Inference in First-order Logic

Vibhav Gogate
The University of Texas at Dallas
Slides borrowed from Rina Dechter and Alex Ihler
Some more notation

• Instantiation: specify values for variables

• Ground term
  – A term without variables

• Substitution
  – Setting a variable equal to something
  – \( \theta = \{x / \text{John}, y / \text{Richard}\} \)
  – Read as “\( x := \text{John}, y:=\text{Richard} \)”

• Write a substitution into sentence \( \alpha \) as
  \( \text{Subst}(\theta, \alpha) \) or just as \( \alpha \theta \)
Universal instantiation (UI)

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

  \[ \forall v \alpha \quad \text{Subst}\{\{v/g\}, \alpha\} \]

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

- E.g., \( \forall x \ King(x) \land Greedy(x) \Rightarrow Evil(x) \) yields:

  \( King(John) \land Greedy(John) \Rightarrow Evil(John) \)

  \( King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard) \)

  \( King(Father(John)) \land Greedy(Father(John)) \Rightarrow Evil(Father(John)) \)

  .

  .

  .
Existential instantiation (EI)

- For any sentence $\alpha$, variable $v$, and constant symbol $k$ that does not appear elsewhere in the knowledge base:

$$
\exists v \alpha \\
\text{Subst}\{\{v/k\}, \alpha\}
$$

- E.g., $\exists x \text{Crown}(x) \land \text{OnHead}(x, \text{John})$ yields:

$$
\text{Crown}(C_1) \land \text{OnHead}(C_1, \text{John})
$$

provided $C_1$ is a new constant symbol, called a Skolem constant
Suppose the KB contains just the following:
∀x King(x) ∧ Greedy(x) ⇒ Evil(x)
King(John)
Greedy(John)
Brother(Richard,John)

• Instantiating the universal sentence in all possible ways, we have:
  King(John) ∧ Greedy(John) ⇒ Evil(John)
  King(Richard) ∧ Greedy(Richard) ⇒ Evil(Richard)
  King(John)
  Greedy(John)
  Brother(Richard,John)

• The new KB is propositionalized: proposition symbols are
  King(John), Greedy(John), Evil(John), King(Richard), etc.
Reduction contd.

• Every FOL KB can be propositionalized so as to preserve entailment

• (A ground sentence is entailed by new KB iff entailed by original KB)

• Idea: propositionalize KB and query, apply resolution, return result

• Problem: with function symbols, there are infinitely many ground terms,
  – e.g., Father(Father(Father(John))))
Theorem: Herbrand (1930). If a sentence $\alpha$ is entailed by an FOL KB, it is entailed by a finite subset of the propositionalized KB

Idea: For $n = 0$ to $\infty$ do
- create a propositional KB by instantiating with depth-$n$ terms
- see if $\alpha$ is entailed by this KB

Problem: works if $\alpha$ is entailed, loops if $\alpha$ is not entailed

Theorem: Turing (1936), Church (1936) Entailment for FOL is semidecidable (algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every nonentailed sentence.)
Problems with propositionalization

- Rather inefficient
- Propositionalization seems to generate lots of irrelevant sentences.
- E.g., from:
  \[ \forall x \text{King}(x) \land \text{Greedy}(x) \implies \text{Evil}(x) \]
  \[ \text{King}(\text{John}) \]
  \[ \forall y \text{Greedy}(y) \]
  \[ \text{Brother}(\text{Richard}, \text{John}) \]
- Given query “evil(x) it seems obvious that Evil(John), but propositionalization produces lots of facts such as Greedy(Richard) that are irrelevant
- With \( p \) \( k \)-ary predicates and \( n \) constants, there are \( p \cdot n^k \) instantiations.
Generalized Modus Ponens (GMP)

\[ p_1', p_2', \ldots, p_n', (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q) \quad \text{where } p_i' \theta = p_i \theta \text{ for all } i \]

\[ q \theta \]

\[ p_1' \text{ is } \textit{King}(\text{John}) \quad p_1 \text{ is } \textit{King}(x) \]

\[ p_2' \text{ is } \textit{Greedy}(y) \quad p_2 \text{ is } \textit{Greedy}(x) \]

\[ \theta \text{ is } \{x/\text{John},y/\text{John}\} \quad q \text{ is } \textit{Evil}(x) \]

\[ q \theta \text{ is } \textit{Evil}(\text{John}) \]

- GMP used with KB of \textit{definite clauses} (one positive literal)
- All variables assumed universally quantified
- \textbf{Lifted version of Modus Ponens}
Unification

- Lifted inference rules require finding substitutions that make different logical expressions look identical
- 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 $\text{Greedy}(y)$

$\theta = \{x/\text{John}, y/\text{John}\}$ works

- $\text{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows(John,x)</td>
<td>Knows(John,Jane)</td>
<td></td>
</tr>
<tr>
<td>Knows(John,x)</td>
<td>Knows(y,OJ)</td>
<td></td>
</tr>
<tr>
<td>Knows(John,x)</td>
<td>Knows(y,Mother(y))</td>
<td></td>
</tr>
<tr>
<td>Knows(John,x)</td>
<td>Knows(x,OJ)</td>
<td></td>
</tr>
</tbody>
</table>

- **Standardizing apart** eliminates overlap of variables, e.g., Knows($z_{17}$,OJ)
Unification

• 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 $\text{Greedy}(y)$

$\theta = \{x/\text{John}, y/\text{John}\}$ works

• $\text{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows(John,x)</td>
<td>Knows(John,Jane)</td>
<td>${x/\text{Jane}}$</td>
</tr>
<tr>
<td>Knows(John,x)</td>
<td>Knows(y,OJ)</td>
<td></td>
</tr>
<tr>
<td>Knows(John,x)</td>
<td>Knows(y,Mother(y))</td>
<td></td>
</tr>
<tr>
<td>Knows(John,x)</td>
<td>Knows(x,OJ)</td>
<td></td>
</tr>
</tbody>
</table>

• **Standardizing apart** eliminates overlap of variables, e.g., $\text{Knows}(z_{17}, \text{OJ})$
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 $\text{Greedy}(y)$

$\theta = \{x/\text{John}, y/\text{John}\}$ works

- **Unify($\alpha, \beta$) = $\theta$ if $\alpha\theta = \beta\theta$**

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\theta$</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>${x/OJ, y/\text{John}}$</td>
</tr>
<tr>
<td>Knows(John,x)</td>
<td>Knows(y,Mother(y))</td>
<td></td>
</tr>
<tr>
<td>Knows(John,x)</td>
<td>Knows(x,OJ)</td>
<td></td>
</tr>
</tbody>
</table>

- **Standardizing apart** eliminates overlap of variables, e.g., Knows($z_{17}$,OJ)
Unification

- 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 \( \text{Greedy}(y) \)

\[ \theta = \{x/\text{John}, y/\text{John}\} \text{ works} \]

- \( \text{Unify}(\alpha, \beta) = \theta \) if \( \alpha\theta = \beta\theta \)

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

- **Standardizing apart** eliminates overlap of variables, e.g., \( \text{Knows}(z_{17}, \text{OJ}) \)
Unification

• 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 \( \text{Greedy}(y) \)

\[ \theta = \{x/\text{John}, y/\text{John}\} \text{ works} \]

• \( \text{Unify}(\alpha, \beta) = \theta \text{ if } \alpha\theta = \beta\theta \)

<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{OJ}) )</td>
<td>( {x/\text{OJ}, y/\text{John}} )</td>
</tr>
<tr>
<td>( \text{Knows}(\text{John}, x) )</td>
<td>( \text{Knows}(y, \text{Mother}(y)) )</td>
<td>( {\text{y/John}, x/\text{Mother(John)}} )</td>
</tr>
<tr>
<td>( \text{Knows}(\text{John}, x) )</td>
<td>( \text{Knows}(x, \text{OJ}) )</td>
<td>( {\text{fail}} )</td>
</tr>
</tbody>
</table>

• **Standardizing apart** eliminates overlap of variables, e.g., \( \text{Knows}(z_{17}, \text{OJ}) \)
Unification

• To unify $\text{Knows}(John,x)$ and $\text{Knows}(y,z)$,

$\theta = \{y/John, x/z \}$ or $\theta = \{y/John, x/John, z/John\}$

• The first unifier is more general than the second.

• There is a single most general unifier (MGU) that is unique up to renaming of variables.

$\text{MGU} = \{y/John, x/z\}$
The unification algorithm

```
function UNIFY(x, y, θ) returns a substitution to make x and y identical
  inputs:  x, a variable, constant, list, or compound
           y, a variable, constant, list, or compound
           θ, the substitution built up so far
  if θ = failure then return failure
  else if x = y then return θ
  else if VARIABLE?(x) then return UNIFY-VAR(x, y, θ)
  else if VARIABLE?(y) then return UNIFY-VAR(y, x, θ)
  else if COMPOUND?(x) and COMPOUND?(y) then
    return UNIFY(ARGS[x], ARGS[y], UNIFY(OP[x], OP[y], θ))
  else if LIST?(x) and LIST?(y) then
    return UNIFY(REST[x], REST[y], UNIFY(FIRST[x], FIRST[y], θ))
  else return failure
```
The unification algorithm

function \textsc{Unify-Var}(var, x, \theta) \textbf{returns} a substitution

inputs: \textup{var}, a variable
\hspace{1em} x, any expression
\hspace{1em} \theta, the substitution built up so far

if \{var/val\} \in \theta \textbf{ then return} \textsc{Unify}(val, x, \theta)
else if \{x/val\} \in \theta \textbf{ then return} \textsc{Unify}(var, val, \theta)
else if \textsc{Occur-Check}\,(var, x) \textbf{ then return} failure
else return add \{var/x\} to \theta
Example knowledge base

- 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 Col. West is a criminal
Example knowledge base contd.

... it is a crime for an American to sell weapons to hostile nations:
\[\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)\]

Nono ... has some missiles, i.e., \(\exists x \ \text{Owns}(\text{Nono},x) \land \text{Missile}(x)\):
\[\text{Owns}(\text{Nono},M_1) \text{ and } \text{Missile}(M_1)\]

... all of its missiles were sold to it by Colonel West
\[\text{Missile}(x) \land \text{Owns}(\text{Nono},x) \Rightarrow \text{Sells}(\text{West},x,\text{Nono})\]

Missiles are weapons:
\[\text{Missile}(x) \Rightarrow \text{Weapon}(x)\]

An enemy of America counts as "hostile“:
\[\text{Enemy}(x,\text{America}) \Rightarrow \text{Hostile}(x)\]

West, who is American ...
\[\text{American}(\text{West})\]

The country Nono, an enemy of America ...
\[\text{Enemy}(\text{Nono},\text{America})\]
Forward chaining algorithm

function FOL-FC-Ask(KB, α) returns a substitution or false

repeat until new is empty

new ← {}

for each sentence r in KB do

(p_1 ∧ ... ∧ p_n ⇒ q) ← STANDARDIZE-APART(r)

for each θ such that (p_1 ∧ ... ∧ p_n)_θ = (p'_1 ∧ ... ∧ p'_n)_θ

for some p'_1, ..., p'_n in KB

q' ← SUBST(θ, q)

if q' is not a renaming of a sentence already in KB or new then do

add q' to new

φ ← UNIFY(q', α)

if φ is not fail then return φ

add new to KB

return false
Forward chaining proof

American(West)  Missile(MI)  Owns(Nono,MI)  Enemy(Nono,America)
Forward chaining proof

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

Missile(x) \Rightarrow Weapon(x)
Forward chaining proof

\[\text{Criminal}(\text{West}) \iff \text{American}(\text{West}) \land \text{Weapon}(y) \land \text{Sells}(\text{West},y,z) \land \text{Hostile}(z) \rightarrow \text{Criminal}(\text{x})\]
Forward chaining proof

*Criminal(West)

*Weapon(M1)  Sells(West,M1,Nono)  Hostile(Nono)  Criminal(West)

*American(West)

*Weapon(M1) and Missile(M1)

*Missile(x)  Owns(Nono,M1)  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

• This is unavoidable: entailment with definite clauses is semidecidable
Backward chaining example

Criminal(West)
Backward chaining example
Backward chaining example

- Criminal(West)
  - American(West)
    - {}
  - Weapon(y)
  - Sells(x,y,z)
  - Hostile(z)

\{x/West\}
Backward chaining example
Backward chaining example

- \text{Criminal(West)}
- \text{American(West) (values: {})}
- \text{Weapon(y) (values: \{ y/M1 \})}
- \text{Sells(x,y,z) (values: \{ x/West, y/M1 \})}
- \text{Hostile(z) (values: {})}
Backward chaining example
Backward chaining example
Backward chaining example
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 for logic programming
Resolution: brief summary

- Full first-order version:
  \[
  \frac{l_1 \lor \cdots \lor l_k, \quad m_1 \lor \cdots \lor m_n}{(l_1 \lor \cdots \lor l_{i-1} \lor l_{i+1} \lor \cdots \lor l_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n)\theta}
  \]
  where \( \text{Unify}(l_i, \neg m_j) = \theta \).

- The two clauses are assumed to be standardized apart so that they share no variables.
- For example,
  \[
  \neg \text{Rich}(x) \lor \text{Unhappy}(x)
  \]
  \[
  \frac{\text{Rich}(Ken)}{\text{Unhappy}(Ken)}
  \]
  with \( \theta = \{x/\text{Ken}\} \)

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

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

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

2. Move \( \neg \) inwards: \( \neg \forall x \ p \equiv \exists x \ \neg p, \ \neg \exists x \ p \equiv \forall x \ \neg p \)
   \[ \forall x \left[ \exists y \neg (\neg \text{Animal}(y) \lor \text{Loves}(x,y)) \right] \lor [\exists y \text{Loves}(y,x)] \]
   \[ \forall x \left[ \exists y \neg \text{Animal}(y) \land \neg \text{Loves}(x,y) \right] \lor [\exists y \text{Loves}(y,x)] \]
   \[ \forall x \left[ [\exists y \text{Animal}(y) \land \neg \text{Loves}(x,y)] \lor [\exists y \text{Loves}(y,x)] \right] \]
Conversion to CNF contd.

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

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

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

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

5. Drop universal quantifiers:

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

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

\[ [Animal(F(x)) \lor Loves(G(x),x)] \land [\neg Loves(x,F(x)) \lor Loves(G(x),x)] \]
... it is a crime for an American to sell weapons to hostile nations:
\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]

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

Missiles are weapons:
\[ \text{Missile}(x) \Rightarrow \text{Weapon}(x) \]

An enemy of America counts as "hostile“:
\[ \text{Enemy}(x,\text{America}) \Rightarrow \text{Hostile}(x) \]

West, who is American ...
\[ \text{American}(\text{West}) \]

The country Nono, an enemy of America ...
\[ \text{Enemy}(\text{Nono},\text{America}) \]
Resolution proof: definite clauses

$\neg \text{American}(x) \lor \neg \text{Weapon}(y) \lor \neg \text{Sells}(x,y,z) \lor \neg \text{Hostile}(z) \lor \text{Criminal}(x)$

$\neg \text{Criminal}(\text{West})$

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

$\neg \text{American}(\text{West}) \lor \neg \text{Weapon}(y) \lor \neg \text{Sells}(\text{West},y,z) \lor \neg \text{Hostile}(z)$

$\text{Missile}(x) \lor \text{Weapon}(x)$

$\neg \text{Weapon}(y) \lor \neg \text{Sells}(\text{West},y,z) \lor \neg \text{Hostile}(z)$

$\text{Missile}(\text{M1})$

$\neg \text{Missile}(y) \lor \neg \text{Sells}(\text{West},y,z) \lor \neg \text{Hostile}(z)$

$\neg \text{Missile}(x) \lor \neg \text{Owns}(\text{Nono},x) \lor \text{Sells}(\text{West},x,\text{Nono})$

$\neg \text{Sells}(\text{West},\text{M1},z) \lor \neg \text{Hostile}(z)$

$\text{Missile}(\text{M1})$

$\neg \text{Missile}(\text{M1}) \lor \neg \text{Owns}(\text{Nono},\text{M1}) \lor \neg \text{Hostile}(\text{Nono})$

$\text{Owns}(\text{Nono},\text{M1})$

$\neg \text{Owns}(\text{Nono},\text{M1}) \lor \neg \text{Hostile}(\text{Nono})$

$\neg \text{Enemy}(x,\text{America}) \lor \text{Hostile}(x)$

$\neg \text{Hostile}(\text{Nono})$

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

$\neg \text{Enemy}(\text{Nono},\text{America})$
Converting to clause form

\[ \forall x, y \ P(x) \land P(y) \land I(x,27) \land I(y,28) \rightarrow S(x, y) \]

Prove I(A,27)
Example: Resolution Refutation Prove \( I(A,27) \)

\[
\neg I(A,27) \quad \text{(negation of wff to be proved)}
\]

\[
I(A,27) \lor I(A,28)
\]

\[
\neg P(x) \lor \neg P(y) \lor \neg I(x,27) \lor \neg I(y,28) \lor S(x,y)
\]

\[
\neg P(x) \lor \neg P(A) \lor \neg I(x,27) \lor S(x,A)
\]

\[
\{A/y\}
\]

\[
P(A)
\]

\[
\neg P(x) \lor \neg I(x,27) \lor S(x,A)
\]

\[
I(B,27)
\]

\[
\neg I(B,27) \lor S(B,A)
\]

\[
\{B/x\}
\]

\[
\neg S(B,A)
\]

\[
S(B,A)
\]

\[
\text{Nil}
\]
Example: Answer Extraction

\[ \neg I(A, u) \lor \text{Ans}(u) \]

(negation of wff to be proved with answer literal)

\[ I(A, 27) \lor I(A, 28) \]

\[ \neg P(x) \lor \neg P(y) \lor \neg I(x, 27) \lor \neg I(y, 28) \lor S(x, y) \]

\[ I(A, 28) \lor \text{Ans}(27) \]

\[ \{27/u\} \]

\[ P(A) \]

\[ \neg P(x) \lor \neg P(A) \lor \neg I(x, 27) \lor S(x, A) \lor \text{Ans}(27) \]

\[ \{A/y\} \]

\[ P(B) \]

\[ \neg P(x) \lor \neg I(x, 27) \lor S(x, A) \lor \text{Ans}(27) \]

\[ I(B, 27) \]

\[ \neg I(B, 27) \lor S(B, A) \lor \text{Ans}(27) \]

\[ \{B/x\} \]

\[ \neg S(B, A) \]

\[ S(B, A) \lor \text{Ans}(27) \]

\[ \text{Ans}(27) \]