First-order logic

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

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
**KBs in the Horn form**

**Horn clause:**

a special type of clause with at most one positive literal

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

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

**KB with statements in the Horn form:**

- Two types of propositional statements:
  - Implications: called **rules** \((B \Rightarrow A)\)
  - Propositional symbols: **facts** \(B\)

**Modus ponens:**

- is the “universal “(complete) rule for the KB with sentences in the Horn form

\[
\begin{align*}
A \Rightarrow B, \quad & A \\
\hline 
& A_1 \land A_2 \land \ldots \land A_k \Rightarrow B, \quad A_1, A_2, \ldots A_k \\
& B
\end{align*}
\]

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

F1: \( A \)
F2: \( B \)
F3: \( D \)

Theorem: \( E \) ?

Rule R1 is satisfied.
F4: \( C \)
**Forward chaining example**

**Theorem:** \( E \)

**KB:**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>( A \land B \implies C )</td>
</tr>
<tr>
<td>R2</td>
<td>( C \land D \implies E )</td>
</tr>
<tr>
<td>R3</td>
<td>( C \land F \implies G )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fact</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>( A )</td>
</tr>
<tr>
<td>F2</td>
<td>( B )</td>
</tr>
<tr>
<td>F3</td>
<td>( D )</td>
</tr>
</tbody>
</table>

**Rule R1 is satisfied.**

<table>
<thead>
<tr>
<th>Fact</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F4</td>
<td>( C )</td>
</tr>
</tbody>
</table>

**Rule R2 is satisfied.**

<table>
<thead>
<tr>
<th>Fact</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F5</td>
<td>( E )</td>
</tr>
</tbody>
</table>

---

**Backward chaining example**

KB:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>( A \land B \implies C )</td>
</tr>
<tr>
<td>R2</td>
<td>( C \land D \implies E )</td>
</tr>
<tr>
<td>R3</td>
<td>( C \land F \implies G )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fact</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>( A )</td>
</tr>
<tr>
<td>F2</td>
<td>( B )</td>
</tr>
<tr>
<td>F3</td>
<td>( D )</td>
</tr>
</tbody>
</table>

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

---

CS 1571 Intro to AI
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

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)  \( \Rightarrow \)  
  - The identity of the organism is streptococcus
Limitations of propositional logic

World we want to represent and reason about consists of a number of objects with variety of properties and relations among them

**Propositional logic:**
- Represents statements about the world without reflecting this structure and without modeling these entities explicitly

**Consequence:**
- Some knowledge is hard or impossible to encode in the propositional logic.
- Two cases that are hard to represent:
  - **Statements about similar objects, relations**
  - **Statements referring to groups of objects.**

---

Limitations of propositional logic

- **Statements about similar objects and relations needs to be enumerated**
- **Example:** Seniority of people domain

**Assume we have:**
- John is older than Mary
  - Mary is older than Paul

**To derive John is older than Paul** we need:
- John is older than Mary \( \land \) Mary is older than Paul
  \[ \Rightarrow \text{John is older than Paul} \]

**Assume we add another fact:**
- Jane is older than Mary

**To derive Jane is older than Paul** we need:
- Jane is older than Mary \( \land \) Mary is older than Paul
  \[ \Rightarrow \text{Jane is older than Paul} \]

**Problem:** KB grows large
**Limitations of propositional logic**

- **Statements about similar objects and relations needs to be enumerated**
- **Example:** Seniority of people domain
  
  For inferences we need:
  
  \[
  \begin{align*}
  John & \text{ is older than Mary} \land \text{ Mary is older than Paul} \\
  \Rightarrow & \text{ John is older than Paul} \\
  Jane & \text{ is older than Mary} \land \text{ Mary is older than Paul} \\
  \Rightarrow & \text{ Jane is older than Paul}
  \end{align*}
  \]

- **Problem:** if we have many people and facts about their seniority we need to represent many rules like this to allow inferences
- **Possible solution:** introduce variables

\[
\begin{align*}
\text{PersA} & \text{ is older than PersB} \land \text{ PersB is older than PersC} \\
\Rightarrow & \text{ PersA is older than PersC}
\end{align*}
\]
Limitations of propositional logic

- **Statements referring to groups of objects require exhaustive enumeration of objects**
- **Example:**
  Assume we want to express *Every student likes vacation*
  Doing this in propositional logic would require to include statements about every student
  
  \[
  John \text{ likes vacation} \land \\
  Mary \text{ likes vacation} \land \\
  Ann \text{ likes vacation} \land \\
  \cdots
  \]

- **Solution:** Allow quantification in statements

First-order logic (FOL)

- More expressive than **propositional logic**
- **Eliminates deficiencies of PL by:**
  - Representing objects, their properties, relations and statements about them;
  - Introducing variables that refer to an arbitrary objects and can be substituted by a specific object
  - Introducing quantifiers allowing quantification statements over objects without the need to represent each of them separately

- **Predicate logic:** first-order logic without the quantification fix
Logic

Logic is defined by:

• A set of sentences
  – A sentence is constructed from a set of primitives according to syntax rules.

• A set of interpretations
  – An interpretation gives a semantic to primitives. It associates primitives with objects, values in the real world.

• The valuation (meaning) function \( V \)
  – Assigns a truth value to a given sentence under some interpretation

\[
V : \text{sentence} \times \text{interpretation} \rightarrow \{\text{True}, \text{False}\}
\]

First-order logic. Syntax.

Term - syntactic entity for representing objects

Terms in FOL:

• Constant symbols:
  – E.g. John, France, car89

• Variables:
  – E.g. \( x, y, z \)

• Functions applied to one or more terms
  – E.g. father-of (John)
    
    father-of(father-of(John))
First order logic. Syntax.

Sentences in FOL:

- Atomic sentences:
  - A predicate symbol applied to 0 or more terms
  
  Examples:
  
  Red(car12),
  Sister(Amy, Jane);
  Manager(father-of(John));

  - \( t_1 = t_2 \) equivalence of terms

  Example:
  
  \( John = father-of(Peter) \)


First order logic. Syntax.

Sentences in FOL:

- Complex sentences:
  
  Assume \( \phi, \psi \) are sentences. Then:

  - \( (\phi \land \psi) \), \( (\phi \lor \psi) \), \( (\phi \Rightarrow \psi) \), \( (\phi \iff \psi) \), \( \neg \psi \)

  and

  - \( \forall x \phi \), \( \exists y \phi \)

  are sentences

Symbols \( \exists, \forall \) - stand for the existential and the universal quantifier
Semantics. Interpretation.

An interpretation $I$ is defined by a **domain** and a **mapping**

- **domain D**: a set of objects in the world we represent;
  domain of discourse;

**An interpretation $I$ maps:**

- Constant symbols to objects in $D$
  $I(John) = \circled{John}$
- Predicate symbols to relations, properties on $D$
  $I(brother) = \{ \langle \circled{John} \circled{Paul} \rangle ; \langle \circled{Paul} \circled{John} \rangle ; \ldots \}$
- Function symbols to functional relations on $D$
  $I(father-of) = \{ \langle \circled{John} \rightarrow \circled{Paul} \rangle ; \langle \circled{Paul} \rightarrow \circled{John} \rangle ; \ldots \}$

Semantics of sentences.

**Meaning (evaluation) function**:

$V : \text{sentence} \times \text{interpretation} \rightarrow \{ \text{True} , \text{False} \}$

A **predicate** $\text{predicate}(\text{term-1}, \text{term-2}, \text{term-3}, \text{term-n})$ is true for the interpretation $I$, iff the objects referred to by $\text{term-1}, \text{term-2}, \text{term-3}, \text{term-n}$ are in the relation referred to by $\text{predicate}$

$I(John) = \circled{John} \quad I(Paul) = \circled{Paul}$

$I(brother) = \{ \langle \circled{John} \circled{Paul} \rangle ; \langle \circled{Paul} \circled{John} \rangle ; \ldots \}$

$\text{brother}(John, Paul) = \langle \circled{John} \circled{Paul} \rangle \quad \text{in } I(brother)$

$V(\text{brother}(John, Paul), I) = \text{True}$
Semantics of sentences.

- **Equality**
  \[ V(\text{term-1} = \text{term-2}, I) = \text{True} \]
  Iff \( I(\text{term-1}) = I(\text{term-2}) \)

- **Boolean expressions: standard**
  E.g.  \[ V(\text{sentence-1} \lor \text{sentence-2}, I) = \text{True} \]
  Iff \( V(\text{sentence-1}, I) = \text{True} \) or \( V(\text{sentence-2}, I) = \text{True} \)

- **Quantifications**
  \[ V(\forall x \phi , I) = \text{True} \]  
  Iff for all \( d \in D \)  \[ V(\phi , I[d/x]) = \text{True} \]
  \[ V(\exists x \phi , I) = \text{True} \]  
  Iff there is a \( d \in D \), s.t.  \[ V(\phi , I[d/x]) = \text{True} \]

Examples of sentences with quantifiers

- **Universal quantification**
  
  All Upitt students are smart
  
  \( \forall x \text{ student}(x) \land \text{at}(x, \text{Upitt}) \Rightarrow \text{smart}(x) \)

  Typically the universal quantifier connects with implication

- **Existential quantification**
  
  Someone at CMU is smart
  
  \( \exists x \text{ at}(x, \text{CMU}) \land \text{smart}(x) \)

  Typically the existential quantifier connects with conjunction
Order of quantifiers

• Order of quantifiers of the same type does not matter
  
  For all $x$ and $y$, if $x$ is a parent of $y$ then $y$ is a child of $x$
  
  $\forall x, y \ \text{parent} \ (x, y) \Rightarrow \text{child} \ (y, x)$
  
  $\forall y, x \ \text{parent} \ (x, y) \Rightarrow \text{child} \ (y, x)$

• Order of different quantifiers changes the meaning
  
  $\forall x \exists y \ \text{loves} \ (x, y)$

---

Order of quantifiers

• Order of quantifiers of the same type does not matter
  
  For all $x$ and $y$, if $x$ is a parent of $y$ then $y$ is a child of $x$
  
  $\forall x, y \ \text{parent} \ (x, y) \Rightarrow \text{child} \ (y, x)$
  
  $\forall y, x \ \text{parent} \ (x, y) \Rightarrow \text{child} \ (y, x)$

• Order of different quantifiers changes the meaning
  
  $\forall x \exists y \ \text{loves} \ (x, y)$
  
  Everybody loves somebody
  
  $\exists y \forall x \ \text{loves} \ (x, y)$
Order of quantifiers

- **Order of quantifiers of the same type does not matter**
  
  *For all x and y, if x is a parent of y then y is a child of x*
  
  \[
  \forall x, y \text{ parent } (x, y) \Rightarrow \text{ child } (y, x)
  \]
  
  \[
  \forall y, x \text{ parent } (x, y) \Rightarrow \text{ child } (y, x)
  \]

- **Order of different quantifiers changes the meaning**

  \[
  \forall x \exists y \text{ loves } (x, y)
  \]

  *Everybody loves somebody*

  \[
  \exists y \forall x \text{ loves } (x, y)
  \]

  *There is someone who is loved by everyone*

Connections between quantifiers

*Everyone likes ice cream*

\[
\forall x \text{ likes } (x, \text{IceCream })
\]

Is it possible to convey the same meaning using an existential quantifier?

*There is no one who does not like ice cream*

\[
\neg \exists x \neg \text{likes } (x, \text{IceCream })
\]

A universal quantifier in the sentence can be expressed using an existential quantifier !!!
Connections between quantifiers

Someone likes ice cream

\[ \exists x \text{ likes} (x, \text{IceCream}) \]

Is it possible to convey the same meaning using a universal quantifier?

Not everyone does not like ice cream

\[ \neg \forall x \neg \text{likes} (x, \text{IceCream}) \]

An existential quantifier in the sentence can be expressed using a universal quantifier!!!

Representing knowledge in FOL

Example:

Kinship domain

- **Objects**: people
  
  \[ \text{John, Mary, Jane, …} \]
- **Properties**: gender
  
  \[ \text{Male} (x), \text{Female} (x) \]
- **Relations**: parenthood, brotherhood, marriage
  
  \[ \text{Parent} (x, y), \text{Brother} (x, y), \text{Spouse} (x, y) \]
- **Functions**: mother-of (one for each person x)
  
  \[ \text{MotherOf} (x) \]
Kinship domain in FOL

**Relations between predicates and functions:** write down what we know about them; how relate to each other.

- Male and female are disjoint categories
  \[ \forall x \ Male (x) \iff \neg Female (x) \]

- Parent and child relations are inverse
  \[ \forall x, y \ Parent (x, y) \iff Child (y, x) \]

- A grandparent is a parent of parent
  \[ \forall g, c \ Grandparent (g, c) \iff \exists p \ Parent (g, p) \land Parent (p, c) \]

- A sibling is another child of one’s parents
  \[ \forall x, y \ Sibling (x, y) \iff (x \neq y) \land \exists p \ Parent (p, x) \land Parent (p, y) \]

- And so on ….