Resolution, Refutation Theorem-proving, and Horn Clauses

Chanseok Oh
chanseok@cs.nyu.edu
Propositional Logic

- Propositional Calculus (Sentential Calculus/Logic)

- literal: a proposition (positive) or its negation (negative)
  \[ P \quad \neg P \]

- clause: a disjunction of literals.
  \[ \neg P \lor Q \lor \neg R \quad P \quad \neg S \lor T \]

- cf. \[ P \land Q \lor \neg R \quad S \lor (\neg T \land Q) \]
Logical Implication

<table>
<thead>
<tr>
<th></th>
<th>Q</th>
<th>P $\rightarrow$ Q</th>
<th>$\neg P \lor Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

If $P$ is false, then $P \rightarrow Q$ is true, regardless of $Q$. 
Resolution

\[
\begin{align*}
(P \lor l_1 \lor l_2 \lor \cdots \lor l_i) & \quad (\neg P \lor m_1 \lor m_2 \lor \cdots \lor m_j) \\
l_1 \lor l_2 \lor \cdots \lor l_i \lor m_1 \lor m_2 \lor \cdots \lor m_j
\end{align*}
\]

The resulting clause is called a *resolvent*.

A resolvent is a natural consequence (deduction), thus redundant.
Resolution

\[(P \lor l_1 \lor l_2 \lor \cdots \lor l_i) \quad (\neg P \lor m_1 \lor m_2 \lor \cdots \lor m_j)\]

\[
\frac{\begin{array}{c}
l_1 \lor l_2 \lor \cdots \lor l_i \\
\lor m_1 \lor m_2 \lor \cdots \lor m_j
\end{array}}{l_1 \lor l_2 \lor \cdots \lor l_i \lor m_1 \lor m_2 \lor \cdots \lor m_j}
\]

It is true that I am:

\(((\text{not sleepy}) \lor \text{hungry}) \quad \text{and} \quad (\text{sleepy} \lor \text{angry})\)

You must be either sleepy or not,
but at least, you are hungry or angry (or both).

a) If you are sleepy, then you must be hungry.
b) If you are not sleepy, then you must be angry.
Modus Ponens

\[
P \rightarrow Q \quad P \\
\hline
Q
\]

Modus Ponens is a special case of resolution

\[
\neg P \vee Q \quad P \\
\hline
Q
\]

(not sleepy) and (sleepy or hungry)

You are hungry.
The Null Clause

• The null or empty clause
  – No literals
  – Can appear as a resolvent (contradiction).
    \[
    \neg P \quad P
    \]
    \[
    \hline
    ()
    \]
  – Cannot be satisfiable.
Refutation Theorem-proving

\[ \Gamma \vdash \varphi \]

- **Goal:** to prove a theorem \( \varphi \)
- From a set of axioms \( \Gamma \)
- Negated goal \( \neg \varphi \)
Refutation Theorem-proving

• If the *empty clause* can be derived by applying a series of resolution, the negated goal $\neg\varphi$ is unsatisfiable (contradictory).

• Proof by contradiction $\Gamma \cup \neg\varphi$
  
  – The initial formula $\varphi$ follows from $\Gamma$.

• If $\varphi$ is a theorem (provable),
  
  – We can prove or deduce it.
  
  – There is a proof.
  
  – There is a refutation proof.
  
  – The null clause can be derived by resolution.
Gödel’s Incompleteness Theorem

• First-order Logic
  \[ \forall x. P(x) \rightarrow \exists y. Q(y) \]

• Propositional Logic
  – Coupled with a complete search algorithm, the resolution yields sound and complete algorithm for deciding the satisfiability.
Horn Clauses

• A clause with at most one positive literal.

\[ \neg P \lor Q \lor \neg R \quad P \quad \neg S \lor T \]

• Any horn clause belongs to one of four categories:
  – The null clause: an empty clause
  – A fact: 1 pos lit, and 0 neg lit.
  – A rule: 1 pos lit, and at least one neg lit.
  – A negated goal: 0 pos lit, at least one neg lit.
Horn Clause Logic

• A fact or unit
  – 1 positive literal, 0 negative literal.

\[
P
\]

\[
Q
\]

  – In Prolog,
    hungry.
    sunny(rochester).
Horn Clause Logic

- A rule
  - 1 positive literal, $\geq 1$ negative literal.

$$

\neg P_1 \lor \neg P_2 \lor \cdots \lor \neg P_n \lor Q

\neg (P_1 \land P_2 \land \cdots \land P_n) \lor Q

P_1 \land P_2 \land \cdots \land P_n \rightarrow Q

- \text{In Prolog,}
  - \text{goal } \leftarrow \text{ subgoal}_1, \text{ subgoal}_2, ..., \text{ subgoal}_n
  - \text{snowy } :- \text{ rainy, cold.}

$$
Rules express conditional statements about our world. Consider the assertion: “All men are mortal.”

Expressible as modus ponens: human → mortal (“human implies mortal.”)
mortal is a goal (or head), and human is a subgoal (or body).

In Prolog, we write it in the following form:
mortal ← human.

Or more generally,
goal ← subgoal.

There can be multiple subgoals. Example:
goal ← subgoal₁, ..., subgoalₙ.

This form is called a Horn clause.
Horn Clause Logic

• A negated goal
  – 0 positive literal, >= 1 negative literal.

$$\neg P_1 \lor \neg P_2 \lor \cdots \lor \neg P_n$$

$$\neg(P_1 \land P_2 \land \cdots \land P_n)$$

– goal is what you want to prove.

$$P \quad P_1 \land P_2 \land P_3$$

– cf. $$P_1 \lor P_2$$
Horn Clause Theories

• Propositional horn theories
  – Can be decided in polynomial time.

• First-order horn theories
  – Only semi-decidable
    – In practice, resolution over horn theories runs much more efficiently than resolution over general first-order theories.
Resolution on Horn Clauses

• If restricted to horn clauses, some interesting properties appear.
Resolution on Horn Clauses

\[
P \lor \neg X_1 \lor \neg X_2 \lor \ldots \lor \neg X_i \quad \neg P \lor l_1 \lor l_2 \lor \ldots \lor l_j
\]

\[
\neg X_1 \lor \neg X_2 \lor \ldots \lor \neg X_i \lor l_1 \lor l_2 \lor \ldots \lor l_j
\]

• \( l_1, \ldots, l_j \): at most one positive literal

• \( \neg X_1 \lor \neg X_2 \lor \ldots \lor \neg X_i \lor l_1 \lor l_2 \lor \ldots \lor l_j \)
  
  – The resolvent is again a horn clause.
Resolution against a Negated Goal

• Negated goal: \( \neg P \lor \neg Y \lor \neg Y_2 \lor \cdots \lor \neg Y_j \)

\[
P \lor \neg X_1 \lor \neg X_2 \lor \cdots \lor \neg X_i \quad \neg P \lor \neg Y \lor \neg Y_2 \lor \cdots \lor \neg Y_j
\]

\[
\frac{\neg X_1 \lor \neg X_2 \lor \cdots \lor \neg X_i \lor \neg Y_1 \lor \neg Y_2 \lor \cdots \lor \neg Y_j}{\neg X_1 \lor \neg X_2 \lor \cdots \lor \neg X_i \lor \neg Y_i \lor \neg Y_2 \lor \cdots \lor \neg Y_j}
\]

• \( \neg X_1 \lor \neg X_2 \lor \cdots \lor \neg X_i \lor \neg Y_1 \lor \neg Y_2 \lor \cdots \lor \neg Y_j \)
  
  – Horn clause
  
  – Negated goal (if not null)
  
  – \( P \) is removed.
Resolution against a Negated Goal

ψ : a negated goal
Γ : a set of facts and rules
¬P ∈ ψ
C ∈ Γ, P ∈ C
ψ' := resolve(ψ, C)

ψ' (resolvent) is another negated goal with P removed.
Resolution in Prolog

• fact1: rainy.
• fact2: cold.
• rule1: snowy :- rainy, cold.
  (rainy & cold) -> snowy
  ~(rainy & cold) | snowy
  ~rainy | ~cold | snowy
• goal: ?- snowy.

• resolve (~snowy, ~rainy | ~cold | snowy)
• resolve (~rainy | ~cold, rainy)
• resolve (~cold, cold)
Backward Chaining

// ψ : negated goal, Γ : a set of facts and rules
bc(ψ, Γ) {
    if ψ = null then succeed;
    pick a literal ¬P in ψ;
    choose C ∈ Γ where P ∈ C;
    ψ' := resolve(ψ, C);
    bc(ψ', Γ)
}

Order doesn’t matter.

If bc(¬φ, Γ) succeeds, φ is a consequence of Γ.
Backward Chaining

• Complete for horn clauses
  – If \( \Gamma \models \varphi \), there is a backward-chaining proof of \( \varphi \) from \( \Gamma \).
Pure Prolog

• An implementation of the bc algorithm
  – Clauses in $\Gamma$ are ordered by the programmer.
    • fact1.
    • fact2.
    • rule1.
  – Negative literals in each clause are ordered by the programmer.
    • goal :- subgoal1, subgoal2. (goal | ~subgoal1 | ~subgoal2)
  – The “pick” operations picks the first literal in $\psi$.
    • subgoal1, subgoal2
  – Search is carried out in depth-first search order.
    • subgoal1, subsubgoal1_1, subsubsubgoal1_1_1, ...