Building KBs in Prolog and Inference in FOPC

Deepak Kumar
November 2021

Knowledge Engineering in FOPC

• Identify the task
• Assemble relevant knowledge
• Decide on a vocabulary of predicates, functions, and constants
• Encode general knowledge about the domain
• Encode a description of the specific problem instance
• Pose queries to the inference procedure and get answers
• Debug the knowledge base
Forward Chaining Inference

• Tell-Ask Systems

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

Inference Engine
(Forward Chaining)

Tell Greedy(John)

\[ \text{Evil}(\text{John})! \]

Backward Chaining Inference

• Tell-Ask Systems

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

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

Inference Engine
(Backward Chaining)

Ask Evil(John)?

Yes!
Backward Chaining Inference

• Tell-Ask Systems

\[
\begin{align*}
\text{King(John)} \\
\text{Greedy(John)} \\
\forall x \ (\text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x)) \\
+ \\
\text{Inference Engine} \\
(\text{Backward Chaining})
\end{align*}
\]

Ask Evil(John)?
Yes!

• Requires wffs to be in Definite Clause form!

Definite Clauses & Horn Clauses

• **Clause**: A disjunction of literals
  e.g. \(\neg R \lor \neg P \lor \neg Q\)

• **Definite Clause**: A clause with exactly one positive literal
  e.g. \(\neg R \lor P \lor \neg Q\)

• **Horn Clause**: A clause with at most one positive literal
  e.g. \(\neg R \lor \neg Q\)
  \(\neg R \lor \neg Q\)
  \(P\)

All definite clauses are Horn Clauses.
Definite Clauses in FOPC

• Disjunction of literals of which exactly one is positive
  \( \neg \omega_1 \lor \neg \omega_2 \lor \ldots \lor \neg \omega_{n-1} \lor \omega_n \)

• A definite clause is either a fact:
  \( \text{American(JoeBiden)} \)

• Or, an implication whose antecedent is a conjunction of positive literals

• Can have variables, but all must be universally quantified (\( \forall \))

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

• We can then rewrite the wff without the quantifier:
  \( \text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x) \)
  \( \neg \text{King}(x) \lor \neg \text{Greedy}(x) \lor \text{Evil}(x) \)

• Most Knowledge bases can be converted to this form.
Logic Programming (Prolog)

• A program is a set of definite clauses (facts, $\omega_1 \land \omega_2 \land \ldots \land \omega_{n-1} \Rightarrow \omega_n$)

• The Syntax is different from FOPC
  • Variables : written in uppercase
  • Constants : written in lowercase
  • Relations : written beginning with lowercase letter
  • Conjunction ($\land$) : comma (,)
  • Implications : written as Prolog rules

$$\omega_1 \land \omega_2 \Rightarrow \omega_n$$ : $C \leftarrow A, B.$

$$\forall x [\text{Missile}(x) \Rightarrow \text{Weapon}(x)]$$ : weapon(X) :- missile(X).

$$\forall x [\text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x)]$$ : evil(X) :- king(X), greedy(x).

A Simple Prolog Program

[Diagram of a network of people with relationships indicated]
A Simple Prolog Program

parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

? :- parent(bob, pat).
true

? :- parent(liz, pat).
false
A Simple Prolog Program

parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

?- parent(bob, pat).
true

?- parent(liz, pat).
false

?- parent(tom, deepak).
false

A Simple Prolog Program

parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

?- parent(X, liz).
X = tom
A Simple Prolog Program

parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

?- parent(X, liz).
X = tom

?- parent(bob, X)
X = ann

What about Pat???
parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

?- parent(X, liz).
X = tom

?- parent(bob, X)
X = ann; What about Pat???
X = pat

?- parent(X, liz).
X = tom

?- parent(bob, X)
X = ann; What about Pat???
X = pat
A Simple Prolog Program

parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

?- parent(X, Y).
X = pam
Y = bob
A Simple Prolog Program

parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

?- parent(X, Y).
X = pam
Y = bob;

X = tom
Y = bob
X = tom,
Y = liz ;
X = bob,
Y = ann ;
...

A Simple Prolog Program

parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).
female(pam).
female(liz).
female(ann).
female(pat).
male(tom).
male(bob).
male(jim).
A Simple Prolog Program

- parent(pam, bob).
- parent(tom, bob).
- parent(tom, liz).
- parent(bob, ann).
- parent(bob, pat).
- parent(pat, jim).
- female(pam).
- female(liz).
- female(ann).
- female(pat).
- male(tom).
- male(bob).
- male(jim).
- mother(X, Y) :- parent(X, Y), female(X).

From FOPC To Prolog

- Knowledge Base – General statements about family relationships

  GrandParent(x, y) : x is a grand parent of y 
  ∀x ∀y ∀z [Parent(x, y) ∧ Parent(y, z) ⇒ GrandParent(x, z)]

  Sibling(x, y) : x is a sibling of y 
  ∀x ∀y ∀z ∀w [father(z, x) ∧ father(z, y) ∧ mother(w, x) ∧ mother(w, y) ∧ (x≠y) ⇒ Sibling(x, y) ]

  AuntOrUncle(x, y) : x is an aunt or uncle of y 
  ∀x ∀y ∀w [Sibling(x, y) ∧ Parent(y, w) ⇒ AuntOrUncle(x, w)]

  ∀w ∀x ∀y ∀w [Married(x, y) ∧ Sibling(y, w) ∧ Parent(w, z) ⇒ AuntOrUncle(x, z)]

  Aunt(x, y) : x is an aunt of y 
  ∀x ∀y [Female(x) ∧ AuntOrUncle(x, y) ⇒ Aunt(x, y)]

  Ancestor(x, y) : x is an ancestor 
  ∀x ∀y [Parent(x, y) ⇒ Ancestor(x, y)]

  ∀x ∀y ∀z [Parent(x, z) ∧ Ancestor(z, y) ⇒ Ancestor(x, y)]
From FOPC To Prolog

- Knowledge Base – General statements about family relationships

\[
\text{GrandParent}(x, y) : x \text{ is a grand parent of } y \\
\forall x \forall y \forall z \left[ \text{Parent}(x, y) \land \text{Parent}(y, z) \Rightarrow \text{GrandParent}(x, z) \right]
\]

\[
\text{Sibling}(x, y) : x \text{ is a sibling of } y \\
\forall x \forall y \forall z \left[ \text{father}(z, x) \land \text{father}(z, y) \land \text{mother}(w, x) \land \text{mother}(w, y) \land (x \neq y) \Rightarrow \text{Sibling}(x, y) \right]
\]

\[
\text{AuntOrUncle}(x, y) : x \text{ is an aunt or uncle of } y \\
\forall x \forall y \forall w \left[ \text{Sibling}(x, y) \land \text{Parent}(y, w) \Rightarrow \text{AuntOrUncle}(x, w) \right]
\]

\[
\forall w \forall x \forall y \forall w \left[ \text{Married}(x, y) \land \text{Sibling}(y, w) \land \text{Parent}(w, z) \Rightarrow \text{AuntOrUncle}(x, z) \right]
\]

\[
\text{Aunt}(x, y) : x \text{ is an aunt of } y \\
\forall x \forall y \left[ \text{Female}(x) \land \text{AuntOrUncle}(x, y) \Rightarrow \text{Aunt}(x, y) \right]
\]

\[
\text{Ancestor}(x, y) : x \text{ is an ancestor} \\
\forall x \forall y \left[ \text{Parent}(x, y) \Rightarrow \text{Ancestor}(x, y) \right]
\]

\[
\forall x \forall y \forall z \left[ \text{Parent}(x, z) \land \text{Ancestor}(z, y) \Rightarrow \text{Ancestor}(x, y) \right]
\]

\[
\text{grandparent}(X, Z) : \neg \text{parent}(X, Y), \text{parent}(Y, Z).
\]

\[
\text{grandparent}(X, Z) : \text{parent}(X, Y), \text{parent}(Y, Z).
\]

\[
\text{grandparent}(X, Z) : \text{parent}(X, Y), \text{parent}(Y, Z).
\]

\[
\text{sibling}(X, Z) : \text{father}(Z, X), \text{father}(Z, Y), \text{mother}(W, X), \text{mother}(W, Y), \neg (X = Y).
\]
From FOPC To Prolog

• Knowledge Base – General statements about family relationships

GrandParent(x, y) : x is a grand parent of y
∀x ∀y ∀z [Parent(x, y) ∧ Parent(y, z) ⇒ GrandParent(x, z)]

Sibling(x, y): x is a sibling of y
∀x ∀y ∀z [father(z, x) ∧ father(z, y) ∧ mother(w, x) ∧ mother(w, y) ∧ (x≠y) ⇒ Sibling(x, y )]

AuntOrUncle(x, y): x is an aunt or uncle of y
∀x ∀y ∀w [Sibling(x, y) ∧ Parent(y, w) ⇒ AuntOrUncle(x, w)]
∀w ∀x ∀y ∀w [Married(x, y) ∧ Sibling(y, w) ∧ Parent(w, z) ⇒ AuntOrUncle(x, z)]
Aunt(x, y): x is an aunt of y
∀x ∀y [Female(x) ∧ AuntOrUncle(x, y) ⇒ Aunt(x, y)]

Ancestor(x, y): x is an ancestor
∀x ∀y [Parent(x, y) ⇒ Ancestor(x, y)]
∀x ∀y ∀z [Parent(x, z) ∧ Ancester(z, y) ⇒ Ancester(x, y)]

grandparent(X, Z) :- parent(X, Y), parent(Y, Z).
sibling(X, Z) :- father(Z, X), father(Z, Y),
                mother(W, X), mother(W, Y), not(X = Y).
auntoruncle(X, W) :- sibling(X, Y), parent(Y, W).
auntoruncle(X, Z) :- married(X, Y), sibling(Y, W), parent(W, Z).
aunt(X, W) :- female(X), auntoruncle(X, W).
uncle(X, W) :- male(X), auntoruncle(X, W).

From FOPC To Prolog

• Knowledge Base – General statements about family relationships

GrandParent(x, y) : x is a grand parent of y
∀x ∀y ∀z [Parent(x, y) ∧ Parent(y, z) ⇒ GrandParent(x, z)]

Sibling(x, y): x is a sibling of y
∀x ∀y ∀z [father(z, x) ∧ father(z, y) ∧ mother(w, x) ∧ mother(w, y) ∧ (x≠y) ⇒ Sibling(x, y )]

AuntOrUncle(x, y): x is an aunt or uncle of y
∀x ∀y ∀w [Sibling(x, y) ∧ Parent(y, w) ⇒ AuntOrUncle(x, w)]
∀w ∀x ∀y ∀w [Married(x, y) ∧ Sibling(y, w) ∧ Parent(w, z) ⇒ AuntOrUncle(x, z)]
Aunt(x, y): x is an aunt of y
∀x ∀y [Female(x) ∧ AuntOrUncle(x, y) ⇒ Aunt(x, y)]

Ancestor(x, y): x is an ancestor
∀x ∀y [Parent(x, y) ⇒ Ancestor(x, y)]
∀x ∀y ∀z [Parent(x, z) ∧ Ancester(z, y) ⇒ Ancester(x, y)]

grandparent(X, Z) :- parent(X, Y), parent(Y, Z).
sibling(X, Z) :- father(Z, X), father(Z, Y),
                mother(W, X), mother(W, Y), not(X = Y).
auntoruncle(X, W) :- sibling(X, Y), parent(Y, W).
auntoruncle(X, Z) :- married(X, Y), sibling(Y, W), parent(W, Z).
aunt(X, W) :- female(X), auntoruncle(X, W).
uncle(X, W) :- male(X), auntoruncle(X, W).
From FOPC To Prolog

- Knowledge Base – General statements about family relationships

GrandParent(x, y) : x is a grand parent of y
∀x ∀y ∀z [Parent(x, y) ∧ Parent(y, z) ⇒ GrandParent(x, z)]

Sibling(x, y) : x is a sibling of y
∀x ∀y ∀z ∀w [ father(z, x) ∧ father(z, y) ∧ mother(w, x) ∧ mother(w, y) ∧ (x≠y) ⇒ Sibling(x, y) ]

AuntOrUncle(x, y) : x is an aunt or uncle of y
∀x ∀y ∀w [Sibling(x, y) ∧ Parent(y, w) ⇒ AuntOrUncle(x, w)]

Aunt(x, y) : x is an aunt of y
∀x ∀y [Female(x) ∧ AuntOrUncle(x, y) ⇒ Aunt(x, y)]

Ancestor(x, y) : x is an ancestor
∀x ∀y [Parent(x, y) ⇒ Ancestor(x, y)]
∀x ∀y ∀z [Parent(x, z) ∧ Ancestor(z, y) ⇒ Ancestor(x, y)]

grandparent(X, Z) :- parent(X, Y), parent(Y, Z).
sibling(X, Z) :- father(Z, X), father(Z, Y),
               mother(W, X), mother(W, Y), not(X = Y).
auntoruncle(X, W) :- sibling(X, Y), parent(Y, W).
auntoruncle(X, Z) :- married(X, Y), sibling(Y, W), parent(W, Z).
aunt(X, W) :- female(X), auntoruncle(X, W).
uncle(X, W) :- male(X), auntoruncle(X, W).

From FOPC To Prolog

- Knowledge Base – General statements about family relationships

GrandParent(x, y) : x is a grand parent of y
∀x ∀y ∀z [Parent(x, y) ∧ Parent(y, z) ⇒ GrandParent(x, z)]

Sibling(x, y) : x is a sibling of y
∀x ∀y ∀z ∀w [ father(z, x) ∧ father(z, y) ∧ mother(w, x) ∧ mother(w, y) ∧ (x≠y) ⇒ Sibling(x, y) ]

AuntOrUncle(x, y) : x is an aunt or uncle of y
∀x ∀y ∀w [Sibling(x, y) ∧ Parent(y, w) ⇒ AuntOrUncle(x, w)]

Aunt(x, y) : x is an aunt of y
∀x ∀y [Female(x) ∧ AuntOrUncle(x, y) ⇒ Aunt(x, y)]

Ancestor(x, y) : x is an ancestor
∀x ∀y [Parent(x, y) ⇒ Ancestor(x, y)]
∀x ∀y ∀z [Parent(x, z) ∧ Ancestor(z, y) ⇒ Ancestor(x, y)]

grandparent(X, Z) :- parent(X, Y), parent(Y, Z).
sibling(X, Z) :- father(Z, X), father(Z, Y),
               mother(W, X), mother(W, Y), not(X = Y).
auntoruncle(X, W) :- sibling(X, Y), parent(Y, W).
auntoruncle(X, Z) :- married(X, Y), sibling(Y, W), parent(W, Z).
aunt(X, W) :- female(X), auntoruncle(X, W).
uncle(X, W) :- male(X), auntoruncle(X, W).
ancestor(X, Y) :- parent(X, Y).
ancestor(X, Y) :- parent(X, Z), ancestor(Z, Y).
Knowledge Engineering in FOPC

- Identify the task
- Assemble relevant knowledge
- Decide on a vocabulary of predicates, functions, and constants
- Encode general knowledge about the domain
- Encode a description of the specific problem instance
- Pose queries to the inference procedure and get answers
- Debug the 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.
Knowledge Engineering in FOPC

• Identify the task
• Assemble relevant knowledge

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.

• Decide on a vocabulary of predicates, functions, and constants

American(x) : x is an American
Enemy(x, y) : x is an enemy of y
Hostile(x) : x is hostile
Criminal(x) : x is a criminal
Missile(x) : x is a missile
Weapon(x) : x is a weapon
Owns(x, y) : x owns y
Sells(x, y, z) : x sells y to z
Knowledge Engineering in FOPC

• Identify the task
• Assemble relevant knowledge

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.

• Decide on a vocabulary of predicates, functions, and constants

American(x): x is an American
Enemy(x, y): x is an enemy of y
Hostile(x): x is hostile
Criminal(x): x is a criminal
Missile(x): x is a missile
Weapon(x): x is a weapon
Owns(x, y): x owns y
Sells(x, y, z): x sells y to z

Knowledge Engineering in FOPC

• Identify the task
• Assemble relevant knowledge

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.

• Decide on a vocabulary of predicates, functions, and constants

• Encode general knowledge about the domain

∀x ∀y ∀z [American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)]
Knowledge Engineering in FOPC

- Identify the task
- Assemble relevant knowledge

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.

- Decide on a vocabulary of predicates, functions, and constants
- Encode general knowledge about the domain

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

∃x [Owns(Nono, x) ∧ Missile(x)]

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

∃x [Owns(Nono, x) ∧ Missile(x)]

∀x [Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)]
Knowledge Engineering in FOPC

- Identify the task
- Assemble relevant knowledge

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.

- Decide on a vocabulary of predicates, functions, and constants
- Encode general knowledge about the domain

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

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

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

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

Also, all missiles are weapons. [Implicit knowledge]

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

An enemy of America is hostile. [Implicit knowledge]
Knowledge Engineering in FOPC

- Identify the task
- Assemble relevant knowledge

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.

- Decide on a vocabulary of predicates, functions, and constants
- Encode general knowledge about the domain

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

∃x [Owns(Nono, x) ∧ Missile(x)]

∀x [Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)]

∀x [Missile(x) ⇒ Weapon(x)] Also, all missiles are weapons. [Implicit knowledge]

∀x [Enemy(x, America) ⇒ Hostile(x)] An enemy of America is hostile. [Implicit knowledge]

- Encode a description of specific problem instance

∀x ∀y ∀z [American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)]
Knowledge Engineering in FOPC

- Identify the task
- Assemble relevant knowledge
  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.
- Decide on a vocabulary of predicates, functions, and constants
- Encode general knowledge about the domain
  \[ \forall x \forall y \forall z \ [\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)] \]
  \[ \exists x \ [\text{Owns}(\text{Nono}, x) \land \text{Missile}(x)] \]
  \[ \forall x \ [\text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{CWest}, x, \text{Nono})] \]
  \[ \forall x \ [\text{Missile}(x) \Rightarrow \text{Weapon}(x)] \]
  Also, all missiles are weapons. [Implicit knowledge]
  \[ \forall x \ [\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)] \]
  An enemy of America is hostile. [Implicit knowledge]
- Encode a description of specific problem instance
  American(Cwest)

Knowledge Engineering in FOPC

- Identify the task
- Assemble relevant knowledge
  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.
- Decide on a vocabulary of predicates, functions, and constants
- Encode general knowledge about the domain
  \[ \forall x \forall y \forall z \ [\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)] \]
  \[ \exists x \ [\text{Owns}(\text{Nono}, x) \land \text{Missile}(x)] \]
  \[ \forall x \ [\text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{CWest}, x, \text{Nono})] \]
  \[ \forall x \ [\text{Missile}(x) \Rightarrow \text{Weapon}(x)] \]
  Also, all missiles are weapons. [Implicit knowledge]
  \[ \forall x \ [\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)] \]
  An enemy of America is hostile. [Implicit knowledge]
- Encode a description of specific problem instance
  American(Cwest)
  Enemy(Nono, America)
Knowledge Engineering in FOPC

- Knowledge Base

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

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

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

\[ \forall x \ [\text{Missile}(x) \Rightarrow \text{Weapon}(x)] \quad \text{Also, all missiles are weapons. [Implicit knowledge]} \]

\[ \forall x \ [\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)] \quad \text{An enemy of America is hostile. [Implicit knowledge]} \]

\text{American(}\text{CWest})

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

- Pose queries to the inference procedure to get answers

Is Colonel West a criminal?

FOPC – Inference Rules

- Inference rules for Quantifiers (\(\forall\), \(\exists\))
  - Universal Instantiation
  - Existential Instantiation

- Generalized Modus Ponens
  - Unification

- Forward & Backward Chaining
  - Definite Clauses
  - Logic Programming in Prolog

- Resolution
  - Reductio ad Absurdum
Inference Rules for Quantifiers (∀, ∃)

• Universal Instantiation Rule

Can infer any sentence obtained by substituting a ground term (a term without variables) for a universally quantified variable (∀)

 e.g.

∀x [King(x) ∧ Greedy(x) ⇒ Evil(x)]

We can replace/substitute John (a ground term) for x in that wff:

King(John) ∧ Greedy(John) ⇒ Evil(John) when {x = John}

Substitutions ({x = John}) formally

A substitution is written as

θ = { v/x } - replace v by x

Given a wff, α

SUBST(θ, α) - apply substitution θ to α

 e.g. θ = { x/John } α = ∀x [King(x) ∧ Greedy(x) ⇒ Evil(x)]

then SUBST(θ, α) = King(John) ∧ Greedy(John) ⇒ Evil(John)
Inference Rules for Quantifiers ($\forall$, $\exists$)

• Universal Instantiation Rule, formally

\[
\forall v \ [\alpha] \quad \text{SUBST}\{\{v/g\}, \alpha\} \quad \text{for any variable, } v \text{ and ground term, } g.
\]

Can infer any sentence obtained by substituting a ground term (a term without variables) for a universally quantified variable ($\forall$)

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

\[
\text{King}(\text{John}) \land \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John}) \quad \text{when } \{x/\text{John}\}
\]

\[
\text{King}(\text{Richard}) \land \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard}) \quad \text{when } \{x/\text{Richard}\}
\]

...

Inference Rules for Quantifiers ($\forall$, $\exists$)

• Existential Instantiation Rule, formally

\[
\exists v \ [\alpha] \quad \text{SUBST}\{\{v/k\}, \alpha\} \quad \text{for any variable, } v \text{ and a constant symbol, } k.
\]

Can infer a sentence obtained by substituting a constant symbol for a existentially quantified variable ($\exists$)

e.g.
\[
\alpha = \exists x \ [\text{Owns}(\text{Nono}, x) \land \text{Missile}(x)]
\]

Let constant be M1, then SUBST($\{x/M1\}$, $\alpha$) gives

\[
\text{Owns}(\text{Nono}, \text{M1}) \land \text{Missile}(\text{M1})
\]
Inference Rules for Quantifiers ($\forall, \exists$)

- Existential Instantiation Rule, formally

$$\exists v [\alpha]$$

$$\text{SUBST}\{\{v/k\}, \alpha\}$$

for any variable, $v$ and a constant symbol, $k$.

Can infer a sentence obtained by substituting a constant symbol for a existentially quantified variable ($\exists$)

e.g.

$$\alpha = \exists x [\text{Owns(Nono, x)} \land \text{Missile(x)}]$$

Let constant be M1, then SUBST({x/M1}, $\alpha$) gives

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

M1 is called a Skolem Constant

Generalized Modus Ponens

For atomic sentences $p_i, p_i', q$

where there is a substitution $\theta$ such that

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

$$p_1'$$
$$p_2'$$
$$...$$
$$p_n'$$

$$p_1' \land p_2' \land ... \land p_n' \Rightarrow q$$

$$\text{SUBST}(\theta, q)$$

e.g.

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

$\theta$ is {x/John}

$\text{SUBST}(\theta, \text{Evil(x)})$ is Evil(John)
Unification - \( \text{UNIFY}(p, q) = \theta \)

How to find substitutions that make different logical expressions look identical?

For wffs \( p \) and \( q \) (sentences with universally quantified variables)

\( \text{UNIFY}(p, q) = \theta \)

- \( p = \text{Knows}(\text{John}, x) \)
  \( q = \text{Knows}(\text{John}, \text{Mary}) \)
  \( \theta = \{x/\text{Mary}\} \)

- \( p = \text{Knows}(\text{John}, x) \)
  \( q = \text{Knows}(y, \text{Bill}) \)
  \( \theta = \{y/\text{John}, x/\text{Bill}\} \)

- \( p = \text{Knows}(\text{John}, x) \)
  \( q = \text{Knows}(x, \text{Elizabeth}) \)
  \( \theta = \{\} \) i.e. Fail! (no unification)

- \( p = \text{Knows}(\text{John}, x_1) \)
  \( q = \text{Knows}(x_2, \text{Elizabeth}) \)
  \( \theta = \{x_1/\text{Elizabeth}, x_2/\text{John}\} \)

Forward Chaining Inference

- Tell-Ask Systems

\[
\text{King}(\text{John}) \\
\forall x \ [\text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x)] \\
+ \text{Inference Engine (Forward Chaining)} \\
\Rightarrow \text{Evil}(\text{John})!
\]
Backward Chaining Inference

- Tell-Ask Systems

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

Inference Engine
(Backward Chaining)

Ask Evil(John)?
Yes!

- Requires wffs to be in Definite Clause form!
Forward Chaining

American(Cwest)  Missile(M1)  Owns(Nono, M1)  Enemy(Nono, America)

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Weapon(x) ⇒ Enemy(x, America) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)
Forward Chaining

American(Cwest) \land \text{Weapon(M1)} \land \text{Sells(CWest, M1, Nono)} \land \text{Hostile(Nono)} \Rightarrow \text{Criminal(M1)}

Missile(M1) \land \text{Owns(Nono, M1)} \Rightarrow \text{Sells(CWest, x, Nono)}

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

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

\text{American(Cwest)}

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

Criminal(CWest)

Weapon(M1) Sells(CWest, M1, Nono) Hostile(Nono)

American(Cwest) Missile(M1) Owns(Nono, M1) Enemy(Nono, America)

Backward Chaining

Criminal(CWest)?

American(Cwest) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(x, America) ⇒ Hostile(x)
American(CWest)
Enemy(Nono, America)
Backward Chaining

American(Cwest)?

Criminal(CWest)?

Weapon(y)?

Sells(x, y, z)?

Hostile(z)?

American(Cwest)
Backward Chaining

American(Cwest) ∧ Weapon(M1) ∧ Sells(Cwest, M1, z) ∧ Hostile(z) ⇒ Criminal(Cwest)  
Missile(M1)  
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)  
Missile(x) ⇒ Weapon(x)  
Enemy(America, x) ⇒ Hostile(x)  
American(Cwest)  
Enemy(Nono, America)

Backward Chaining

American(Cwest) ∧ Weapon(M1) ∧ Sells(Cwest, M1, z) ∧ Hostile(z) ⇒ Criminal(Cwest)  
Missile(M1)  
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)  
Missile(x) ⇒ Weapon(x)  
Enemy(America, x) ⇒ Hostile(x)  
American(Cwest)  
Enemy(Nono, America)
Backward Chaining

\[ \text{American}(\text{Cwest}) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]
\[ \text{Owns} (\text{Nono}, \text{M1}) \]
\[ \text{Missile} (\text{M1}) \]
\[ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells} (\text{CWest}, x, \text{Nono}) \]
\[ \text{Hostile}(\text{Nono}) \]

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

Backward Chaining

\[ \text{Criminal}(\text{CWest})? \]
\[ \text{Weapon}(\text{M1}) \]
\[ \text{Sells}(\text{Cwest}, \text{M1}, \text{Nono}) \]
\[ \text{Hostile}(z)? \]
\[ \{y/\text{M1}\} \]
\[ \{z/\text{Nono}\} \]
\[ \{x/\text{Cwest}\} \]
\[ \text{American}(\text{Cwest}) \]
\[ \text{Missile}(\text{M1}) \]
\[ \text{Owns}(\text{Nono}, \text{M1}) \]

Backward Chaining

\[ \text{American}(\text{Cwest}) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]
\[ \text{Owns} (\text{Nono}, \text{M1}) \]
\[ \text{Missile} (\text{M1}) \]
\[ \text{Missile}(x) \Rightarrow \text{Weapon}(x) \]
\[ \text{Enemy}(\text{America}, x) \Rightarrow \text{Hostile}(x) \]
\[ \text{American}(\text{Cwest}) \]
\[ \text{Enemy}(\text{Nono}, \text{America}) \]

\[ \{x/\text{Cwest}\} \]
\[ \text{American}(\text{Cwest}) \]
\[ \text{Missile}(\text{M1}) \]
\[ \text{Owns}(\text{Nono}, \text{M1}) \]
Backward Chaining

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

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

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

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

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

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

\[
\text{Criminal}(CWest)?
\]

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

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

\[
\text{Missile}(x) \land \text{Owns}(Nono, M1) \Rightarrow \text{Sells}(CWest, M1, Nono)
\]

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

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

Backward Chaining

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

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

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

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

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

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

\[
\text{Criminal}(CWest)?
\]

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

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

\[
\text{Missile}(x) \land \text{Owns}(Nono, M1) \Rightarrow \text{Sells}(CWest, M1, Nono)
\]

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

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

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