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 $\alpha$ also true?
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 $\alpha$? $KB \models \alpha$

Approaches to solve the logical inference problem:
- Truth-table approach
- Inference rules
- Conversion to SAT
  - Resolution refutation

Properties of inference solutions

- Truth-table approach
  - Blind
  - Exponential in the number of variables
- Inference rules
  - More efficient
  - Many inference rules to cover logic
- Conversion to SAT - Resolution refutation
  - More efficient
  - Sentences must be converted into CNF
  - One rule – the resolution rule - is sufficient to perform all inferences
**KB in restricted forms**

If the sentences in the KB are restricted to some special forms, some of the sound inference rules may become complete.

**Example:**
- **Horn form (Horn normal form)**
  - a clause with at most one positive literal
    \[(A \lor \neg B) \land (\neg A \lor \neg C \lor D)\]

  Can be written also as:
  \[(B \Rightarrow A) \land ((A \land C) \Rightarrow D)\]

- **Two inference rules that are sound and complete for inferences on propositional symbols for KBs in the Horn normal form:**
  - Resolution
  - Modus ponens

---

**KB in Horn form**

- **Horn form:** a clause with at most one positive literal
  \[(A \lor \neg B) \land (\neg A \lor \neg C \lor D)\]

- **Note:** Not all sentences in propositional logic can be converted into the Horn form.

- **KB in Horn normal form:**
  - Two types of propositional statements:
    - **Rules**
      \[\neg B_1 \lor \neg B_2 \lor \ldots \neg B_k \lor A\]
      \[\equiv\]
      \[\neg (B_1 \land B_2 \land \ldots B_k) \lor A\]
      \[\equiv\]
      \[(B_1 \land B_2 \land \ldots B_k \Rightarrow A)\]
  - Propositional symbols: facts \(B\)
Why KB in Horn form is useful?

KB in Horn normal form:

- **Rules (in implicative form):** If then statements known to be true
  \[ (B_1 \land B_2 \land \ldots B_k \implies A) \]
  
  If
  1. The stain of the organism is gram-positive, and
  2. The morphology of the organism is coccus, and
  3. The growth conformation of the organism is chains
  Then
  the identity of the organism is streptococcus

- **Facts** = propositions known to be true
  Examples: \( B_1 \)
  or
  The stain of the organism is gram-positive

**Inferences:** let us infer new true propositions, such as \( A \), or
the identity of the organism is streptococcus in the rule conclusion
These are referred as inferences on propositional symbols

KB in Horn form

- **Application of the resolution rule:**
  - Infers new facts from previous facts
  \[
  \frac{(A \lor \neg B), B}{A} \quad \frac{(A \lor \neg B), (B \lor \neg C)}{(A \lor C)}
  \]
  
  - Resolution is **sound and complete** for inferences on
  propositional symbols for KB in the Horn normal form
  (clausal form)

- **Similarly, modus ponens is sound and complete** when the
  HNF is written in the implicative form
Complexity of inferences for KBs in HNF

Question:
How efficient the inferences in the HNF can be?

Answer:
Inference on propositional symbols →
Procedures linear in the size of the KB in the Horn form exist.

- Size of a clause: the number of literals it contains.
- Size of the KB in the HNF: the sum of the sizes of its elements.

Example:

\[ A, B, (A \land B \Rightarrow C), (C \Rightarrow D), (C \Rightarrow E), (E \land F \Rightarrow G) \]

or

\[ A, B, (\neg A \lor \neg B \lor C), (\neg C \lor D), (\neg C \lor E), (\neg E \lor \neg F \lor G) \]

The size is: 12

Complexity of inferences for KBs in HNF

How to do the inference on propositional symbols? If the HNF (is in the clausal form) we can apply resolution.

\[ A, B, (\neg A \lor \neg B \lor C), (\neg C \lor D), (\neg C \lor E), (\neg E \lor \neg F \lor G) \]

\[ \neg C \lor D, (\neg C \lor E), (\neg E \lor \neg F \lor G) \]

\[ \neg B \lor C \]

\[ C \]

\[ D \]

\[ E \]
Complexity of inferences for KBs in HNF

How to do the inference on propositional symbols? If the HNF (is in the clausal form) we can apply resolution.

\[ A, B, (\neg A \lor \neg B \lor C), (\neg C \lor D), (\neg C \lor E), (\neg E \lor \neg F \lor G) \]

Inferred propositional symbols (or facts)

Complexity of inferences for KBs in HNF

**Features:**
- Every resolution is a **positive unit resolution**; that is, a resolution in which one clause is a positive unit clause (i.e., a proposition symbol).

\[ A, B, (\neg A \lor \neg B \lor C), (\neg C \lor D), (\neg C \lor E), (\neg E \lor \neg F \lor G) \]
Complexity of inferences for KBs in HNF

Features:
• At each resolution, the input clause which is not a unit clause is a logical consequence of the result of the resolution. (Thus, the input clause may be deleted upon completion of the resolution operation.)

\[ A, B, (\neg A \lor \neg B \lor C), (\neg C \lor D), (\neg C \lor E), (\neg E \lor \neg F \lor G) \]
Complexity of inferences for KBs in HNF

Features:
• Following this deletion, the size of the KB (the sum of the lengths of the remaining clauses) is one less than it was before the operation.)

Now let us see one more step …
Complexity of inferences for KBs in HNF

Features:
• Following the deletion, the size of the KB (the sum of the lengths of the remaining clauses) is one less than it was before the operation. Now let us see one more step …

A, B, (¬A ∨ B ∨ C), (¬C ∨ D), (¬C ∨ E), (¬E ∨ ¬F ∨ G)

Complexity of inferences for KBs in HNF

Features:
• If n is the size of the KB, then at most n positive unit resolutions may be performed on it.

A, B, (¬A ∨ B ∨ C), (¬C ∨ D), (¬C ∨ E), (¬E ∨ ¬F ∨ G)
Complexity of inferences for KBs in HNF

A linear time resolution algorithm:
• The number of positive unit resolutions is limited to the size of the KB (n)

• But to assure overall linear time we need to access each proposition in a constant time:
  • Data structures indexed by proposition names may be accessed in constant time. (This is possible if the proposition names are numbers in a range (e.g., 1..n), so that array lookup is the access operation.
  • If propositions are accessed by name, then a symbol table is necessary, and the algorithm will run in time $O(n \cdot \log(n))$.

Forward and backward chaining

Two inference procedures based on modus ponens for Horn KBs:
• Forward chaining
  Idea: Whenever the premises of a rule are satisfied, infer the conclusion. Continue with rules that became satisfied.

• Backward chaining (goal reduction)
  Idea: To prove the fact that appears in the conclusion of a rule prove the premises of the rule. Continue recursively.

Both procedures are complete for KBs in the Horn form !!!
Forward chaining example

- **Forward chaining**
  
  **Idea:** Whenever the premises of a rule are satisfied, infer the conclusion. Continue with rules that became satisfied.

Assume the KB with the following rules and facts:

**KB:**

R1: \( A \land B \Rightarrow C \)

R2: \( C \land D \Rightarrow E \)

R3: \( C \land F \Rightarrow G \)

**Facts:**

F1: \( A \)

F2: \( B \)

F3: \( D \)

**Theorem:** \( E \) ?

---

Forward chaining example

**Theorem:** \( E \)

**KB:**

R1: \( A \land B \Rightarrow C \)

R2: \( C \land D \Rightarrow E \)

R3: \( C \land F \Rightarrow G \)

**Facts:**

F1: \( A \)

F2: \( B \)

F3: \( D \)
Forward chaining example

Theorem: $E$

KB:

R1: $A \land B \Rightarrow C$
R2: $C \land D \Rightarrow E$
R3: $C \land F \Rightarrow G$

F1: $A$
F2: $B$
F3: $D$

Rule R1 is satisfied.

F4: $C$

Rule R2 is satisfied.

F5: $E$
Forward chaining

- Efficient implementation: linear in the size of the KB
- Example:

\[
\begin{align*}
P & \Rightarrow Q \\
L \land M & \Rightarrow P \\
B \land L & \Rightarrow M \\
A \land P & \Rightarrow L \\
A \land B & \Rightarrow L \\
A & \\
B &
\end{align*}
\]

Forward chaining

- Count the number of facts in the antecedent of the rule

\[
\begin{align*}
P & \Rightarrow Q \\
L \land M & \Rightarrow P \\
B \land L & \Rightarrow M \\
A \land P & \Rightarrow L \\
A \land B & \Rightarrow L \\
A & \\
B &
\end{align*}
\]
Forward chaining

- Inferred facts decrease the count

\[ P \implies Q \]
\[ L \land M \implies P \]
\[ B \land L \implies M \]
\[ A \land P \implies L \]
\[ A \land B \implies L \]
\[ A \]
\[ B \]

Forward chaining

- New facts can be inferred when the count associated with a rule becomes 0

\[ P \implies Q \]
\[ L \land M \implies P \]
\[ B \land L \implies M \]
\[ A \land P \implies L \]
\[ A \land B \implies L \]
\[ A \]
\[ B \]
Forward chaining

- $P \implies Q$
- $L \land M \implies P$
- $B \land L \implies M$
- $A \land P \implies L$
- $A \land B \implies L$
- $A$
- $B$
Forward chaining

- $P \Rightarrow Q$
- $L \land M \Rightarrow P$
- $B \land L \Rightarrow M$
- $A \land P \Rightarrow L$
- $A \land B \Rightarrow L$
- $A$
- $B$

\[
\begin{array}{c}
Q & 0 \\

P & 0 \\

A & 0 \\

B & 0 \\

\end{array}
\]
Backward chaining example

- Backward chaining is more focused:
  - tries to prove the theorem only

Theorem: E

KB:

R1: \( A \land B \Rightarrow C \)
R2: \( C \land D \Rightarrow E \)
R3: \( C \land F \Rightarrow G \)
F1: \( A \)
F2: \( B \)
F3: \( D \)

M. Hauskrecht
Backward chaining

- Efficient implementation

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]
Backward chaining

• Efficient implementation

\[
P \Rightarrow Q
\]

\[
L \land M \Rightarrow P
\]

\[
B \land L \Rightarrow \overline{M}
\]

\[
A \land P \Rightarrow L
\]

\[
A \land B \Rightarrow L
\]

\[
A
\]

\[
B
\]
Backward chaining

• Efficient implementation

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]

\[ A \]

\[ B \]
Backward chaining

- Efficient implementation

\[
P \Rightarrow Q \\
L \land M \Rightarrow P \\
B \land L \Rightarrow M \\
A \land P \Rightarrow L \\
A \land B \Rightarrow L
\]

M. Hauskrecht
Backward chaining

- Efficient implementation

\[ P \implies Q \quad \text{\color{green}{\leftarrow}} \]
\[ (L \land M) \implies P \quad \text{\color{red}{\leftarrow}} \]
\[ B \land L \implies M \quad \text{\color{red}{\leftarrow}} \]
\[ A \land P \implies L \quad \text{\color{green}{\leftarrow}} \]
\[ A \land B \implies \neg L \quad \text{\color{red}{\leftarrow}} \]
\[ A \quad \text{\color{red}{\leftarrow}} \]
\[ B \quad \text{\color{red}{\leftarrow}} \]
Forward vs Backward chaining

- **FC is data-driven**, automatic, unconscious processing,
  - e.g., object recognition, routine decisions

- May do lots of work that is irrelevant to the goal

- **BC is goal-driven**, appropriate for problem-solving,
  - e.g., Where are my keys? How do I get into a PhD program?

- Complexity of BC can be **much less** than **linear in size of KB**

KB agents based on propositional logic

- Propositional logic allows us to build **knowledge-based agents** capable of answering queries about the world by inferring new facts from the known ones

- **Example**: an agent for diagnosis of a bacterial disease

  **Facts:**
  - The stain of the organism is gram-positive
  - The growth conformation of the organism is chains

  **Rules:**
  - (If) The stain of the organism is gram-positive \(\land\)
    The morphology of the organism is coccus \(\land\)
    The growth conformation of the organism is chains
  - (Then) \(\implies\) The identity of the organism is streptococcus