인공지능
8차시: Inference in First-Order Logic

서울대학교 컴퓨터공학부
담당 교수: 장병탁

Seoul National University
Byoung-Tak Zhang
Introduction

- In the previous lecture, we learned how the first-order logic (FOL) can model the world as objects and their relations. Here we study how we can answer questions (or make inferences) about the world posed in first-order logic.

- There are four major ways to make inferences in FOL.
  - Propositionalization
  - Forward chaining
  - Backward chaining
  - Resolution inference
Motivating Example: Colonel West Problem

Colonel West Problem

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.

Prove that Colonel West is a criminal.

First-order logic description of the problem

\[
\begin{align*}
\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) & \Rightarrow \text{Criminal}(x) \\
\text{Owns}(\text{Nono, } M1) \\
\text{Missile}(M1) \\
\text{Missile}(x) \land \text{Owns}(\text{Nono, } x) & \Rightarrow \text{Sells}(\text{West, } x, \text{ Nono}) \\
\text{Missile}(x) & \Rightarrow \text{Weapon}(x) \\
\text{Enemy}(x, \text{ America}) & \Rightarrow \text{Hostile}(x) \\
\text{American}(\text{West}) \\
\text{Enemy}(\text{Nono, America})
\end{align*}
\]
Motivating Example: Proof Tree

Proving if Colonel West is a criminal

- Criminal(West)
  - Weapon(M1)
    - American(West)
    - Missile(M1)
      - Owns(Nono,M1)
  - Sells(West,M1,Nono)
  - Hostile(Nono)
    - Enemy(Nono,America)
8.1 Propositional vs. First-Order Inference ............... 6
8.2 Unification and Lifting .................................. 10
8.3 Forward Chaining ........................................ 14
8.4 Backward Chaining ....................................... 20
8.5 Resolution .................................................. 24
Summary ......................................................... 28
Homework ....................................................... 29
8.1 Propositional vs. First-Order Inference (1/4)

- First-order inferences can be done by converting the first-order KB to propositional logic and using propositional inference, which we already know how to do (Lecture 6).

Inference rules for quantifiers

1. Universal instantiation (UI)

   We can infer any sentence $\alpha$ obtained by substituting a ground term $g$ for the variable $v$:

   $\forall v \alpha$  
   $\text{Subst}([v/g], \alpha)$

   $\forall x \text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x)$

   yields

   $\text{King}(\text{John}) \land \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John})$

   $\text{King}(\text{Richard}) \land \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard})$

   $\text{King}(\text{Father}(\text{John})) \land \text{Greedy}(\text{Father}(\text{John})) \Rightarrow \text{Evil}(\text{Father}(\text{John}))$
8.1 Propositional vs. First-Order Inference (2/4)

2. Existential instantiation (EI)

- The variable $v$ in the sentence $\alpha$ is replaced by a single new constant symbol $k$: $\exists v \alpha \quad \text{Subst}\{\{v/k\}, \alpha\}$

- From the sentence $\exists x \ Crown(x) \land OnHead(x, \text{John})$
  
  we can infer the sentence $\ Crown(C_1) \land OnHead(C_1, \text{John})$
  
  provided $C_1$ is a new constant symbol, called a Skolem constant.

- Inferentially equivalent (but not logically equivalent) in the sense that it is satisfiable exactly when the original KB is satisfiable.
8.1 Propositional vs. First-Order Inference (3/4)

Reduction to Propositional Inference

» Suppose the KB contains just the following

\[ \forall x \ \text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x) \]

\[ \text{King}(\text{John}) \]

\[ \text{Greedy}(\text{John}) \]

\[ \text{Brother(Richard, John)} \]

» Instantiating the universal sentence in all possible ways, \( \{x/\text{John}\} \) and \( \{x/\text{Richard}\} \), we have

\[ \text{King}(\text{John}) \land \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John}) \]

\[ \text{King}(\text{Richard}) \land \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard}) \]

» The new KB is propositionalized:

\[ \text{King}(\text{John}), \ \text{Greedy}(\text{John}), \ \text{Evil}(\text{John}), \ \text{King}(\text{Richard}), \ etc. \]
Technique of Propositionalization

- First-order inference via propositionalization is complete—that is, any entailed sentence can be proved.

- What happens when the sentence is not entailed? We cannot tell.

- The question of entailment for first-order logic is semidecidable—that is, algorithms exist that say yes to every entailed sentence, but not algorithm exists that also says no to every non-entailed sentence.

- Cf. The halting problem for Turing machines.
Motivating Example

\[ \forall x \text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x) \]

\[ \text{King}(\text{John}) \]

\[ \forall y \text{Greedy}(y) \]

- Propositionalization approach may generate

\[ \text{King}(\text{Richard}) \land \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard}) \]

which does not match the KB and, thus, useless for proving \( \text{Evil}(\text{John}) \).

- We can get the inference immediately if we can find a substitution \( \theta \) such that \( \text{King}(x) \) and \( \text{Greedy}(x) \) match \( \text{King}(\text{John}) \) and \( \text{Greedy}(\text{John}) \)

- \( \theta = \{ x/\text{John}, y/\text{John} \} \) works
8.2 Unification and Lifting (2/4)

### Unification

- \( \text{Unify}(p, q) = \theta \) where \( p\theta = q\theta \) (\( \theta \): most general unifier (MGU))

- Lifted inference rules require finding substitutions that make different logical expressions look identical. This process is called unification.

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( \theta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{Knows}(\text{John}, x) )</td>
<td>( \text{Knows}(\text{John}, \text{Jane}) )</td>
<td>( { x / \text{Jane} } )</td>
</tr>
<tr>
<td>( \text{Knows}(\text{John}, x) )</td>
<td>( \text{Knows}(y, \text{Bill}) )</td>
<td>( { x / \text{OJ}, y / \text{Jane} } )</td>
</tr>
<tr>
<td>( \text{Knows}(\text{John}, x) )</td>
<td>( \text{Knows}(y, \text{Mother}(y)) )</td>
<td>( { y / \text{Jane}, x / \text{Mother}(\text{John}) } )</td>
</tr>
<tr>
<td>( \text{Knows}(\text{John}, x) )</td>
<td>( \text{Knows}(x, \text{Elizabeth}) )</td>
<td>fail</td>
</tr>
</tbody>
</table>
Generalized Modus Ponens (GMP)

\[ p_1', p_2', \ldots, p_n', (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q) \]
\[ \text{Subst}(\theta, q) \]

where \( \text{Subst}(\theta, p_i') = \text{Subst}(\theta, p_i) \) or \( p_i' \theta = p_i \theta \) for \( \forall i \).

- \( p_1' \) is \( \text{King}(\text{John}) \)
- \( p_2' \) is \( \text{Greedy}(y) \)
- \( \theta \) is \{x /\text{John}, y /\text{John}\}
- \( q \) is \( \text{Evil}(x) \)
- \( \text{Subst}(\theta, q) = q\theta \) is \( \text{Evil}(\text{John}) \)

*GMP is a lifted version of MP. It raises MP from ground (variable-free) propositional logic to first-order logic.*
8.2 Unification and Lifting (4/4)

Soundness of GMP

\[ p \models p\theta \text{ by Ul} \]

\[ p^', \ldots, p^n \models p'^1\theta, \ldots, p'^n\theta \]

\[ p_1 \land \ldots \land p_n \Rightarrow q \models p_1\theta \land \ldots \land p_n\theta \Rightarrow q\theta \]

Now \( \theta \) in GMP is defined as \( p'^i\theta = p_i\theta \) for \( \forall i \).

Thus, we have

\[ p'^1, \ldots, p'^n, (p_1 \land \ldots \land p_n \Rightarrow q) \models q\theta \]

provided that \( p'^i\theta = p_i\theta \) for \( \forall i \)
8.3 Forward Chaining (1/6)

- Inference methods for direct manipulation of FOL (unlike propositionalization)

**Example: Colonel West Problem**

- 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.
- Prove that Colonel West is a criminal.

\[
\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)
\]

\[
\text{Owns}(\text{Nono}, \text{M1})
\]

\[
\text{Missile}(\text{M1})
\]

\[
\text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})
\]

\[
\text{Missile}(x) \Rightarrow \text{Weapon}(x)
\]

\[
\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)
\]

\[
\text{American}(\text{West})
\]

\[
\text{Enemy}(\text{Nono}, \text{America})
\]

First-order definite clauses

Datalog: First-order definite clauses with no function symbols
<table>
<thead>
<tr>
<th>American(West)</th>
<th>Missile(M1)</th>
<th>Owns(Nono,M1)</th>
<th>Enemy(Nono,America)</th>
</tr>
</thead>
</table>

8.3 Forward Chaining (2/6)
8.3 Forward Chaining (3/6)

- **Weapon(M1)**
- **Sells(West,M1,Nono)**
- **Hostile(Nono)**
- **American(West)**
- **Missile(M1)**
- **Owns(Nono,M1)**
- **Enemy(Nono,America)**
8.3 Forward Chaining (4/6)

```
Criminal(West)
  \--- Weapon(M1)
  |    \--- American(West)
  |         \--- Missile(M1)
  \--- Sells(West,M1,Nono)
      \--- Owns(Nono,M1)
  \--- Hostile(Nono)
      \--- Enemy(Nono,America)
```

사진 출처 #4
function FOL-FC-ASK(KB, α) returns a substitution or false

inputs: KB, the knowledge base, a set of first-order definite clauses
        α, the query, an atomic sentence

local variables: new, the new sentences inferred on each iteration

repeat until new is empty
    new ← {}
    for each rule in KB do
        (p₁ ∧ ... ∧ pₙ → q) ← STANDARDIZE-VARIABLES(rule)
        for each θ such that SUBST(θ, p₁ ∧ ... ∧ pₙ) = SUBST(θ, p₁' ∧ ... ∧ pₙ')
            for some p₁', ..., pₙ' in KB
                q' ← SUBST(θ, q)
                if q' does not unify with some sentence already in KB or new then
                    add q' to new
                    φ ← UNIFY(q', α)
                    if φ is not fail then return φ
                add new to KB
    return false
Properties of forward chaining

- **Sound and complete for first-order definite clauses**
  - proof similar to propositional proof

- **Datalog** = first-order definite clauses + no functions (e.g., crime KB)

- FC terminates for Datalog in polynomial number of iterations

- May not terminate in general if $\alpha$ is not entailed

- This is unavoidable: entailment with definite clauses is **semidecidable**

- Forward chaining is widely used in **deductive databases**
8.4 Backward Chaining (1/4)

American(x)

Weapon(y)

Sells(x,y,z)

Hostile(Nono)

Criminal(West) \{x/West, y/M1, z/Nono\}

Missile(y)

{y/M1}

Missile(M1)

Owns(Nono,M1)

Enemy(Nono,America)

{z/Nono}

{z/Nono}
function FOL-BC-Ask(KB, goals, θ) returns a set of substitutions

inputs: KB, a knowledge base
        goals, a list of conjuncts forming a query (θ already applied)
        θ, the current substitution, initially the empty substitution { }

local variables: answers, a set of substitutions, initially empty

if goals is empty then return {θ}
q' ← SUBST(θ, FIRST(goals))
for each sentence r in KB
    where STANDARDIZE-Apart(r) = ( p_1 ∧ ... ∧ p_n ⇒ q )
    and θ' ← UNIFY(q, q') succeeds
    new_goals ← [ p_1, ..., p_n | REST(goals) ]
    answers ← FOL-BC-Ask(KB, new_goals, COMPOSE(θ', θ)) ∪ answers
return answers
Properties of backward chaining

- **Depth-first** recursive proof search: space is linear in size of proof
- **Incomplete** due to infinite loops
  - fix by checking current goal against every goal on stack
- **Inefficient** due to repeated subgoals (both success and failure)
  - fix using caching of previous results (extra space!)
- Widely used (without improvements!) for logic programming
8.4 Backward Chaining (4/4)

Logic programming (PROLOG)

Algorithm = Logic + Control

Prolog program = sets of definite clauses

criminal(X) :- american(X), weapon(Y), sells(X, Y, Z), hostile(Z)

append([], Y, Y).
append([A|X], Y, [A|Z]) :- append(X, Y, Z)

Prolog uses database semantics, i.e. closed-world assumption and negation as failure

Depth-first backward-chaining search
8.5 Resolution (1/4)

Resolution in full first-order

\[ A_1 \lor \ldots \lor A_k, m_1 \lor \ldots \lor m_n \]

\[ \frac{(A_1 \lor \ldots \lor A_{i-1} \lor A_{i+1} \lor \ldots \lor A_k \lor m_1 \lor \ldots \lor m_{j-1} \lor m_{j+1} \lor \ldots \lor m_n)}{(A_1 \lor \ldots \lor A_i - 1 \lor A_i + 1 \lor \ldots \lor A_k \lor m_1 \lor \ldots \lor m_n)} \theta \]

where \( Unify(A_i, \neg m_j) = \theta \).

For example,

\[ \neg Rich(x) \lor Unhappy(x) \]

\[ Rich(Ken) \]

\[ Unhappy(Ken) \]

with \( \theta = \{x/\text{Ken}\} \)

Apply resolution steps to \( CNF (KB \land \neg \alpha) \) complete for FOL
### Conversion to CNF

Everyone who loves all animals is loved by someone:

\[ \forall x \left[ \forall y \ Animal(y) \Rightarrow Loves(x, y) \right] \Rightarrow [\exists y \ Loves(y, x)] \]

1. **Eliminate biconditionals and implications**

\[ \forall x \left[ \neg \forall y \ \neg Animal(y) \lor Loves(x, y) \right] \lor [\exists y \ Loves(y, x)] \]

2. **Move \( \neg \) inwards:** \( \neg \forall x \ p \equiv \exists x \ \neg p, \quad \neg \exists x \ p \equiv \forall x \ \neg p: \)

\[ \forall x \ [\exists y \ \neg \left( \neg Animal(y) \lor \neg Loves(x, y) \right) ] \lor [\exists y \ Loves(y, x)] \]

\[ \forall x \ [\exists y \ \neg \neg Animal(y) \lor \neg Loves(x, y) ] \lor [\exists y \ Loves(y, x)] \]

\[ \forall x \ [\exists y \ Animal(y) \lor \neg Loves(x, y) ] \lor [\exists y \ Loves(y, x)] \]
8.5 Resolution (3/4)

Conversion to CNF (contd.)

3. **Standardize variables**: each quantifier should use a different one

   \[ \forall x \left[ \exists y \ Animal(y) \land \neg \ Loves(x, y) \right] \lor \left[ \exists z \ Loves(z, x) \right] \]

4. **Skolemize**: a more general form of existential instantiation. Each existential variable is replaced by a **Skolem function** of the enclosing universally quantified variables:

   \[ \forall x \left[ \Animal(F(x)) \land \neg \ Loves(x, F(x)) \right] \lor Loves(G(x), x) \]

5. **Drop universal quantifiers**:

   \[ \left[ \Animal(F(x)) \land \neg \ Loves(x, F(x)) \right] \lor Loves(G(x), x) \]

6. **Distribute over**:

   \[ \left[ \Animal(F(x)) \lor Loves(G(x), x) \right] \land \left[ \neg Loves(x, F(x)) \lor Loves(G(x), x) \right] \]

← CNF
Example proofs

\[\neg \text{American}(x) \lor \neg \text{Weapon}(y) \lor \neg \text{Sells}(x,y,z) \lor \neg \text{Hostile}(z) \lor \text{Criminal}(x) \]
\[\neg \text{Criminal}(\text{West})\]
\[\text{American}(\text{West}) \lor \neg \text{Weapon}(y) \lor \neg \text{Sells}(\text{West},y,z) \lor \neg \text{Hostile}(z)\]
\[\neg \text{Weapon}(y) \lor \neg \text{Sells}(\text{West},y,z) \lor \neg \text{Hostile}(z)\]
\[\text{Missile}(x) \lor \text{Weapon}(x)\]
\[\neg \text{Missile}(x) \lor \neg \text{Sells}(\text{West},x,z) \lor \neg \text{Hostile}(z)\]
\[\text{Missile}(M_1) \lor \text{Weapon}(x)\]
\[\neg \text{Missile}(M_1) \lor \neg \text{Sells}(\text{West},y,z) \lor \neg \text{Hostile}(z)\]
\[\neg \text{Missile}(x) \lor \neg \text{Owns}(\text{Nono},x) \lor \text{Sells}(\text{West},x,\text{Nono})\]
\[\text{Sells}(\text{West},M_1,z) \lor \neg \text{Hostile}(z)\]
\[\text{Missile}(M_1) \lor \neg \text{Owns}(\text{Nono},M_1) \lor \neg \text{Hostile}(\text{Nono})\]
\[\neg \text{Missile}(M_1) \lor \neg \text{Owns}(\text{Nono},M_1) \lor \neg \text{Hostile}(\text{Nono})\]
\[\text{Owns}(\text{Nono},M_1) \lor \neg \text{Owns}(\text{Nono},M_1) \lor \neg \text{Hostile}(\text{Nono})\]
\[\neg \text{Enemy}(x,\text{America}) \lor \neg \text{Hostile}(x)\]
\[\text{Hostile}(\text{Nono})\]
\[\text{Enemy}(\text{Nono},\text{America}) \lor \neg \text{Enemy}(\text{Nono},\text{America})\]
Inference by Propositionalization (using universal instantiation and existential instantiation) is slow, unless the domain is small.

The use of unification to identify appropriate substitutions for variables eliminates the instantiation step in FO proofs, making the process more efficient.

A lifted version of Modus Ponens uses unification to provide a powerful inference rule, generalized Modus Ponens. The forward-chaining and backward-chaining algorithms apply this rule to sets of definite clauses.

Forward chaining is used in deductive databases and production systems. Forward chaining is complete for Datalog.

Prolog, unlike first-order logic, uses a closed world with the unique names assumption and negation as failure.

The generalized resolution inference rule provides a complete proof system for first-order logic, using knowledge bases in CNF.
Homework

Exercises

» 9.4
» 9.6
» 9.14
» 9.16
# 1~5 Stuart J. Russell. Berkeley University. Lecture Slides for Artificial Intelligence: A Modern Approach