Propositional logic

Logical inference problem

Logical inference problem:
• Given:
  – a knowledge base KB (a set of sentences) and
  – a sentence $\alpha$ (called a theorem),
• Does a KB semantically entail $\alpha$? $KB \models \alpha$?
In other words: In all interpretations in which sentences in the KB are true, is also $\alpha$ true?

Question: Is there a procedure (program) that can decide this problem in a finite number of steps?
Answer: Yes. Logical inference problem for the propositional logic is decidable.
Solving logical inference problem

In the following:

How to design the procedure that answers:

\[ KB \models \alpha \]?

Three approaches:
• Truth-table approach
• Inference rules
• Conversion to the inverse SAT problem
  – Resolution-refutation

Truth-table approach

Problem: \( KB \models \alpha \) ?
• We need to check all possible interpretations for which the KB is true (models of KB) whether \( \alpha \) is true for each of them

Truth table:
• enumerates truth values of sentences for all possible interpretations (assignments of True/False values to propositional symbols)

Example:

<table>
<thead>
<tr>
<th>( P )</th>
<th>( Q )</th>
<th>( P \lor Q )</th>
<th>( P \leftrightarrow Q )</th>
<th>( (P \lor \neg Q) \land Q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

\[ \alpha = | KB \models \alpha \]
Truth-table approach

Problem: \( KB \models \alpha \) ?
- We need to check all possible interpretations for which the KB is true (models of KB) whether \( \alpha \) is true for each of them

Truth table:
- enumerates truth values of sentences for all possible interpretations (assignments of True/False to propositional symbols)

Example:

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>( P \lor Q )</th>
<th>( P \iff Q )</th>
<th>( (P \lor \neg Q) \land Q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

Truth-table approach

Problem: \( KB \models \alpha \) ?
- We need to check all possible interpretations for which the KB is true (models of KB) whether \( \alpha \) is true for each of them

Truth table:
- enumerates truth values of sentences for all possible interpretations (assignments of True/False to propositional symbols)

Example:

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>( P \lor Q )</th>
<th>( P \iff Q )</th>
<th>( (P \lor \neg Q) \land Q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

\( \checkmark \)
Inference rules approach.

\[ KB \models \alpha \ ? \]

Problem with the truth table approach:
- the truth table is exponential in the number of propositional symbols (we checked all assignments)
- KB is true on only a smaller subset

How to make the process more efficient?
- Solution: check only entries for which KB is True.
- The idea is implemented in the inference rules approach

Inference rules

Inference rules:
- Represent sound inference patterns repeated in inferences
- Can be used to generate new (sound) sentences from the existing ones

- Modus ponens

\[
\frac{A \Rightarrow B, \ A}{B} \quad \text{premise} \quad \text{conclusion}
\]

- If both sentences in the premise are true then conclusion is true.
Inference rules for logic

• **Modus ponens**

\[
\frac{A \Rightarrow B, \ A}{B}
\]

• If both sentences in the premise are true then conclusion is true.
• The **modus ponens** inference rule is **sound**.
  – We can prove this through the truth table.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>A \Rightarrow B</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

Inference rules for logic

• **And-elimination**

\[
\frac{A_1 \land A_2 \land \ldots \land A_n}{A_i}
\]

• **And-introduction**

\[
\frac{A_1, A_2, \ldots, A_n}{A_1 \land A_2 \land \ldots \land A_n}
\]

• **Or-introduction**

\[
\frac{A_i}{A_1 \lor A_2 \lor \ldots \lor A_i \lor A_n}
\]
Inference rules for logic

- Elimination of double negation
  \[ \neg \neg A \quad \Rightarrow \quad A \]

- Unit resolution
  \[ A \lor B, \quad \neg A \quad \Rightarrow \quad B \]

- Resolution
  \[ A \lor B, \quad \neg B \lor C \quad \Rightarrow \quad A \lor C \]

- All of the above inference rules are sound. We can prove this through the truth table, similarly to the modus ponens case.

Example. Inference rules approach.

KB: \( P \land Q \quad P \Rightarrow R \quad (Q \land R) \Rightarrow S \) \quad Theorem: S

1. \( P \land Q \)
2. \( P \Rightarrow R \)
3. \( (Q \land R) \Rightarrow S \)
Example. Inference rules approach.

**KB:** \( P \land Q \quad P \Rightarrow R \quad (Q \land R) \Rightarrow S \quad \text{Theorem: } S \)

1. \( P \land Q \)
2. \( P \Rightarrow R \)
3. \( (Q \land R) \Rightarrow S \)
4. \( P \)  
   **From 1 and And-elim**  
   \[ A_1 \land A_2 \land A_n \]
   \[ \quad A_i \]

5. \( R \)  
   **From 2,4 and Modus ponens**  
   \[ A \Rightarrow B, \quad A \]
   \[ \quad B \]
Example. Inference rules approach.

**KB:** $P \land Q \quad P \Rightarrow R \quad (Q \land R) \Rightarrow S$  
**Theorem:** $S$

1. $P \land Q$
2. $P \Rightarrow R$
3. $(Q \land R) \Rightarrow S$
4. $P$
5. $R$
6. $Q$  
   From 1 and And-elim
   $$ \frac{A_1 \land A_2 \land \ldots \land A_n}{A_i} $$

Example. Inference rules approach.

**KB:** $P \land Q \quad P \Rightarrow R \quad (Q \land R) \Rightarrow S$  
**Theorem:** $S$

1. $P \land Q$
2. $P \Rightarrow R$
3. $(Q \land R) \Rightarrow S$
4. $P$
5. $R$
6. $Q$
7. $(Q \land R)$  
   From 5,6 and And-introduction
   $$ \frac{A_1, A_2, \ldots, A_n}{A_1 \land A_2 \land \ldots \land A_n} $$
Example. Inference rules approach.

KB: \( P \land Q \quad P \Rightarrow R \quad (Q \land R) \Rightarrow S \)  
Theorem: S

1. \( P \land Q \)
2. \( P \Rightarrow R \)
3. \( (Q \land R) \Rightarrow S \)
4. \( P \)
5. \( R \)
6. \( Q \quad A \Rightarrow B, \quad A \quad \frac{A \Rightarrow B}{B} \)
7. \( (Q \land R) \)
8. \( S \quad \text{From 7,3 and Modus ponens} \)

Proved: S

Example. Inference rules approach.

KB: \( P \land Q \quad P \Rightarrow R \quad (Q \land R) \Rightarrow S \)  
Theorem: S

1. \( P \land Q \)
2. \( P \Rightarrow R \)
3. \( (Q \land R) \Rightarrow S \)
4. \( P \quad \text{From 1 and And-elim} \)
5. \( R \quad \text{From 2,4 and Modus ponens} \)
6. \( Q \quad \text{From 1 and And-elim} \)
7. \( (Q \land R) \quad \text{From 5,6 and And-introduction} \)
8. \( S \quad \text{From 7,3 and Modus ponens} \)

Proved: S
Logic inferences and search

- To show that theorem $\alpha$ holds for a KB
  - we may need to apply a number of sound inference rules

**Problem**: many possible rules to can be applied next

**Looks familiar?**

Truth table method (from the search perspective):
  - blind enumeration and checking

---

Logic inferences and search

**Inference rule method as a search problem:**

- **State**: a set of sentences that are known to be true
- **Initial state**: a set of sentences in the KB
- **Operators**: applications of inference rules
  - Allow us to add new sound sentences to old ones
- **Goal state**: a theorem $\alpha$ is derived from KB

**Logic inference:**

- **Proof**: A sequence of sentences that are immediate consequences of applied inference rules
- **Theorem proving**: process of finding a proof of theorem
Normal forms

Sentences in the propositional logic can be transformed into one of the normal forms. This can simplify the inferences.

Normal forms used:

Conjunctive normal form (CNF)

- conjunction of clauses (clauses include disjunctions of literals)

\[(A \lor B) \land (\neg A \lor \neg C \lor D)\]

Disjunctive normal form (DNF)

- Disjunction of terms (terms include conjunction of literals)

\[(A \land \neg B) \lor (\neg A \land C) \lor (C \land \neg D)\]

Conversion to a CNF

Assume: \(\neg(A \Rightarrow B) \lor (C \Rightarrow A)\)

1. Eliminate \(\Rightarrow, \Leftrightarrow\)

\[\neg(\neg A \lor B) \lor (\neg C \lor A)\]

2. Reduce the scope of signs through DeMorgan Laws and double negation

\[(A \land \neg B) \lor (\neg C \lor A)\]

3. Convert to CNF using the associative and distributive laws

\[(A \lor \neg C \lor A) \land (\neg B \lor \neg C \lor A)\]

and

\[(A \lor \neg C) \land (\neg B \lor \neg C \lor A)\]
Satisfiability (SAT) problem

Determine whether a sentence in the conjunctive normal form (CNF) is satisfiable (i.e. can evaluate to true)

\((P \lor Q \lor \neg R) \land (\neg P \lor \neg R \lor S) \land (\neg P \lor Q \lor \neg T)\)...

It is an instance of a constraint satisfaction problem:

- **Variables:**
  - Propositional symbols \((P, R, T, S)\)
  - Values: *True, False*
- **Constraints:**
  - Every conjunct must evaluate to true, at least one of the literals must evaluate to true
- **All techniques developed for CSPs can be applied to solve the logical inference problem. Why?**

Inference problem and satisfiability

**Inference problem:**
- we want to show that the sentence \(\alpha\) is entailed by KB

**Satisfiability:**
- The sentence is satisfiable if there is some assignment (interpretation) under which the sentence evaluates to true

**Connection:**

\[ KB \models \alpha \quad \text{if and only if} \quad (KB \land \neg \alpha) \text{ is unsatisfiable} \]

**Consequences:**
- inference problem is NP-complete
- programs for solving the SAT problem can be used to solve the inference problem
Universal inference rule: Resolution rule

Sometimes inference rules can be combined into a single rule

Resolution rule

- sound inference rule that works for CNF
- It is complete for **propositional logic (refutation complete)**

\[
A \lor B, \quad \neg A \lor C \\
B \lor C
\]

Initial obstacle:

- Repeated application of the resolution rule to a KB in CNF
  may fail to derive new valid sentences

Example:

We know: \((A \land B)\)  We want to show: \((A \lor B)\)

Resolution rule fails to derive it (**incomplete ??**)

A trick to make things work:

- proof by contradiction
  - Disproving: \(KB, \neg \alpha\)
  - Proves the entailment \(KB \models \alpha\)
Resolution algorithm

Algorithm:
• Convert KB to the CNF form;
• Apply iteratively the resolution rule starting from $KB, \neg \alpha$ (in CNF form)
• Stop when:
  – Contradiction (empty clause) is reached:
    • $A, \neg A \rightarrow Q$
    • proves entailment.
  – No more new sentences can be derived
    • disproves it.

Example. Resolution.

$KB$: $(P \land Q) \land (P \Rightarrow R) \land [(Q \land R) \Rightarrow S]$

Step 1. convert KB to CNF:
• $P \land Q \quad \Rightarrow \quad P \land Q$
• $P \Rightarrow R \quad \Rightarrow \quad (\neg P \lor R)$
• $(Q \land R) \Rightarrow S \quad \Rightarrow \quad (\neg Q \lor \neg R \lor S)$

$KB$: $P \quad Q \quad (\neg P \lor R) \quad (\neg Q \lor \neg R \lor S)$

Step 2. Negate the theorem to prove it via refutation
$S \quad \Rightarrow \quad \neg S$

Step 3. Run resolution on the set of clauses
$P \quad Q \quad (\neg P \lor R) \quad (\neg Q \lor \neg R \lor S) \quad \neg S$
Example. Resolution.

**KB:** \((P \land Q) \land (P \Rightarrow R) \land [(Q \land R) \Rightarrow S]\)  \quad \text{Theorem: } S

\[
P \quad Q \quad (\neg P \lor R) \quad (\neg Q \lor \neg R \lor S) \quad \neg S
\]
Example. Resolution.

**KB:** \((P \land Q) \land (P \Rightarrow R) \land [(Q \land R) \Rightarrow S]\)  
**Theorem:** \(S\)

\[
\begin{array}{c}
P & Q & (\neg P \lor R) & (\neg Q \lor R \lor S) & \neg S \\
\mid \\
R & (\neg R \lor S) \\
\end{array}
\]
Example. Resolution.

KB: \((P \land Q) \land (P \Rightarrow R) \land [(Q \land R) \Rightarrow S]\)

Theorem: \(S\)

\[
\begin{array}{c}
P \quad Q \quad (\neg P \lor R) \quad (\neg Q \lor \neg R \lor S) \quad \neg S \\
\downarrow R \\
\quad \quad (\neg R \lor S) \\
\downarrow S \\
\text{Contradiction} \quad \{\} \\
\text{Proved: } S
\end{array}
\]

SAT solvers

- SAT is an instance of CSP problem

CSP definition
- Variables:
  - Propositional symbols
  - Values: True, False
- Constraints:
  - Every conjunct must evaluate to true, at least one of the literals must evaluate to true
- Example:
  \[
  (P \lor Q \lor \neg R) \land (\neg P \lor \neg R \lor S) \land (\neg P \lor Q \lor \neg T)\ldots
  \]
- Variables: P,Q,R,S,T
SAT solvers

- Solving CSP by backtracking
- Idea:
  - Repeatedly select an unassigned variable
  - Select its value
  - Check if the constraints are not violated
  - If yes backtrack to the previous choice
  - If the all choices are exhausted backtrack to the higher level

- Goes under the name: **Davis Putnam algorithm**