ARTIFICIAL INTELLIGENCE

Huỳnh Tấn Đạt
Email: htdat@cse.hcmut.edu.vn
Home Page: http://www.cse.hcmut.edu.vn/~htdat/
Ch5: Using Predicate Logic

- Representing simple facts in Logic
- Representing Instance and Isa relationships
- Make an exception
- Computable functions and predicates
- Resolution
- Natural deduction
Representing Simple Facts in Logic

- Using propositional logic
  - Represent real-world facts as logical propositions written as well-formed formulas (wff’s)
  - Example 1:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>It is raining</td>
<td>RAINING</td>
</tr>
<tr>
<td>It is sunny</td>
<td>SUNNY</td>
</tr>
<tr>
<td>It is Windy</td>
<td>WINDY</td>
</tr>
<tr>
<td>If it is raining, then it is not sunny</td>
<td>RAINING → ¬SUNNY</td>
</tr>
</tbody>
</table>
Using propositional logic

1. Modus Ponens (MP): 
   \[ A, A \rightarrow B \models B \]

2. Modus Tollens (MT): 
   \[ A \rightarrow B, \neg B \models \neg A \]

3. Conjunction (Conj): 
   \[ A, B \models A \land B \]

4. Simplification (Simp): 
   \[ A \land B \models A, B \]
Representing Simple Facts in Logic

5. Addition (Add):
\[ A \models A \lor B \]

6. Disjunctive Syllogism (DS):
\[ A \lor B, \neg A \models B \]

7. Hypothetical Syllogism (HS):
\[ A \rightarrow B, B \rightarrow C \models A \rightarrow C \]
Representing Simple Facts in Logic

– Example 2:

<table>
<thead>
<tr>
<th>Socrates is a man</th>
<th>SOCRATESMAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plato is a man</td>
<td>PLATOMAN</td>
</tr>
<tr>
<td>All men are mortal</td>
<td>MORTALMAN</td>
</tr>
</tbody>
</table>

=> Can’t draw any conclusion about similarities between Socrates and Plato

=> Can’t capture the relationship between any individual being a man and that individual being a mortal
Representing Simple Facts in Logic

- Using predicate logic

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Socrates is a man</td>
<td>man(socrates)</td>
</tr>
<tr>
<td>Plato is a man</td>
<td>man(plato)</td>
</tr>
<tr>
<td>All men are mortal</td>
<td>∀X (man(X) → mortal(X))</td>
</tr>
</tbody>
</table>

=> mortal(socrates)
Representing Simple Facts in Logic

- Propositional logic vs. predicate logic
  - Using propositional logic
    • Theorem proving is decidable
    • Cannot represent objects and quantification
  - Using predicate logic
    • Can represent objects and quantification
    • Theorem proving is semi-decidable
Representing Simple Facts in Logic

1. Marcus was a man.
2. Marcus was a Pompeian.
3. All Pompeians were Romans.
4. Caesar was a ruler.
5. All Romans were either loyal to Caesar or hated him.
6. Every one is loyal to someone.
7. People only try to assassinate rulers they are not loyal to.
8. Marcus tried to assassinate Caesar
Representing Simple Facts in Logic

1. Marcus was a man.
   \[\text{man(Marcus)}\]

2. Marcus was a Pompeian.
   \[\text{Pompeian(Marcus)}\]

3. All Pompeians were Romans.
   \[\forall x: \text{Pompeian}(x) \rightarrow \text{Roman}(x)\]

4. Caesar was a ruler.
   \[\text{ruler(Caesar)}\]
5. All Romans were either loyal to Caesar or hated him.

inclusive-or

$$\forall x: \text{Roman}(x) \rightarrow \text{loyalto}(x, \text{Caesar}) \lor \text{hate}(x, \text{Caesar})$$

exclusive-or (XOR)

$$\forall x: \text{Roman}(x) \rightarrow (\text{loyalto}(x, \text{Caesar}) \land \neg \text{hate}(x, \text{Caesar}))$$

$$\lor (\neg \text{loyalto}(x, \text{Caesar}) \land \text{hate}(x, \text{Caesar}))$$
Representing Simple Facts in Logic

6. Every one is loyal to someone.
   \[ \forall x: \exists y: \text{loyalto}(x, y) \quad \exists y: \forall x: \text{loyalto}(x, y) \]
   \[ \forall x: \exists y: \text{loyalto}(y, x) \]

7. People only try to assassinate rulers they are not loyal to.
   \[ \forall x: \forall y: \text{person}(x) \land \text{ruler}(y) \land \text{tryassassinate}(x, y) \rightarrow \neg \text{loyalto}(x, y) \]

8. Marcus tried to assassinate Caesar.
   \[ \text{tryassassinate}(\text{Marcus, Caesar}) \]
Representing Simple Facts in Logic

- Was Marcus loyal to Caesar?
  
  man(Marcus)
  ruler(Caesar)
  tryassassinate(Marcus, Caesar)
  \[\downarrow\]
  \[\forall x: \text{man}(x) \rightarrow \text{person}(x)\]
  
  \neg\text{loyalto}(Marcus, Caesar)
Many English sentences are ambiguous.

There is often a choice of how to represent knowledge.

Obvious information may be necessary for reasoning.

We may not know in advance which statements to deduce (P or \( \neg P \)).
Representing Simple Facts in Logic

- Formalizing sentences:

  - Some politician is crooked

    \[ \exists x \ (p(x) \land q(x)) \quad \text{or} \quad \exists x \ (p(x) \rightarrow q(x)) \]

    \[ \exists x \ (p(x) \rightarrow q(x)) \equiv \forall x \ p(x) \rightarrow \exists x \ q(x) \]

    “if everyone is a politician, then someone is crooked”
Representing Simple Facts in Logic

- Some politician is crooked: $\exists x \ (p(x) \land q(x))$
- No politician is crooked: $\forall x \ (p(x) \rightarrow \neg q(x))$
- All politicians are crooked: $\forall x \ (p(x) \rightarrow q(x))$
- Not all politicians are crooked: $\exists x \ (p(x) \land \neg q(x))$
- Every politician is crooked: $\forall x \ (p(x) \rightarrow q(x))$
- There is an honest politician: $\exists x \ (p(x) \land \neg q(x))$
- No politician is honest: $\forall x \ (p(x) \rightarrow q(x))$
- All politicians are honest: $\forall x \ (p(x) \rightarrow \neg q(x))$
Representing Simple Facts in Logic

- Formalizing sentences:
  - Each formalization satisfies one of the following two properties:
    - The universal quantifier $\forall x$ quantifies a conditional
    - The existential quantifier $\exists x$ quantifies a conjunction
Representing Instance & Isa Relationships

\[\text{Pompeian(Marcus)}\]
\[\forall x: \text{Pompeian}(x) \rightarrow \text{Roman}(x)\]

\[\text{instance(Marcus, Pompeian)}\]
\[\forall x: \text{instance}(x, \text{Pompeian}) \rightarrow \text{instance}(x, \text{Roman})\]

\[\text{instance(Marcus, Pompeian)}\]
\[\text{isa(Pompeian, Roman)}\]
\[\forall x: \forall y: \forall z: \text{instance}(x, y) \land \text{isa}(y, z) \rightarrow \text{instance}(x, z)\]
Example

“Paulus was a Pompeian. Paulus neither hate Caesar nor are loyal to him”

$\text{Pompeian(Paulus)} \land \neg[\text{loyalto(Paulus, Caesar)} \lor \text{hate(Paulus, Caesar)}]$

$\Rightarrow$ make the knowledge base inconsistent
Make an Exception

Solution:

- Modify the original assertion to which an exception is being made.

\[ \forall X: \text{Roman}(X) \land \neg \text{eq}(X, \text{Paulus}) \rightarrow \text{loyalto}(X, \text{Caesar}) \lor \text{hate}(X, \text{Caesar}) \]
Computable Functions and Predicates

- **Computable predicates**
  - greater-than(1, 0)
  - greater-than(2, 1)
  - greater-than(3, 2)
  - ....
  - less-than(0, 1)
  - less-than(1, 2)
  - less-than(2, 3)
  - ....

- **Computable functions**
  - greater-than(2 + 3, 1)
1. Marcus was a Pompeian
2. All Pompeians died when the volcano erupted in 79 A.D.

Question: Is Marcus alive now?

Pompeian(Marcus)
\[ \forall x: \text{erupted(volcano, 79)} \land \text{Pompeian(x)} \rightarrow \text{died(x, 79)} \text{ OR } \text{erupted(volcano, 79)} \land (\forall x: \text{Pompeian(x)} \rightarrow \text{died(x, 79)})? \]
\[ \forall x: \forall t_1: \forall t_2: \text{died(x, t_1)} \land \text{greater-than(t_2, t_1)} \rightarrow \neg \text{alive(x, t2)} \]
\[ \neg \text{alive(Marcus, 2009)} \]
Resolution

KB |= α

(α is a logical consequence of KB)

How to prove it automatically?
Resolution


- The basic ideas

  \[ \text{KB} \models \alpha \iff \text{KB} \land \neg \alpha \models \text{false} \]

  Refutation proof procedure

- Resolution inference rule

  \[
  \frac{(\alpha \lor P) \land (\gamma \lor \neg P)}{(\alpha \lor \gamma)} \quad \text{premise}
  \]

  conclusion
Resolution in Propositional Logic

1. Convert all the propositions of KB to clause form (S).
2. Negate $\alpha$ and convert it to clause form. Add it to S.
3. Repeat until either a contradiction is found or no progress can be made.
   a. Select two clauses $(\alpha \lor P)$ and $(\gamma \lor \neg P)$.
   b. Add the resolvent $(\alpha \lor \gamma)$ to S.

Clause form = Conjunctive normal form – CNF
Resolution in Propositional Logic

Example

\[ KB = \{ P, (P \land Q) \rightarrow R, (S \lor T) \rightarrow Q, T \} \]

\[ \alpha = R \]

\[ KB \models \alpha ? \]
Resolution in Predicate Logic

Example

\[ KB = \{ P(a), \forall x: (P(x) \land Q(x)) \rightarrow R(x), \forall y: (S(y) \lor T(y)) \rightarrow Q(y), T(a) \} \]

\[ \alpha = R(a) \]

\[ KB \models \alpha ? \]
Resolution in Predicate Logic

- Unification

\[
UNIFY(p, q) = \text{unifier } \theta \text{ where } \text{SUBST}(\theta, p) = \text{SUBST}(\theta, q)
\]

\[
\forall x: \text{knows}(John, x) \to \text{hates}(John, x)
\]

knows(John, Jane)

\[
\forall y: \text{knows}(y, \text{Leonid})
\]

\[
\forall y: \text{knows}(y, \text{mother}(y))
\]

\[
\forall x: \text{knows}(x, \text{Elizabeth})
\]
Resolution in Predicate Logic

- **Unification**

  UNIFY(knows(John, x), knows(John, Jane)) = \{Jane/x\}

  UNIFY(knows(John, x), knows(y, Leonid)) = \{Leonid/x, John/y\}

  UNIFY(knows(John, x), knows(y, mother(y))) = \{John/y, mother(John)/x\}

  UNIFY(knows(John, x), knows(x, Elizabeth)) = \text{FAIL}

  ⇒ Standardizing apart:

  UNIFY(knows(John, x), knows(y, Elizabeth)) = \{John/y, Elizabeth/x\}
Resolution in Predicate Logic

- **Unification: Most general unifier**

  \[ \text{UNIFY}(\text{knows}(\text{John}, x), \text{knows}(y, z)) = \{\text{John}/y, \text{John}/x, \text{John}/z\} = \{\text{John}/y, \text{Jane}/x, \text{Jane}/z\} = \{\text{John}/y, v/x, v/z\} = \{\text{John}/y, z/x, \text{Jane}/v\} = \{\text{John}/y, z/x\} \]

- **Unification: Occur check**

  \[ \text{UNIFY}(\text{knows}(x, x), \text{knows}(y, \text{mother}(y))) = \text{FAIL} \]
Conversion to Clause Form

1. Eliminate $\rightarrow$.
   
   $P \rightarrow Q \equiv \neg P \lor Q$

2. Reduce the scope of each $\neg$ to a single term.
   
   $\neg(P \lor Q) \equiv \neg P \land \neg Q$
   
   $\neg(P \land Q) \equiv \neg P \lor \neg Q$
   
   $\neg\forall x: P \equiv \exists x: \neg P$
   
   $\neg\exists x: p \equiv \forall x: \neg P$
   
   $\neg\neg P \equiv P$

3. Standardize variables so that each quantifier binds a unique variable.

   $(\forall x: P(x)) \lor (\exists x: Q(x)) \equiv (\forall x_1: P(x_1)) \lor (\exists x_2: Q(x_2))$
4. Move all quantifiers to the left without changing their relative order.

\[ \forall x: (P(x) \lor (\exists y: Q(y))) \equiv \forall x: \exists y: (P(x) \lor (Q(y))) \]

\[ (\forall x: P(x)) \lor (\exists y: Q(y)): \text{Don’t move} \]

5. Eliminate \(\exists\) (Skolemization).

\[ \exists x: P(x) \equiv P(c) \quad \text{Skolem constant} \]

\[ \forall x: \exists y: P(x, y) \equiv \forall x: P(x, f(x)) \quad \text{Skolem function} \]

6. Drop \(\forall\).

\[ \forall x: P(x) \equiv P(x) \]
7. Convert the formula into a conjunction of disjunctions.

\[(P \land Q) \lor R \equiv (P \lor R) \land (Q \lor R)\]

8. Create a separate clause corresponding to each conjunct.

\[ (P \lor R) \land (Q \lor R) \Rightarrow P \lor R, \ Q \lor R \]

9. Standardize apart the variables in the set of obtained clauses.
Conversion to Clause Form

1. Eliminate \( \rightarrow \).
2. Reduce the scope of each \( \neg \) to a single term.
3. Standardize variables so that each quantifier binds a unique variable.
4. Move all quantifiers to the left without changing their relative order.
5. Eliminate \( \exists \) (Skolemization).
6. Drop \( \forall \).
7. Convert the formula into a conjunction of disjunctions.
8. Create a separate clause corresponding to each conjunction.
9. Standardize apart the variables in the set of obtained clauses.
Resolution in Predicate Logic

1. Convert all the propositions of KB to clause form (S).

2. Negate $\alpha$ and convert it to clause form. Add it to S.

3. Repeat until a contradiction is found:
   a. Select two clauses
      $(\alpha \lor p(t_1, t_2, \ldots, t_n))$ and $(\gamma \lor \neg p(t'_1, t'_2, \ldots, t'_n))$.
   b. $\theta = \text{mgu}(p(t_1, t_2, \ldots, t_n), p(t'_1, t'_2, \ldots, t'_n))$
   c. Add the resolvent $\theta(\alpha \lor \gamma)$ to S.
Resolution in Predicate Logic

Example

\( \text{KB} = \{ \ P(a), \ \forall x: (P(x) \land Q(x)) \rightarrow R(x), \ \\
\forall y: (S(y) \lor T(y)) \rightarrow Q(y), \ T(a) \ \} \)

\( \alpha = R(a) \)

\( \text{KB} \models \alpha \ ? \)
Resolution in Predicate Logic

1. Marcus was a man.
2. Marcus was a Pompeian.
3. All Pompeians were Romans.
4. Caesar was a ruler.
5. All Romans were either loyal to Caesar or hated him.
6. Every one is loyal to someone.
7. People only try to assassinate rulers they are not loyal to.
8. Marcus tried to assassinate Caesar.

Prove: Marcus hated Caesar.
Resolution in Predicate Logic

1. man(Marcus).
2. Pompeian(Marcus).
3. \( \forall x: \text{Pompeian}(x) \rightarrow \text{Roman}(x). \)
4. ruler(Caesar).
5. \( \forall x: \text{Roman}(x) \rightarrow \text{loyalto}(x, \text{Caesar}) \lor \text{hate}(x, \text{Caesar}). \)
6. \( \forall x: \exists y: \text{loyalto}(x, y). \)
7. \( \forall x: \forall y: \text{person}(x) \land \text{ruler}(y) \land \text{tryass}(x, y) \rightarrow \neg \text{loyalto}(x, y). \)
8. tryass(Marcus, Caesar).
9. \( \forall x: \text{man}(x) \rightarrow \text{person}(x). \)

Prove: \( \text{hate}(\text{Marcus}, \text{Caesar}). \)
Resolution in Predicate Logic

1. Marcus was a Pompeian
2. All Pompeians died when the volcano erupted in 79 A.D
Prove: Marcus is not alive now.

1. Pompeian(Marcus)
2. erupted(volcano, 79) \land (\forall x: \text{Pompeian}(x) \rightarrow \text{died}(x, 79))
3. \forall x: \forall t_1: \forall t_2: \text{died}(x, t_1) \land \text{greater-than}(t_2, t_1) 
   \rightarrow \neg \text{alive}(x, t_2)

Prove: \neg \text{alive}(Marcus, 2009)
<table>
<thead>
<tr>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. When did Marcus die?</td>
</tr>
<tr>
<td>2. Whom did Marcus hate?</td>
</tr>
<tr>
<td>3. Who tried to assassinate a ruler?</td>
</tr>
<tr>
<td>4. What happen in 79 A.D.?</td>
</tr>
<tr>
<td>5. Did Marcus hate everyone?</td>
</tr>
</tbody>
</table>

- Answering = finding a known statement that matches the terms given in the question
Question Answering

When did Marcus die?

\[ \exists t: \text{died}(\text{Marcus}, t) \]

Resolution:

\[ \neg \exists t: \text{died}(\text{Marcus}, t) = \neg \text{died}(\text{Marcus}, t) \]

\[ \neg \text{Pompeian}(x) \lor \text{died}(x, 79) \]

\[ \neg \text{died}(\text{Marcus}, t) \]

\[ \neg \text{Pompeian}(\text{Marcus}) \]

\[ \neg \text{Pompeian}(\text{Marcus}) \]

false
Question Answering

- Eliminate the nil clause

\[ \neg \text{Pompeian}(x) \lor \text{died}(x, 79) \rightarrow \neg \text{died}(\text{Marcus}, t) \lor \text{died}(\text{Marcus}, t) \]

\[ \text{Pompeian}(\text{Marcus}) \rightarrow \neg \text{Pompeian}(\text{Marcus}) \lor \text{died}(\text{Marcus}, 79) \]

\[ \text{Died}(\text{Marcus}, 79) \]
Resolution

- **Soundness**
  - An inference algorithm $R$ that derives only entailed sentences is called sound (or truth-preserving).
  
  $\text{if } KB \text{ derives } \alpha \text{ using } R, \text{ then } KB \models \alpha$

- **Completeness**
  - An inference algorithm is complete if it can derive any sentence that is entailed.

  $\text{if } KB \models \alpha, \text{ then } KB \text{ derives } \alpha \text{ using } R$

- Resolution algorithm is sound and complete
Resolution

In general:

- **Soundness**: any returned answer is a correct answer.
- **Completeness**: all correct answers are returned.
Natural Deduction

- Problems with resolution
  - The heuristic information contained in the original statements can be lost in the transformation
    \[ \forall x: \text{judge}(x) \land \neg \text{crooked}(x) \rightarrow \text{educated}(x) \]
    \[ \neg \text{judge}(x) \lor \neg \text{crooked}(x) \lor \text{educated}(x) \]
  - People do not think in resolution

- Natural deduction: A way of doing machine theorem proving that corresponds more closely to processes used in human theorem proving
Exercise: 1 – 13 (Textbook 1).

Formalize each of the following English sentences, where the domain of discourse is the set of all people.

– Every committee member is rich and famous.
– Some committee members are old.
– All college graduates are smart.
– No college graduate is dumb.
– Not all college graduates are smart.
Homework

- Convert into Clause Form

1. \( \forall x \ A(x) \lor \exists x \ B(x) \rightarrow \forall x \ C(x) \land \exists x \ D(x) \)
2. \( \forall x \ (p(x) \lor q(x)) \rightarrow \forall x \ p(x) \lor \forall x \ q(x) \)
3. \( \exists x \ p(x) \land \exists x \ q(x) \rightarrow \exists x (p(x) \land q(x)) \)
4. \( \forall x \ \exists y \ p(x, y) \rightarrow \exists y \ \forall x \ p(x, y) \)
5. \( \forall x \ \forall y \ (p(x, f(x)) \rightarrow p(x, y)) \)
Homework

- Convert into Clause Form

  - All Romans who know Marcus either hate Caesar or think that anyone who hates everyone is crazy.

  \[ \forall x: [\text{Roman}(x) \land \text{know}(x, \text{Marcus})] \rightarrow [\text{hate}(x, \text{Caesar}) \lor (\forall y: \forall z: \text{hate}(y, z) \rightarrow \text{thinkcrazy}(x, y))] \]

  - Everyone who loves all animals is loved by someone

  \[ \forall x: [\forall y: \text{animal}(y) \rightarrow \text{loves}(x, y)] \rightarrow [\exists y: \text{loves}(y, x)] \]
Homework

Transform each informal argument into a formalized wff. Then give a formal proof of the wff.

- Every dog either likes people or hates cats. Rover is a dog. Rover loves cats. Therefore some dog likes people.

- Every committee member is rich and famous. Some committee members are old. Therefore some committee members are old and famous.

- Every rational number is a real number. There is a rational number. Therefore there is a real number.
Homework

– The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American. Therefore West is a criminal.

– Every one who loves all animals is loved by someone. Anyone who kills an animal is loved by no one. Jack loves all animals. Either Jack or Curiosity killed the cat, who is named Tuna. Therefore Curiosity killed the cat.