{1,1}))$$

4. Apply the distributivity law
Converting to CNF

1. Eliminate $\iff$.
   \[\alpha \iff \beta\]
   replaced with
   \[(\alpha \implies \beta) \land (\beta \implies \alpha)\]
   \[B_{1,1} \iff (P_{1,2} \lor P_{2,1})\]
   \[(B_{1,1} \implies (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \implies B_{1,1})\]

2. Eliminate $\implies$.
   \[\alpha \implies \beta\]
   \[\neg \alpha \lor \beta\]
   \[\neg B_{1,1} \lor P_{1,2} \lor P_{2,1} \land (\neg (P_{1,2} \lor P_{2,1}) \lor B_{1,1})\]

3. Move $\neg$ inwards, repeatedly applying
   \[\neg (\neg \alpha) \equiv \alpha\]
   \[\neg (\alpha \lor \beta) \equiv (\neg \alpha \land \neg \beta)\]
   \[\neg (\alpha \land \beta) \equiv (\neg \alpha \lor \neg \beta)\]
   \[\neg B_{1,1} \lor \neg P_{1,2} \lor P_{2,1} \land (\neg P_{1,2} \land \neg P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1})\]

4. Apply the distributivity law
   \[\neg B_{1,1} \lor \neg P_{1,2} \lor P_{2,1} \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{1,2} \lor B_{1,1})\]
Parse every propositional sentence in the KB as an arithmetic expression to construct an expression tree (Com S 228).

Operators (connectives): ¬, ∧, ∨, ⇒, ⇔

Operands (atomic sentences): P, Q, R, S, T, ...
CNF Conversion Algorithm (Optional)

- Parse every propositional sentence in the KB as an arithmetic expression to construct an expression tree (Com S 228).

Operators (connectives): \(\neg, \land, \lor, \Rightarrow, \Leftrightarrow\)

Operands (atomic sentences): \(P, Q, R, S, T, \ldots\)

\[\neg(P \land \neg Q) \lor R \Rightarrow S \land \neg T\]

(infix expression)
CNF Conversion Algorithm (Optional)

Parse every propositional sentence in the KB as an arithmetic expression to construct an expression tree (Com S 228).

Operators (connectives): \(\neg, \land, \lor, \Rightarrow, \iff\)

Operands (atomic sentences): \(P, Q, R, S, T, \ldots\)

\[\neg(P \land \neg Q) \lor R \Rightarrow S \land \neg T\]

(infix expression)
Parse every propositional sentence in the KB as an arithmetic expression to construct an expression tree (Com S 228).

Operators (connectives): ¬, ∧, ∨, ⇒, ⇔
Operands (atomic sentences): P, Q, R, S, T, ...

¬(P ∧ ¬Q) ∨ R ⇒ S ∧ ¬T  
(infix expression)

Construction is similar to the algorithm employing a stack to convert an infix expression into a postfix expression (Com S 228).
Parse every propositional sentence in the KB as an arithmetic expression to construct an expression tree (Com S 228).

- Operators (connectives): \( \neg \), \( \land \), \( \lor \), \( \Rightarrow \), \( \iff \)
- Operands (atomic sentences): \( P, Q, R, S, T, ... \)

\[
\neg (P \land \neg Q) \lor R \Rightarrow S \land \neg T
\]

(infix expression)

Construction is similar to the algorithm employing a stack to convert an infix expression into a postfix expression (Com S 228).

\[
PQ \neg \land \neg R \lor ST \neg \land \Rightarrow
\]

(postfix expression)
CNF Conversion Algorithm (Optional)

- Parse every propositional sentence in the KB as an arithmetic expression to construct an **expression tree** (Com S 228).

\[
\neg(P \land \neg Q) \lor R \Rightarrow S \land \neg T
\]

(infix expression)

- Construction is similar to the algorithm employing a stack to convert an infix expression into a postfix expression (Com S 228).

\[
PQ\neg \land \neg R \lor ST \neg \land \Rightarrow
\]

(postfix expression)

- Instead of outputting an operator after its two operands in the postfix format, now you just make the logical operator the parent of the two roots of the subtrees that store the same operator’s subexpression operands.
Postorder Traversal

Perform a postorder traversal of the expression tree.

- When visiting an internal node $n$ (representing a connective), its left and right children (or its unique child in the case of a $\neg$ node) store the CNFs for the expressions represented by the left and right subtrees.
Postorder Traversal

- Perform a postorder traversal of the expression tree.

  - When visiting an internal node $n$ (representing a connective), its left and right children (or its unique child in the case of a $\neg$ node) store the CNFs for the expressions represented by the left and right subtrees.

  ![Diagram of a tree with nodes labeled $n$, $u$, and $v$, and a CNF node labeled $u$.]

  - Conversion is done in five cases depending on the logical operator stored at $n$. 
Case 1

The node $n$ stores $\land$.

$$CNF \equiv CNF_1 \land CNF_2$$
Case 2

The node \( n \) stores \( V \).

\[
CNF_1 \equiv C_1 \land \cdots \land C_k
\]

\[
CNF_2 \equiv C'_1 \land \cdots \land C'_m
\]

\[
CNF \equiv CNF_1 \lor CNF_2 \equiv \land_{i=1,\ldots,k} \left( C_i \lor C'_j \right)
\]

\[
\land_{j=1,\ldots,m}
\]
Case 3

The node $n$ stores $\neg$.

$$CNF \equiv (l_{11} \lor \cdots \lor l_{1k_1}) \land \cdots \land (l_{r1} \lor \cdots \lor l_{rk_r})$$
Case 3

The node $n$ stores $\neg$.

$$CNF \equiv (l_{11} \lor \cdots \lor l_{1k_1}) \land \cdots \land (l_{r1} \lor \cdots \lor l_{rk_r})$$

$$\neg CNF \equiv \bigwedge_{1 \leq j_1 \leq k_1} (\neg l_{1j_1} \lor \cdots \lor \neg l_{rj_r})$$
Case 3

The node $n$ stores $\neg$.

$CNF \equiv (l_{11} \lor \cdots \lor l_{1k_1}) \land \cdots \land (l_{r1} \lor \cdots \lor l_{rk_r})$

$\neg CNF \equiv \land_{1 \leq j_1 \leq k_1} (\neg l_{1j_1} \lor \cdots \lor \neg l_{rj_r})$

- If $l_{ij}$ is a negative literal, i.e., $l_{ij} = \neg p_{ij}$, then $\neg l_{ij}$ reduces to $p_{ij}$ because the two occurrences of $\neg$ cancel out.
Case 4

The node $n$ stores $\Rightarrow$.

- Logically equivalent to $\neg CNF_1 \lor CNF_2$.
- First, convert $\neg CNF_1$ into conjunctive normal form $CNF_1'$ (see case 3).
- Then, convert the disjunction $CNF_1' \lor CNF_2$ into conjunctive normal form (see case 2)
Case 5

The node $n$ stores $\iff$.

- Logically equivalent to $(CNF_1 \Rightarrow CNF_2) \land (CNF_2 \Rightarrow CNF_1)$.

- Convert $CNF_1 \Rightarrow CNF_2$ and $CNF_2 \Rightarrow CNF_1$ separately into conjunctive normal forms $CNF_1'$ and $CNF_2'$ (see case 4).

- Return $CNF_1' \land CNF_2'$.
\( \neg(P \land \neg Q) \lor R \Rightarrow S \land \neg T \) has the following conjunctive normal form:

\[
(P \lor S) \land (\neg Q \lor S) \land (\neg R \lor S) \land 
(P \lor \neg T) \land (\neg Q \lor \neg T) \land (\neg R \lor \neg T)
\]
III. Proof by Resolution – An Example

\[ KB: \]

\[ P \]
\[ P \rightarrow (Q \lor R) \]
\[ Q \rightarrow S \]
\[ R \rightarrow (S \land T) \]
III. Proof by Resolution – An Example

KB:

\[
\begin{align*}
P \\
P \rightarrow (Q \lor R) \\
Q \rightarrow S \\
R \rightarrow (S \land T)
\end{align*}
\]

\[\neg P \lor Q \lor R\]
III. Proof by Resolution – An Example

**KB:**

\[
\begin{align*}
P & \\
P \rightarrow (Q \lor R) & \\
Q \rightarrow S & \\
R \rightarrow (S \land T) & \\
& \quad \quad \rightarrow \neg P \lor Q \lor R \\
& \quad \quad \rightarrow \neg Q \lor S
\end{align*}
\]
III. Proof by Resolution – An Example

KB:

\[
\begin{align*}
P & \\
P \rightarrow (Q \lor R) & \quad \rightarrow \quad \neg P \lor Q \lor R \\
Q \rightarrow S & \quad \rightarrow \quad \neg Q \lor S \\
R \rightarrow (S \land T) & \quad \rightarrow \quad \neg R \lor (S \land T)
\end{align*}
\]
III. Proof by Resolution – An Example

**KB:**

\[
\begin{align*}
P \\
P &\rightarrow (Q \lor R) \\
Q &\rightarrow S \\
R &\rightarrow (S \land T)
\end{align*}
\]

\[
\begin{align*}
\neg P \lor Q \lor R \\
\neg Q \lor S \\
\neg R \lor (S \land T) \\
(\neg R \lor S) \land (\neg R \lor T)
\end{align*}
\]
III. Proof by Resolution – An Example

KB:

\begin{align*}
P \\
P \rightarrow (Q \lor R) \\
Q \rightarrow S \\
R \rightarrow (S \land T)
\end{align*}

\begin{align*}
\neg P \lor Q \lor R \\
\neg Q \lor S \\
\neg R \lor (S \land T) \\
(\neg R \lor S) \land (\neg R \lor T)
\end{align*}

Q: KB \models S ?
III. Proof by Resolution – An Example

KB:

\[ P \]
\[ P \rightarrow (Q \lor R) \]
\[ Q \rightarrow S \]
\[ R \rightarrow (S \land T) \]

Q: \text{KB} \vdash S? 

1. Converting sentences to CNF
III. Proof by Resolution – An Example

**KB:**

\[
P
P \rightarrow (Q \lor R)
Q \rightarrow S
R \rightarrow (S \land T)
\]

\[
\neg P \lor Q \lor R
\neg Q \lor S
\neg R \lor (S \land T)
\neg R \lor S \land (\neg R \lor T)
\]

**Q:** \( KB \vdash S \) ?

1. Converting sentences to CNF

**KB:**

\[
P
\neg P \lor Q \lor R
\neg Q \lor S
\neg R \lor (S \land T)
\neg R \lor S \land (\neg R \lor T)
\]
III. Proof by Resolution – An Example

**KB:**

- $P$
- $P \rightarrow (Q \lor R)$
- $Q \rightarrow S$
- $R \rightarrow (S \land T)$

### Converting sentences to CNF

1. **KB:**
   - $P$
   - $\neg P \lor Q \lor R$
   - $\neg Q \lor S$
   - $\neg R \lor (S \land T)$
   - $(\neg R \lor S) \land (\neg R \lor T)$

**Q:** $KB \vdash S$?

1. Converting sentences to CNF
2. Split each conjunction into clauses.

**KB:**

- $P$
- $\neg P \lor Q \lor R$
- $\neg Q \lor S$
- $(\neg R \lor S) \land (\neg R \lor T)$
III. Proof by Resolution – An Example

KB:

\[
\begin{align*}
P \\
P & \rightarrow (Q \lor R) \\
Q & \rightarrow S \\
R & \rightarrow (S \land T)
\end{align*}
\]

\[
\begin{align*}
\neg P \lor Q \lor R \\
\neg Q \lor S \\
\neg R \lor (S \land T) \\
(\neg R \lor S) \land (\neg R \lor T)
\end{align*}
\]

Q: \textit{KB} \vdash S ?

1. Converting sentences to CNF
2. Split each conjunction into clauses.

KB:

\[
\begin{align*}
P \\
\neg P \lor Q \lor R \\
\neg Q \lor S \\
(\neg R \lor S) \land (\neg R \lor T)
\end{align*}
\]
III. Proof by Resolution – An Example

KB:

\[
\begin{align*}
P & \\
P \rightarrow (Q \lor R) & \rightarrow \neg P \lor Q \lor R \\
Q \rightarrow S & \rightarrow \neg Q \lor S \\
R \rightarrow (S \land T) & \rightarrow \neg R \lor (S \land T) \\
& \rightarrow (\neg R \lor S) \land (\neg R \lor T)
\end{align*}
\]

Q: KB ⊢ S ?

1. Converting sentences to CNF

KB:

\[
\begin{align*}
P & \\
\neg P \lor Q \lor R & \\
\neg Q \lor S & \\
(\neg R \lor S) \land (\neg R \lor T) & \\
\end{align*}
\]

2. Split each conjunction into clauses.

KB:

\[
\begin{align*}
P & \\
\neg P \lor Q \lor R & \\
\neg Q \lor S & \\
\neg R \lor S & \\
\neg R \lor T & \\
\end{align*}
\]
Proof by Resolution

$KB$ (updated):

1. $P$
2. $\neg P \lor Q \lor R$
3. $\neg Q \lor S$
4. $\neg R \lor S$
5. $\neg R \lor T$

(1) $P$  (2) $\neg P \lor Q \lor R$
Proof by Resolution

**KB (updated):**

1. $P$
2. $\neg P \lor Q \lor R$
3. $\neg Q \lor S$
4. $\neg R \lor S$
5. $\neg R \lor T$

(1) $P$  (2) $\neg P \lor Q \lor R$

Resolve:

(3) $\neg Q \lor S$
(4) $\neg R \lor S$
(5) $\neg R \lor T$
(6) $Q \lor R$

(1) $P$
(2) $\neg P \lor Q \lor R$

resolve

(6) $Q \lor R$
Proof by Resolution

KB (updated):

(1)  $P$
(2)  $\neg P \lor Q \lor R$
(3)  $\neg Q \lor S$
(4)  $\neg R \lor S$
(5)  $\neg R \lor T$

<table>
<thead>
<tr>
<th>(1)</th>
<th>(2)</th>
<th>resolve</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P$</td>
<td>$\neg P \lor Q \lor R$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\neg Q \lor S$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$Q \lor R$</td>
<td></td>
</tr>
</tbody>
</table>
Proof by Resolution

\[ KB \text{ (updated):} \]

(1) \( P \)
(2) \( \neg P \lor Q \lor R \)
(3) \( \neg Q \lor S \)
(4) \( \neg R \lor S \)
(5) \( \neg R \lor T \)

\[
\begin{align*}
(1) \quad P & \quad (2) \quad \neg P \lor Q \lor R \\
(3) \quad \neg Q \lor S & \quad \text{resolve} \\
(4) \quad \neg R \lor S & \\
(5) \quad \neg R \lor T & \\
(6) \quad Q \lor R & \\
(7) \quad S \lor R &
\end{align*}
\]
Proof by Resolution

KB (updated):

(1) $P$
(2) $\neg P \lor Q \lor R$
(3) $\neg Q \lor S$
(4) $\neg R \lor S$
(5) $\neg R \lor T$

(1) $P$
(2) $\neg P \lor Q \lor R$

Resolve:

(3) $\neg Q \lor S$
(4) $\neg R \lor S$
(6) $Q \lor R$
(7) $S \lor R$
Proof by Resolution

KB (updated):

(1)  𝑃
(2)  ¬𝑃 ∨ 𝑄 ∨ 𝑅
(3)  ¬𝑄 ∨ 𝑆
(4)  ¬𝑅 ∨ 𝑆
(5)  ¬𝑅 ∨ 𝑇

(1)  𝑃  (2)  ¬𝑃 ∨ 𝑄 ∨ 𝑅

resolve

(3)  ¬𝑄 ∨ 𝑆  (6)  𝑄 ∨ 𝑅

(4)  ¬𝑅 ∨ 𝑆  (7)  𝑆 ∨ 𝑅

(8)  𝑆 ∨ 𝑆 ≡ 𝑆
Proof by Resolution

$KB$ (updated):
1. $P$
2. $\neg P \lor Q \lor R$
3. $\neg Q \lor S$
4. $\neg R \lor S$
5. $\neg R \lor T$

Resolution tree:

1. $P$
2. $\neg P \lor Q \lor R$
3. $\neg Q \lor S$
4. $\neg R \lor S$
5. $\neg R \lor T$
6. $Q \lor R$
7. $S \lor R$
8. $S \lor S \equiv S$
Resolution Refutation

(Proof by contradiction)
To show that $KB \models \alpha$, we show that $KB \land \neg \alpha$ is unsatisfiable.
Resolution Refutation

(Proof by contradiction)
To show that $KB \models \alpha$, we show that $KB \land \neg \alpha$ is unsatisfiable.

KB (about a summer day):

(1) If it is raining and you are outside then you will get wet.
(2) If it is warm and there is no rain then it is a pleasant day.
(3) You are not wet.
(4) You are outside.
(5) It is a warm day.

Resolution Refutation

(Proof by contradiction)
To show that $KB \vDash \alpha$, we show that $KB \land \neg \alpha$ is unsatisfiable.

KB (about a summer day):

1. If it is raining and you are outside then you will get wet.
2. If it is warm and there is no rain then it is a pleasant day.
3. You are not wet.
4. You are outside.
5. It is a warm day.

Prove

It is a pleasant day.

KB (rewritten):

(1) (rain ∧ outside) ⇒ wet
(2) (warm ∧ ¬rain) ⇒ pleasant
(3) ¬wet
(4) outside
(5) warm
KB in Propositional Sentences

KB (rewritten):

(1) \((\text{rain} \land \text{outside}) \Rightarrow \text{wet}\)
(2) \((\text{warm} \land \neg \text{rain}) \Rightarrow \text{pleasant}\)
(3) \neg \text{wet}
(4) \text{outside}
(5) \text{warm}

converted into clauses

(1) \neg \text{rain} \lor \neg \text{outside} \lor \text{wet}
(2) \neg \text{warm} \lor \text{rain} \lor \text{pleasant}
(3) \neg \text{wet}
(4) \text{outside}
(5) \text{warm}
KB in Propositional Sentences

KB (rewritten):

(1) (rain ∧ outside) ⇒ wet
(2) (warm ∧ ¬rain) ⇒ pleasant
(3) ¬wet
(4) outside
(5) warm

converted into clauses

(1) ¬rain V ¬outside V wet
(2) ¬warm V rain V pleasant
(3) ¬wet
(4) outside
(5) warm

We add ¬pleasant to KB and try to derive an empty clause ∅.
Resolution Refutation Tree

¬pleasant  (2) ¬warm ∨ rain ∨ pleasant
Resolution Refutation Tree

\[ \neg \text{pleasant} \]

\[ \neg \text{warm} \lor \text{rain} \]

(2) \[ \neg \text{warm} \lor \text{rain} \lor \text{pleasant} \]
Resolution Refutation Tree

\[ \neg \text{pleasant} \]
\[ (2) \quad \neg \text{warm} \lor \text{rain} \lor \neg \text{pleasant} \]
\[ \neg \text{warm} \lor \text{rain} \]
\[ (5) \quad \text{warm} \]
Resolution Refutation Tree

¬pleasant

¬warm ∨ rain

rain

(1) ¬rain V ¬outside V wet

(2) ¬warm V rain V pleasant

(5) warm
Resolution Refutation Tree

¬pleasant

¬warm ∨ rain

rain

¬outside ∨ wet

(2) ¬warm ∨ rain ∨ pleasant

(5) warm

(1) ¬rain ∨ ¬outside ∨ wet
Resolution Refutation Tree

¬pleasant

¬warm ∨ rain

rain

¬outside ∨ wet

¬rain ∨ ¬outside ∨ wet (1)

¬outside ∨ wet (4) outside

¬warm ∨ rain ∨ pleasant (2)

warm (5)
Resolution Refutation Tree

¬pleasant

¬warm V rain

rain

¬outside V wet

wet

(1) ¬rain V ¬outside V wet

(2) ¬warm V rain V pleasant

(4) outside

(5) warm
Resolution Refutation Tree

\[ \neg\text{pleasant} \]

\[ \neg\text{warm} \lor \text{rain} \]

\[ \text{rain} \]

\[ \neg\text{outside} \lor \text{wet} \]

\[ \text{wet} \]
Resolution Refutation Tree

¬pleasant

¬warm ∨ rain

rain

¬outside ∨ wet

¬wet

∅ (empty clause)
Resolution Refutation Tree

\[ \neg \text{pleasant} \]
\[ \neg \text{warm} \lor \text{rain} \]
\[ \text{rain} \]
\[ \neg \text{outside} \lor \text{wet} \]
\[ \text{wet} \]
\[ \emptyset \text{ (empty clause)} \]

contradiction!
Proving \( \neg P_{1,2} \) in the Wumpus World

\[
(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})
\]
Resolution Algorithm

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 $C_i, C_j$ in clauses do
      resolvents ← PL-RESOLVE($C_i, C_j$)
      if resolvents contains the empty clause then return true
      new ← new ∪ resolvents
      if new ⊆ clauses then return false // no new clauses can be added.
    clauses ← clauses ∪ new

The process ends in one of two situations below:

- No new clauses can be added, in which case $KB$ does not entail $α$;
- Two clauses resolve to yield the empty clause, in which case $KB$ entails $α$. 
Completeness of Resolution

Given a set of clauses $S$, its *resolution closure* $RC(S)$ includes all the clauses in $S$ as well as all the resolvents from repeated applications of the resolution rule.
Completeness of Resolution

Given a set of clauses $S$, its resolution closure $RC(S)$ includes all the clauses in $S$ as well as all the resolvents from repeated applications of the resolution rule.

$RC(S)$ is finite because only $3^n$ distinct clauses can be constructed out of $n$ propositional symbols appearing in $S$. 
Completeness of Resolution

Given a set of clauses $S$, its \textit{resolution closure} $RC(S)$ includes all the clauses in $S$ as well as all the resolvents from repeated applications of the resolution rule.

$RC(S)$ is \textit{finite} because only $3^n$ distinct clauses can be constructed out of $n$ propositional symbols appearing in $S$.

\textbf{Ground Resolution Theorem}: If $S$ is unsatisfiable, then $RC(S)$ contains the empty clause $\emptyset$. 
Completeness of Resolution

Given a set of clauses $S$, its *resolution closure* $RC(S)$ includes all the clauses in $S$ as well as all the resolvents from repeated applications of the resolution rule.

$RC(S)$ is finite because only $3^n$ distinct clauses can be constructed out of $n$ propositional symbols appearing in $S$.

**Ground Resolution Theorem**: If $S$ is unsatisfiable, then $RC(S)$ contains the empty clause $\emptyset$.

Constructive proof by explicitly generating an assignment for $S$ if $\emptyset \notin RC(S)$. 
IV. Horn Clauses

A clause is called a *Horn clause* if it contains $\leq 1$ positive literal.

$P$: positive literal   $\neg P$: negative literal
A clause is called a *Horn clause* if it contains $\leq 1$ positive literal.

- **$P$:** positive literal  
- **$\neg P$:** negative literal

- *Definite clause* (1 positive literal and $\geq 1$ negative literal)

\[ \neg P \lor \neg Q \lor R \]
A clause is called a Horn clause if it contains $\leq 1$ positive literal.

$P$: positive literal $\neg P$: negative literal

- **Definite clause** (1 positive literal and $\geq 1$ negative literal)

\[ \neg P \lor \neg Q \lor R \equiv P \land Q \Rightarrow R \]
IV. Horn Clauses

A clause is called a *Horn clause* if it contains $\leq 1$ positive literal.

- $\neg P \lor \neg Q \lor R \equiv P \land Q \Rightarrow R$

- **Definite clause** (1 positive literal and $\geq 1$ negative literal)

- $\neg \text{rain} \lor \neg \text{outside} \lor \text{wet}$
IV. Horn Clauses

A clause is called a *Horn clause* if it contains $\leq 1$ positive literal.

- $P$: positive literal       $\neg P$: negative literal

- *Definite clause* (1 positive literal and $\geq 1$ negative literal)

$$

\neg P \lor \neg Q \lor R \equiv P \land Q \Rightarrow R \\

\neg \text{rain} \lor \neg \text{outside} \lor \text{wet} \equiv \text{rain} \land \text{outside} \Rightarrow \text{wet}

$$
IV. Horn Clauses

A clause is called a *Horn clause* if it contains $\leq 1$ positive literal.

$P$: positive literal \hspace{1cm} \neg P$: negative literal

- *Definite clause* (1 positive literal and $\geq 1$ negative literal)

\[
\neg P \lor \neg Q \lor R \equiv P \land Q \Rightarrow R \\
\neg \text{rain} \lor \neg \text{outside} \lor \text{wet} \equiv \text{rain} \land \text{outside} \Rightarrow \text{wet} \\
\neg P_1 \lor \neg P_2 \lor \cdots \lor \neg P_k \lor Q \equiv (P_1 \land P_2 \land \cdots \land P_k) \Rightarrow Q
\]
IV. Horn Clauses

A clause is called a *Horn clause* if it contains $\leq 1$ positive literal.

$P$: positive literal  $\neg P$: negative literal

- **Definite clause** (1 positive literal and $\geq 1$ negative literal)

  \[ \neg P \lor \neg Q \lor R \equiv P \land Q \Rightarrow R \]

  \[ \neg \text{rain} \lor \neg \text{outside} \lor \text{wet} \equiv \text{rain} \land \text{outside} \Rightarrow \text{wet} \]

  \[ \neg P_1 \lor \neg P_2 \lor \ldots \lor \neg P_k \lor Q \equiv (P_1 \land P_2 \land \ldots \land P_k) \Rightarrow Q \]

- **Fact** (1 positive literal and 0 negative literal)

  \[ P_{1,2} \]
IV. Horn Clauses

A clause is called a *Horn clause* if it contains $\leq 1$ positive literal.

$P$: positive literal \hspace{1cm} \neg P$: negative literal

- **Definite clause** (1 positive literal and $\geq 1$ negative literal)
  
  \[ \neg P \lor \neg Q \lor R \equiv P \land Q \Rightarrow R \]
  \[ \neg \text{rain} \lor \neg \text{outside} \lor \text{wet} \equiv \text{rain} \land \text{outside} \Rightarrow \text{wet} \]
  \[ \neg P_1 \lor \neg P_2 \lor \cdots \lor \neg P_k \lor Q \equiv (P_1 \land P_2 \land \cdots \land P_k) \Rightarrow Q \]

- **Fact** (1 positive literal and 0 negative literal)
  
  \[ P_{1,2} \equiv \text{true} \Rightarrow P_{1,2} \]
A clause is called a **Horn clause** if it contains \( \leq 1 \) positive literal.

\[
\neg P \lor \neg Q \lor R \equiv P \land Q \Rightarrow R
\]

\[
\neg \text{rain} \lor \neg \text{outside} \lor \text{wet} \equiv \text{rain} \land \text{outside} \Rightarrow \text{wet}
\]

\[
\neg P_1 \lor \neg P_2 \lor \cdots \lor \neg P_k \lor Q \equiv (P_1 \land P_2 \land \cdots \land P_k) \Rightarrow Q
\]

**Definite clause** (1 positive literal and \( \geq 1 \) negative literal)

**Fact** (1 positive literal and 0 negative literal)

\[
P_{1,2} \equiv \text{true} \Rightarrow P_{1,2}
\]

**Goal clause** (0 positive literal and \( \geq 1 \) negative literal)

\[
\neg P_1 \lor \neg P_2 \lor \cdots \lor \neg P_k
\]
IV. Horn Clauses

A clause is called a **Horn clause** if it contains \( \leq 1 \) positive literal.

\[ P: \text{positive literal} \quad \neg P: \text{negative literal} \]

- **Definite clause** (1 positive literal and \( \geq 1 \) negative literal)

\[
\neg P \lor \neg Q \lor R \quad \equiv \quad P \land Q \Rightarrow R
\]

\[
\neg \text{rain} \lor \neg \text{outside} \lor \text{wet} \quad \equiv \quad \text{rain} \land \text{outside} \Rightarrow \text{wet}
\]

\[
\neg P_1 \lor \neg P_2 \lor \cdots \lor \neg P_k \lor Q \quad \equiv \quad (P_1 \land P_2 \land \cdots \land P_k) \Rightarrow Q
\]

- **Fact** (1 positive literal and 0 negative literal)

\[
P_{1,2} \quad \equiv \quad \text{true} \Rightarrow P_{1,2}
\]

- **Goal clause** (0 positive literal and \( \geq 1 \) negative literal)

\[
\neg P_1 \lor \neg P_2 \lor \cdots \lor \neg P_k \quad \equiv \quad (P_1 \land P_2 \land \cdots \land P_k) \Rightarrow \text{false}
\]
Why Horn Clauses?

Horn clauses are the basis of logic programming, and play an important role in automated theorem proving.
Why Horn Clauses?

Horn clauses are the basis of logic programming, and play an important role in automated theorem proving.

- Every definite clause can be written as an implication.

\[-P_1 \lor -P_2 \lor \cdots \lor -P_k \lor Q \equiv (P_1 \land P_2 \land \cdots \land P_k) \Rightarrow Q\]
Why Horn Clauses?

Horn clauses are the basis of logic programming, and play an important role in automated theorem proving.

- Every definite clause can be written as an implication.

\[
\neg P_1 \lor \neg P_2 \lor \cdots \lor \neg P_k \lor Q \equiv (P_1 \land P_2 \land \cdots \land P_k) \Rightarrow Q
\]

\[
\downarrow
\]

\[
Q \Leftarrow (P_1 \land P_2 \land \cdots \land P_k)
\]
Why Horn Clauses?

Horn clauses are the basis of logic programming, and play an important role in automated theorem proving.

- Every definite clause can be written as an implication.

\[
\neg P_1 \lor \neg P_2 \lor \cdots \lor \neg P_k \lor Q \equiv (P_1 \land P_2 \land \cdots \land P_k) \Rightarrow Q
\]

\[
\downarrow
\]

\[
Q \Leftarrow (P_1 \land P_2 \land \cdots \land P_k)
\]

\[
\downarrow
\]

\[
Q :\ P_1, P_2, \ldots, P_k.
\]

(Prolog programming language)
Horn clauses are the basis of logic programming, and play an important role in automated theorem proving.

- Every definite clause can be written as an implication.

\[
\neg P_1 \lor \neg P_2 \lor \ldots \lor \neg P_k \lor Q \quad \equiv \quad (P_1 \land P_2 \land \ldots \land P_k) \Rightarrow Q
\]

\[
\Downarrow
\]

\[
Q \Leftarrow (P_1 \land P_2 \land \ldots \land P_k)
\]

\[
\Downarrow
\]

\[
Q : P_1, P_2, \ldots, P_k.
\]

(Prolog programming language)

- Horn clauses are closed under resolution, i.e., the resolvent of a Horn clause is still a Horn clause.
Why Horn Clauses?

Horn clauses are the basis of logic programming, and play an important role in automated theorem proving.

- Every definite clause can be written as an implication.

\[ \neg P_1 \lor \neg P_2 \lor \cdots \lor \neg P_k \lor Q \equiv (P_1 \land P_2 \land \cdots \land P_k) \Rightarrow Q \]

\[ \Downarrow \]

\[ Q \Leftarrow (P_1 \land P_2 \land \cdots \land P_k) \]

\[ \Downarrow \]

\[ Q :\neg P_1, P_2, \ldots, P_k. \]

(Prolog programming language)

- Horn clauses are closed under resolution, i.e., the resolvent of a Horn clause is still a Horn clause.

\[ \neg P \lor \neg Q \lor R \]

\[ \neg R \lor S \]

\[ \Downarrow \]

\[ \neg P \lor \neg Q \lor S \]
Why Horn Clauses?

Horn clauses are the basis of logic programming, and play an important role in automated theorem proving.

- Every definite clause can be written as an implication.
  \[
  \neg P_1 \lor \neg P_2 \lor \cdots \lor \neg P_k \lor Q \equiv (P_1 \land P_2 \land \cdots \land P_k) \Rightarrow Q
  \]
  
  \[
  \Downarrow
  \]
  
  \[
  Q \Leftarrow (P_1 \land P_2 \land \cdots \land P_k)
  \]
  
  \[
  \Downarrow
  \]
  
  \[
  Q :- P_1, P_2, \ldots, P_k.
  \]
  
  (Prolog programming language)

- Horn clauses are closed under resolution, i.e., the resolvent of a Horn clause is still a Horn clause.
Why Horn Clauses?

Horn clauses are the basis of logic programming, and play an important role in automated theorem proving.

- Every definite clause can be written as an implication.

\[ \neg P_1 \lor \neg P_2 \lor \cdots \lor \neg P_k \lor Q \equiv (P_1 \land P_2 \land \cdots \land P_k) \Rightarrow Q \]

\[ \downarrow \]

\[ Q \Leftarrow (P_1 \land P_2 \land \cdots \land P_k) \]

\[ \downarrow \]

\[ Q \iff P_1, P_2, \ldots, P_k. \]  

(Prolog programming language)

- Horn clauses are *closed* under resolution, i.e., the resolvent of a Horn clause is still a Horn clause.

\[ \neg P \lor \neg Q \lor R \]

\[ \neg R \lor S \]

\[ \neg P \lor \neg Q \lor R \]

\[ \neg P \lor \neg Q \lor \neg S \]

\[ \neg R \lor S \]

\[ R \]

\[ \neg P \lor \neg Q \lor R \]

\[ \neg R \lor \neg S \]
Inferences with Horn clauses are through forward- and backward-chaining algorithms.

Logic programming

(natural inference steps easy for humans to follow)

Low computational complexity: deciding entailment with Horn clauses takes $O(n)$ time.

size of the KB