Propositional logic.

Horn clauses

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

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?
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

---

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

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

- **Two inference rules that are sound and complete with respect to propositional symbols** for KBs in the Horn normal form:
  - Resolution (positive unit resolution)
  - Modus ponens
KB in Horn form

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

- Not all sentences in propositional logic can be converted into the Horn form

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

---

KB in Horn form

- **Application of the modus ponens**:
  - Infers new facts from previous facts

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

\[
\frac{(B_1 \land B_2 \land \ldots \land B_k \Rightarrow A), B_1, B_2, \ldots, B_k}{A}
\]

- Modus ponens is sound and complete with respect to propositional symbols for the KBs in the Horn normal form
Complexity of inferences for KBs in HNF

Question:
How efficient the inferences in HNF can be?

Answer:
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

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

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

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

- 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:

- 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.)

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

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.

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

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

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

Linear time algorithm:
- The number of positive unit resolutions is limited to the size of the formula \((n)\)
- 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 number 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:

<table>
<thead>
<tr>
<th>KB:</th>
<th>R1: $A \land B \Rightarrow C$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R2: $C \land D \Rightarrow E$</td>
</tr>
<tr>
<td></td>
<td>R3: $C \land F \Rightarrow G$</td>
</tr>
<tr>
<td>F1:</td>
<td>$A$</td>
</tr>
<tr>
<td>F2:</td>
<td>$B$</td>
</tr>
<tr>
<td>F3:</td>
<td>$D$</td>
</tr>
<tr>
<td>Theorem:</td>
<td>$E$ ?</td>
</tr>
</tbody>
</table>
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: \( C \)
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

- Runs in time linear in the number of literals in the Horn formulae

```language=prolog
function PL-FC-ENTAILS?(KB, q) returns true or false
  local variables: count, a table, indexed by clause, initially the number of premises
                  inferred, a table, indexed by symbol, each entry initially false
                  agenda, a list of symbols, initially the symbols known to be true
  while agenda is not empty do
    p ← Pop(agenda)
    unless inferred[p] do
      inferred[p] ← true
      for each Horn clause c in whose premise p appears do
        decrement count[c]
        if count[c] = 0 then do
          if Head[c] = q then return true
          Push(Head[c], agenda)
    return false
```

Forward chaining

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

agenda
Forward chaining

•

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

\[ Q \]
\[ 1 \]
\[ 2 \]
\[ M \]
\[ 2 \]
\[ L \]
\[ 1 \]
\[ 1 \]
\[ A \]
\[ B \]

inferred

add to agenda

inferred
Forward chaining

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

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

Backward chaining example

KB: \[ A \land B \Rightarrow C \]
R2: \[ C \land D \Rightarrow E \]
R3: \[ C \land F \Rightarrow G \]
F1: \[ A \]
F2: \[ B \]
F3: \[ D \]

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

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

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

Backward chaining

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

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

Backward chaining

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

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

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

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]
Backward 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 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) The identity of the organism is streptococcus

---

Example. Animal identification system.

11. If the animal has hair then it is a mammal
12. If the animal gives milk then it is a mammal
13. If the animal has feathers then it is a bird
14. If the animal flies and it lays eggs then it is a bird
15. If the animal is a mammal and it eats meat then it is a carnivore
16. If the animal is a mammal and it has pointed teeth and it has claws and its eyes point forward then it is a carnivore
17. If the animal is a mammal and it has hoofs then it is an ungulate
18. If the animal is a mammal and it chews cud then it is an ungulate and it is even-toed
19. If the animal is a carnivore and it has a tawny color and it has dark spots then it is a cheetah
20. If the animal is a carnivore and it has a tawny color and it has black strips then it is a tiger
21. If the animal is an ungulate and it has long legs and it has a long neck and it has a tawny color and it has dark spots then it is a giraffe
22. If the animal is an ungulate and it has a white color and it has black stripes then it is a zebra
23. If the animal is a bird and it does not fly and it has long legs and it has a long neck and it is black and white then it is an ostrich,
24. If the animal is a bird and it does not fly and it swims and it is black and white then it is a penguin
25. If the animal is a bird and it is a good flyer then it is an albatross.