Inference

Chapter 8 (some of the materials are from Welling at UCI)
How do you do Inference in FOL?

- In Propositional logic:
  - Resolution
  - Forward Chaining
  - Backward chaining

- FOL
  - Reduce to Propositional
  - Adapt Propositional logic algorithms to FOL
Notation: \( \text{Subst}(\{v/g\}, \alpha) \) substitute \( g \) for variable \( v \) in sentence \( \alpha \)

Every instantiation of a universally quantified sentence is entailed by it:

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

for any variable \( v \) and ground term \( g \)

E.g., \( \forall x \, \text{King}(x) \land \text{Greedy}(x) \implies \text{Evil}(x) \) yields:

\[
\begin{align*}
\{x/\text{John}\} & \quad \text{King}(\text{John}) \land \text{Greedy}(\text{John}) \implies \text{Evil}(\text{John}) \\
\{x/\text{Richard}\} & \quad \text{King}(\text{Richard}) \land \text{Greedy}(\text{Richard}) \implies \text{Evil}(\text{Richard})
\end{align*}
\]
Existential instantiation (EI)

- \( \exists x \ Crown(x) \land OnHead(x, John) \) yields.

- We need to add a constant that doesn’t exist anywhere in KB

- E.g., \( C_1 \) is a new constant symbol, called a **Skolem constant**

\[
\text{Crown}(C_1) \land \text{OnHead}(C_1, John)
\]
Suppose the KB contains the following:
\[ \forall x \ King(x) \land Greedy(x) \Rightarrow Evil(x) \]
King(John)
Greedy(John)
Brother(Richard, John)

Instantiating the universal sentence in all possible ways, we have:

(there are only two ground terms: John and Richard)

- King(John) \land Greedy(John) \Rightarrow Evil(John)
- King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard)
- King(John)
- Greedy(John)
- Brother(Richard, John)
Every FOL KB can be propositionalized so as to preserve entailment: A ground sentence is entailed by new KB iff entailed by original KB

Problems?
- function symbols, there are infinitely many ground terms, e.g., \(Father(Father(Father(John))))\), etc
- Propositionalization generates lots of irrelevant sentences, why is this a problem?
How do you do Inference in FOL?

In Propositional logic:
- Resolution
- Forward Chaining
- Backward chaining

FOL
- Reduce to Propositional
- Adapt Propositional logic algorithms to FOL

UNIFICATION
Unification

- Subst(θ, p) = result of substituting θ into sentence p
- Unification algorithm:
  - takes 2 sentences p and q and returns a unifier if one exists
  - \( \text{Unify}(p, q) = \theta \) where \( \text{Subst}(\theta, p) = \text{Subst}(\theta, q) \)

Example:

\[ p = \text{Knows}(\text{John}, x) \]
\[ q = \text{Knows}(\text{John}, \text{Jane}) \]
\[ \text{Unification}(p, q) = \{x/\text{Jane}\} \]
Unification examples

- simple example: query = Knows(John,x), i.e., who does John know?

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>θ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows(John,x)</td>
<td>Knows(John,Jane)</td>
<td>{x/Jane}</td>
</tr>
<tr>
<td>Knows(John,x)</td>
<td>Knows(y,OJ)</td>
<td>{x/OJ,y/John}</td>
</tr>
<tr>
<td>Knows(John,x)</td>
<td>Knows(y,Mother(y))</td>
<td>{y/John,x/Mother(John)}</td>
</tr>
<tr>
<td>Knows(John,x)</td>
<td>Knows(x,OJ)</td>
<td>{fail}</td>
</tr>
</tbody>
</table>

- Last unification fails: only because x can’t take values John and OJ at the same time

- Simple solution: Standardizing apart eliminates overlap of variables, e.g., Knows(z,OJ)
Generalized Modus Ponens (GMP)

\[ p_1', p_2', \ldots, p_n', (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q) \]

Subst(\theta,q)

where we can unify \( p_i' \) and \( p_i \) for all \( i \)

Example:

\( p_1' \) is \( \text{King}(John) \)  \hspace{1cm} \( p_1 \) is \( \text{King}(x) \)

\( p_2' \) is \( \text{Greedy}(y) \)  \hspace{1cm} \( p_2 \) is \( \text{Greedy}(x) \)

\( \theta \) is \( \{x/John, y/John\} \)  \hspace{1cm} \( q \) is \( \text{Evil}(x) \)

\( \text{Subst}(\theta,q) \) is \( \text{Evil}(John) \)
How do you do Inference in FOL?

- In Propositional logic:
  - Resolution
  - Forward Chaining
  - Backward chaining

- FOL
  - Reduce to Propositional
  - Adapt Propositional logic algorithms to FOL
Inference approaches in FOL

- **Forward-chaining**
  - Uses GMP to add new atomic sentences
  - Useful for systems that make inferences as information streams in
  - Requires KB to be in form of first-order definite clauses

- **Backward-chaining**
  - Works backwards from a query to try to construct a proof
  - Can suffer from repeated states and incompleteness
  - Useful for query-driven inference

- **Resolution-based inference (FOL)**
  - Can be used to confirm or refute a sentence p
  - Requires FOL KB to be reduced to CNF
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.
Knowledge Base in FOL

... it is a crime for an American to sell weapons to hostile nations:

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

Nono ... has some missiles, i.e., \( \exists x \ \text{Owns}(\text{Nono},x) \land \text{Missile}(x) \):

\[ \text{Owns}(\text{Nono},M_1) \land \text{Missile}(M_1) \]

... all of its missiles were sold to it by Colonel West

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

Missiles are weapons:

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

An enemy of America counts as "hostile“:

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

West, who is American ...

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

The country Nono, an enemy of America ...

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

\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]
\[ \text{Owns}(\text{Nono},M_1) \text{ and } \text{Missile}(M_1) \]
\[ \text{Missile}(x) \land \text{Owns}(\text{Nono},x) \Rightarrow \text{Sells}(\text{West},x,\text{Nono}) \]
\[ \text{Missile}(x) \Rightarrow \text{Weapon}(x) \]
\[ \text{Enemy}(x,\text{America}) \Rightarrow \text{Hostile}(x) \]
\[ \text{American}(\text{West}) \]
\[ \text{Enemy}(\text{Nono, America}) \]
American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono,M₁) and Missile(M₁)
Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)
Missile(x) ⇒ Weapon(x)
Enemy(x,America) ⇒ Hostile(x)
American(West)
Enemy(Nono,America)
Forward chaining proof

American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono,M₁) and Missile(M₁)
Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)
Missile(x) ⇒ Weapon(x)
Enemy(x,America) ⇒ Hostile(x)
American(West)
Enemy(Nono,America)
Properties of forward chaining

- Sound and complete for first-order definite clauses
- **Datalog** = first-order definite clauses + no functions
- FC terminates for Datalog in finite number of iterations
- May not terminate in general if $\alpha$ is not entailed
Backward chaining example

\[
\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)
\]
\[
\text{Owns}(\text{Nono},M_1) \text{ and } \text{Missile}(M_1)
\]
\[
\text{Missile}(x) \land \text{Owns}(\text{Nono},x) \Rightarrow \text{Sells}(\text{West},x,\text{Nono})
\]
\[
\text{Missile}(x) \Rightarrow \text{Weapon}(x)
\]
\[
\text{Enemy}(x,\text{America}) \Rightarrow \text{Hostile}(x)
\]
\[
\text{American}(\text{West})
\]
\[
\text{Enemy}(\text{Nono},\text{America})
\]
Backward chaining example

\[ \text{Backward chaining example} \]

\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]
\[ \text{Owns}(\text{Nono},M_1) \text{ and } \text{Missile}(M_1) \]
\[ \text{Missile}(x) \land \text{Owns}(\text{Nono},x) \Rightarrow \text{Sells}(\text{West},x,\text{Nono}) \]
\[ \text{Missile}(x) \Rightarrow \text{Weapon}(x) \]
\[ \text{Enemy}(x,\text{America}) \Rightarrow \text{Hostile}(x) \]
\[ \text{American}(\text{West}) \]
\[ \text{Enemy}(\text{Nono},\text{America}) \]
American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono,M₁) and Missile(M₁)
Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)
Missile(x) ⇒ Weapon(x)
Enemy(x,America) ⇒ Hostile(x)
American(West)
Enemy(Nono,America)
Backward chaining example

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

\[
\text{Owns}(\text{Nono}, M_1) \text{ and Missiles}(M_1)
\]

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

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

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

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

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

American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)

Owns(Nono,M₁) and Missile(M₁)

Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)

Missile(x) ⇒ Weapon(x)

Enemy(x,America) ⇒ Hostile(x)

American(West)

Enemy(Nono,America)
Backward chaining example

\[
\begin{align*}
\text{American}(x) & \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \\
\text{Owns}(\text{Nono},M_1) \text{ and } \text{Missile}(M_1) \\
\text{Missile}(x) & \land \text{Owns}(\text{Nono},x) \Rightarrow \text{Sells}(\text{West},x,\text{Nono}) \\
\text{Missile}(x) & \Rightarrow \text{Weapon}(x) \\
\text{Enemy}(x,\text{America}) & \Rightarrow \text{Hostile}(x) \\
\text{American}(\text{West}) \\
\text{Enemy}(\text{Nono},\text{America})
\end{align*}
\]
Backward chaining example

\[
\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)\\
\text{Owns}(\text{Nono},M_1) \text{ and Missile}(M_1)\\
\text{Missile}(x) \land \text{Owns}(\text{Nono},x) \Rightarrow \text{Sells}(\text{West},x,\text{Nono})\\
\text{Missile}(x) \Rightarrow \text{Weapon}(x)\\
\text{Enemy}(x,\text{America}) \Rightarrow \text{Hostile}(x)\\
\text{American}(\text{West})\\
\text{Enemy}(\text{Nono},\text{America})
\]
Properties of backward chaining

- Depth-first recursive proof search: space is linear in size of proof
- Incomplete due to infinite loops
  - \(\Rightarrow\) fix by checking current goal against every goal on stack
- Inefficient due to repeated subgoals (both success and failure)
  - \(\Rightarrow\) fix using caching of previous results (memorization)
Resolution in FOL

Full first-order version:

\[ l_1 \lor \cdots \lor l_k, \quad m_1 \lor \cdots \lor m_n \]

\[
\text{Subst}(\theta, l_1 \lor \cdots \lor l_{i-1} \lor l_{i+1} \lor \cdots \lor l_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n)
\]

where \( \text{Unification}(l_i, \neg m_j) = \theta \).

For example,

\[
\neg \text{Rich}(x) \lor \text{Unhappy}(x), \quad \text{Rich}(\text{Ken})
\]

\[
\text{Unhappy}(\text{Ken})
\]

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

Apply resolution steps to \( \text{CNF}(KB \land \neg \alpha) \); complete for FOL
Converting FOL sentences to CNF

Original sentence:
Everyone who loves all animals is loved by someone:
\[ \forall x \left[ \forall y \text{Animal}(y) \implies \text{Loves}(x,y) \right] \implies \left[ \exists y \text{Loves}(y,x) \right] \]

1. Eliminate biconditionals and implications
\[ \forall x \left[ \neg \forall y \neg \text{Animal}(y) \lor \text{Loves}(x,y) \right] \lor \left[ \exists y \text{Loves}(y,x) \right] \]

2. Move \(\neg\) inwards:
Recall: \(\neg \forall x \, p \equiv \exists x \, \neg p\), \(\neg \exists x \, p \equiv \forall x \, \neg p\)
\[ \forall x \left[ \exists y \neg \left( \neg \text{Animal}(y) \lor \text{Loves}(x,y) \right) \right] \lor \left[ \exists y \text{Loves}(y,x) \right] \]
\[ \forall x \left[ \exists y \neg \text{Animal}(y) \land \neg \text{Loves}(x,y) \right] \lor \left[ \exists y \text{Loves}(y,x) \right] \]
\[ \forall x \left[ \exists y \text{Animal}(y) \land \neg \text{Loves}(x,y) \right] \lor \left[ \exists y \text{Loves}(y,x) \right] \]
3. **Standardize variables:** each quantifier should use a different one

\[ \forall x [\exists y \text{Animal}(y) \land \neg \text{Loves}(x,y)] \lor [\exists z \text{Loves}(z,x)] \]

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

\[ \forall x [\text{Animal}(F(x)) \land \neg \text{Loves}(x,F(x))] \lor \text{Loves}(G(x),x) \]

(reason: animal y could be a different animal for each x.)
5. Drop universal quantifiers:

\[ \text{Animal}(F(x)) \land \neg \text{Loves}(x,F(x))] \lor \text{Loves}(G(x),x) \]

(all remaining variables assumed to be universally quantified)

6. Distribute \( \lor \) over \( \land \):

\[ \text{Animal}(F(x)) \lor \text{Loves}(G(x),x)] \land [\neg \text{Loves}(x,F(x)) \lor \text{Loves}(G(x),x)] \]

\( \uparrow \) Need to include negated query

\( \uparrow \) Then use resolution to attempt to derive the empty clause
which show that the query is entailed by the KB
\( \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \)

\( \text{Owns}(\text{Nono},M_1) \land \text{Missile}(M_1) \)

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

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

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

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

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

Convert to CNF

Q: Criminal(\text{West})?
KB:

Everyone who loves all animals is loved by someone

Anyone who kills animals is loved by no-one

Jack loves all animals

Either Curiosity or Jack killed the cat, who is named Tuna

Query: Did Curiosity kill the cat?

Inference Procedure:

Express sentences in FOL

Convert to CNF form and negated query