Inference in FOL

• All rules of inference for propositional logic apply to first-order logic
• We just need to reduce FOL sentences to PL sentences by instantiating variables and removing quantifiers
Reduction of FOL to PL

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

• How can we reduce this to PL?

• Let’s instantiate the universal sentence in all possible ways:
  King(John) \land \text{Greedy}(John) \Rightarrow \text{Evil}(John)
  King(\text{Richard}) \land \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard})
  King(\text{John}) \land \text{Greedy}(\text{John}) \land \text{Brother}(\text{Richard,John})

• The KB is *propositionalized*
  – Proposition symbols are King(John), Greedy(John), Evil(John), King(\text{Richard}), etc.
Reduction of FOL to PL

- What about existential quantification, e.g.,
  \[ \exists x \text{ Crown}(x) \land \text{OnHead}(x,\text{John}) \]?
- Let’s instantiate the sentence with a new constant that doesn’t appear anywhere in the KB:
  \[ \text{Crown}(C_1) \land \text{OnHead}(C_1,\text{John}) \]
Propositionalization

• Every FOL KB can be *propositionalized* so as to preserve entailment
  – A ground sentence is entailed by the new KB iff it is entailed by the original KB

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

• **Problem:** with function symbols, there are infinitely many ground terms
  – For example, Father(X) yields Father(John), Father(Father(John)), Father(Father(Father(John))), etc.
Propositionalization

• **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 Infinity 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
Inference in FOL

• “All men are mortal. Socrates is a man; therefore, Socrates is mortal.”

• Can we prove this without full propositionalization as an intermediate step?
Substitution

- **Substitution** of variables by *ground terms*:

  \[ \text{SUBST}\{v/g\}, P\]

  - Result of \(\text{SUBST}\{x/Harry, y/Sally\}, \text{Loves}(x,y)\):
    \[\text{Loves}(Harry, Sally)\]

  - Result of \(\text{SUBST}\{x/John\}, \text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x)\):
    \[\text{King}(John) \land \text{Greedy}(John) \Rightarrow \text{Evil}(John)\]
Universal instantiation (UI)

• A universally quantified sentence entails every instantiation of it:
  \[ \forall v \, P(v) \]
  \[ \text{SUBST} \left( \{v/g\}, \, P(v) \right) \]

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

• E.g., \( \forall x \, \text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x) \) yields:
  \( \text{King}(\text{John}) \land \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John}) \)
  \( \text{King}(\text{Richard}) \land \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard}) \)
  \( \text{King}(\text{Father}(\text{John})) \land \text{Greedy}(\text{Father}(\text{John})) \Rightarrow \text{Evil}(\text{Father}(\text{John})) \)
Existential instantiation (EI)

• An existentially quantified sentence entails the instantiation of that sentence with a new constant:

\[
\exists v \, P(v) \\
\text{SUBST(}\{v/C\}, \, P(v))
\]

for any sentence \( P \), variable \( v \), and constant \( C \) that does not appear elsewhere in the knowledge base

• 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
Generalized Modus Ponens (GMP)
Generalized Modus Ponens (GMP)

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

such that \(\text{SUBST}(\theta, p_i) = \text{SUBST}(\theta, p_i')\) for all i

\[\text{SUBST}(\theta, q)\]

• All variables assumed universally quantified

• Example:

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

King(John) Greedy(John) Brother(Richard, John)

\[p_1 \text{ is King}(x), \quad p_2 \text{ is Greedy}(x), \quad q \text{ is Evil}(x)\]

\[p_1' \text{ is King}(John), \quad p_2' \text{ is Greedy}(y), \quad \theta \text{ is } \{x/John, y/John\}\]

\[\text{SUBST}(\theta, q) \text{ is Evil}(John)\]
Unification

\[ \text{UNIFY}(\alpha, \beta) = \theta \] means that \( \text{SUBST}(\theta, \alpha) = \text{SUBST}(\theta, \beta) \)

<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,Mary)</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,Mary)</td>
<td></td>
</tr>
<tr>
<td>Knows(John,x)</td>
<td>Knows(y,z)</td>
<td></td>
</tr>
</tbody>
</table>

- Standardizing apart eliminates overlap of variables
- Most general unifier
Inference with GMP

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

such that \( \text{SUBST}(\theta, p_i) = \text{SUBST}(\theta, p_i') \) for all \( i \)

\[\text{SUBST}(\theta, q)\]

• **Forward chaining**
  – Like search: keep proving new things and adding them to the KB until we can prove \( q \)

• **Backward chaining**
  – Find \( p_1, \ldots, p_n \) such that knowing them would prove \( q \)
  – Recursively try to prove \( p_1, \ldots, p_n \)
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

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

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

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

American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono,M1) ∧ 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) ∧ 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,M_1) ∧ Missile(M_1)
Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)
Missile(x) ⇒ Weapon(x)
American(West) ⇒ Enemy(x,America) ⇒ Hostile(x)
American(West) ⇒ Enemy(Nono,America)
Backward chaining example

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

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

American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono,M_1) ∧ Missile(M_1)
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

\[
\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \\
\text{Owns}(\text{Nono}, \text{M}_1) \land \text{Missile}(\text{M}_1) \\
\text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \\
\text{Missile}(x) \Rightarrow \text{Weapon}(x) \\
\text{American}(\text{West}) \\
\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x) \\
\text{Enemy}(\text{Nono}, \text{America})
\]
Backward chaining example

\[
\begin{align*}
\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) & \Rightarrow \text{Criminal}(x) \\
\text{Owns}(\text{Nono}, \text{M}_1) \land \text{Missile}(\text{M}_1) & \\
\text{Missile}(x) \land \text{Owns}(\text{Nono}, x) & \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \\
\text{Missile}(x) & \Rightarrow \text{Weapon}(x) \\
\text{Enemy}(x, \text{America}) & \Rightarrow \text{Hostile}(x) \\
\text{American}(\text{West}) & \\
\text{Enemy}(\text{Nono}, \text{America}) &
\end{align*}
\]
Backward chaining example

American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono,M₁) ∧ Missile(M₁)
Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)
Missile(x) ⇒ Weapon(x)
American(West) ⇒ Enemy(x,America) ⇒ Hostile(x)
American(West)
Enemy(Nono,America)
Backward chaining example

American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono,M₁) ∧ 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)
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) = (p₁ ∧ ... ∧ pₙ ⇒ q)
      and θ' ← UNIFY(q, q') succeeds
      new_goals ← [p₁, ..., pₙ|REST(goals)]
      answers ← FOL-BC-Ask(KB, new_goals, COMPOSE(θ', θ)) ∪ answers
  return answers
Resolution: FOL version

\[
p_1 \lor \cdots \lor p_k, \quad q_1 \lor \cdots \lor q_n
\]
such that \( \text{UNIFY}(p_i, \neg q_j) = \theta \)

\[
\text{SUBST}(\theta, p_1 \lor \cdots \lor p_{i-1} \lor p_{i+1} \lor \cdots \lor p_k \lor q_1 \lor \cdots \lor q_{j-1} \lor q_{j+1} \lor \cdots \lor q_n)
\]

- For example,

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

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

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

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

- Apply resolution steps to \( \text{CNF}(KB \land \neg \alpha) \); complete for FOL
Resolution proof: definite clauses

\[ \neg American(x) \lor \neg Weapon(y) \lor \neg Sells(x,y,z) \lor \neg Hostile(z) \lor Criminal(x) \]

\[ \neg American(West) \]

\[ \neg Missle(x) \lor Weapon(x) \]

\[ \neg Missle(x) \lor \neg Owns(Nono,x) \lor Sells(West,x,Nono) \]

\[ \neg Sells(West,M1,z) \lor \neg Hostile(z) \]

\[ Missle(M1) \]

\[ \neg Missle(M1) \lor \neg Owns(Nono,M1) \lor \neg Hostile(Nono) \]

\[ Owns(Nono,M1) \]

\[ \neg Enemy(x,America) \lor Hostile(x) \]

\[ \neg Hostile(Nono) \]

\[ Enemy(Nono,America) \]

\[ Enemy(Nono,America) \]
Logic programming: Prolog

• FOL:
  \[\text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x)\]
  \[\text{Greedy}(y)\]
  \[\text{King}(\text{John})\]

• Prolog:
  \[\text{evil}(X) :- \text{king}(X), \text{greedy}(X).
  \text{greedy}(Y).
  \text{king}(\text{john}).\]

• Closed-world assumption:
  – Every constant refers to a unique object
  – Atomic sentences not in the database are assumed to be false

• Inference by backward chaining, clauses are tried in the order in which they are listed in the program, and literals (predicates) are tried from left to right
Prolog example

parent(abraham,ishmael).
parent(abraham,isaac).
parent(isaac,esau).
parent(isaac,jacob).

grandparent(X,Y) :- parent(X,Z), parent(Z,Y).
descendant(X,Y) :- parent(Y,X).
descendant(X,Y) :- parent(Z,X), descendant(Z,Y).

? parent(david,solomon).
? parent(abraham,X).
? grandparent(X,Y).
? descendant(X,abraham).
Prolog example

parent(abraham,ishmael).
pARENT(abraham,isaac).
pARENT(ISAAC,ESAU).
pARENT(ISAAC,JACOB).

• What if we wrote the definition of descendant like this:
descendant(X,Y) :- descendant(Z,Y), parent(Z,X).
descendant(X,Y) :- parent(Y,X).

? descendant(W,abraham).

• Backward chaining would go into an infinite loop!
  – Prolog inference is **not complete**, so the ordering of the clauses and the literals is really important
colorable(Wa,Nt,Sa,Q,Nsw,V) :-
  diff(Wa,Nt), diff(Wa,Sa), diff(Nt,Q), diff(Nt,Sa), diff(Q,Nsw),
  diff(Q,Sa), diff(Nsw,V), diff(Nsw,Sa), diff(V,Sa).

diff(red,blue).  diff(red,green).  diff(green,red).
diff(green,blue).  diff(blue,red).  diff(blue,green).
Prolog lists

- Appending two lists to produce a third:
  
  ```prolog
  append([], Y, Y).
  append([X|L], Y, [X|Z]) :- append(L, Y, Z).
  ```

- query: `append(A, B, [1,2])`

- answers: 
  - A=[], B=[1,2]
  - A=[1,2], B=[]