Inference in First-Order Predicate Calculus

Deepak Kumar
November 2017

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
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
  \[ \forall x \forall y \forall z \ [\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)] \]
  \[ \exists x \ [\text{Owns}(\text{Nono}, x) \land \text{Missile}(x)] \]
  \[ \forall x \ [\text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{CWest}, x, \text{Nono})] \]
  \[ \forall x \ [\text{Missile}(x) \Rightarrow \text{Weapon}(x)] \quad \text{Also, all missiles are weapons. [Implicit knowledge]} \]
  \[ \forall x \ [\text{Enemy}(\text{America}, x) \Rightarrow \text{Hostile}(x)] \quad \text{An enemy of America is hostile. [Implicit knowledge]} \]

• Encode a description of specific problem instance
  \text{American}(\text{Cwest})
  \text{Enemy}(\text{Nono, America})

Knowledge Engineering in FOPC

• Knowledge Base
  \[ \forall x \forall y \forall z \ [\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)] \]
  \[ \exists x \ [\text{Owns}(\text{Nono}, x) \land \text{Missile}(x)] \]
  \[ \forall x \ [\text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{CWest}, x, \text{Nono})] \]
  \[ \forall x \ [\text{Missile}(x) \Rightarrow \text{Weapon}(x)] \quad \text{Also, all missiles are weapons. [Implicit knowledge]} \]
  \[ \forall x \ [\text{Enemy}(\text{America}, x) \Rightarrow \text{Hostile}(x)] \quad \text{An enemy of America is hostile. [Implicit knowledge]} \]
  \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 (∀, ∃)
  • 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 \(\{x = \text{John}\}\) formally

A substitution is written as

\[ \theta = \{v/x\} \quad \text{- replace } v \text{ by } x \]

Given a wff, \(\alpha\)

\[ \text{SUBST}(\theta, \alpha) \quad \text{- apply substitution } \theta \text{ to } \alpha \]

e.g. \(\theta = \{x/\text{John}\}\) \quad \alpha = \forall x [\text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x)]

then \quad \text{SUBST}(\theta, \alpha) = \text{King}(\text{John}) \land \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John})

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

• Universal Instantiation Rule, formally

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

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 an existentially quantified variable ($\forall$)

e.g.

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

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

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

M1 is called a Skolem Constant

Generalized Modus Ponens

For atomic sentences $p_i$, $p'_i$, and $q$

where there is a substitution $\theta$ such that

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

$\begin{align*}
& p'_1 \\
& p'_2 \\
& \vdots \\
& p'_n \\
& p_1 \land p_2 \land \ldots \land p_n \Rightarrow q
\end{align*}$

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

e.g.

\begin{align*}
\text{King(John)} \\
\text{Greedy(John)} \\
\forall x \ [\text{King(x) } \land \text{Greedy(x)} \Rightarrow \text{Evil(x)}] \\
\theta \text{ is } \{x/\text{John}\} \\
\text{SUBST}(\theta, \text{Evil(x)}) \text{ is } \text{Evil(John)}
\end{align*}
Unification - 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)
UNIFY(p, q) = \theta

\begin{align*}
\text{e.g.} & \quad p = \text{Knows(John, x)} \\
& \quad q = \text{Knows(John, Mary)} \\
& \quad \theta = \{x/\text{Mary}\} \\
\text{p = Knows(John, x)} \\
\text{q = Knows(y, Bill)} \\
& \quad \theta = \{y/\text{John}, x/\text{Bill}\} \\
\text{p = Knows(John, x)} \\
\text{q = Knows(x, Elizabeth)} \\
& \quad \theta = \{\} \quad \text{i.e. Fail! (no unification)} \\
\text{p = Knows(John, x}_1) \\
\text{q = Knows(x}_2, \text{Elizabeth)} \\
& \quad \theta = \{x_1/\text{Elizabeth}, x_2/\text{John}\}
\end{align*}

Forward Chaining Inference

• Tell-Ask Systems

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

• Tell-Ask Systems

Recall - 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. \( \neg R \lor P \lor \neg Q \)

All definite clauses are Horn Clauses.
Definite Clauses in FOPC

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

• A definite clause is either a fact: American(Cwest)

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

• Can have variables, but all have to be universally quantified (\( \forall \))
  \( \forall x [\text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x)] \)

• We can then rewrite the wff without the quantifier: King(x) \land Greedy(x) \Rightarrow Evil(x)

• Most Knowledge bases can be converted to this form.

Forward Chaining

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

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  
    \( A \land B \Rightarrow C \)  
    \( C : A, B. \)  
    \( \text{weapon(X) :- missile(X)} \)  
    \( \text{evil(X) :- king(X), greedy(X)}. \)
From FOPC To Prolog

**Knowledge Base**

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

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

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

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

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

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

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

```
criminal(X) :- \text{american}(X), \text{weapon}(Y), \text{sells}(X, Y, Z), \text{hostile}(Z).
owns(\text{nono}, \text{m1}).
\text{missile}(\text{m1}).
sells(cwest, X, nono) :- \text{missile}(X), owns(\text{nono}, X).
\text{weapon}(X) :- \text{missile}(X).
hostile(X) :- \text{enemy}(X, \text{america}).
\text{american}(\text{cwest}).
\text{enemy}(\text{nono}, \text{america}).
```
From FOPC To Prolog

- **Knowledge Base**

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

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

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

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

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

- **American(\text{CWest})**
- **Enemy(\text{America}, \text{Nono})**

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

criminal(X) :- american(X), weapon(Y), sells(X, Y, Z), hostile(Z).
owns(Nono, m1).
missile(m1).
sells(cwest, X, nono) :- missile(X), owns(Nono, X).
weapon(X) :- missile(X).
hostile(X) :- enemy(X, america).
american(cwest).
enemy(Nono, America).

?- criminal(WHO).

A Simpler Prolog Program

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

?- parent(bob, pat).
yes