ARTIFICIAL INTELLIGENCE

CHAPTER – 8 First Order Logic

Propositional logic is a declarative language because its semantics is based on a truth relation between sentences and possible words. It also has sufficient expressive power to deal with partial information using disjunction and negation.

The primary difference between propositional logic and first order logic lies in the ontological commitment made by each language. Propositional logic assumes that there are facts that either hold or do not hold in the world. Each fact can be in one of two states: true or false.

First order logic assumes that the world consists of objects with certain relations among them that do or do not hold.

Syntax of propositional logic –

1. ~A (Negation of A)
2. A & B (Conjunction of A with B)
3. A v B (Inclusive disjunction of A with B)
4. A → B (A implies B)
5. A ⟷ B (Material bi-conditional of A with B)
6. A B (Exclusive disjunction of A with B)
7. A | B (Joint denial of A with B)
8. A ↓ B (Disjoint denial of A with B)

Syntax and semantics of first order logic:

Propositional logic cannot be used to represent knowledge of complex environments. First order logic is used to represent our common sense knowledge.

Models for propositional logic are sets of truth values for proposition symbols. Models for first order logic have objects.

Domain of model is set of objects it contains. These objects are called domain elements.

Relation is a set of tuples of objects that are related. Tuple is a collection of objects arranged in fixed order.

Models in first order logic require total functions – that is there must be a value for every input tuple. The basic syntactic elements of first order logic are the symbols that stand for objects, relations and functions. The language of first order logic is built around objects and relations.

Symbols come in 3 kinds –

a) Constant: stands for objects where objects can be nouns and noun phrases.
b) Predicate: stands for relations or properties where predicates are verbs and verb phrases.
c) Function: relation in which there is only one value for a given input.

Ex: 1. one plus two gives three

Objects are one, two and three

Predicate (Relation) is equals

Function is plus

2. Squares neighboring wumpus are smelly

Objects are wumpus and squares.

Predicate (Property) is smelly

Relation is neighboring

Each predicate and function symbol comes with an arity that fixes the number of arguments.

Interpretation specifies exactly which objects, relations and functions are referred to by the constant, predicate and function symbols.

Syntax of first order logic specified in Backus – Naur form:

$$
\begin{array}{rcl}
Sentence & \rightarrow & AtomicSentence \\
& | & (\ Sentence\ Connective\ Sentence\ ) \\
& | & Quantifier\ Variable, \ldots\ Sentence \\
& | & \neg\ Sentence \\
\\
AtomicSentence & \rightarrow & Predicate(Term, \ldots) \mid Term = Term \\
\\
Term & \rightarrow & Function(Term, \ldots) \\
& | & Constant \\
& | & Variable \\
\\
Connective & \rightarrow & \Rightarrow \mid \wedge \mid \vee \mid \Leftrightarrow \\
Quantifier & \rightarrow & \forall \mid \exists \\
Constant & \rightarrow & A \mid X_1 \mid John \mid \cdots \\
Variable & \rightarrow & a \mid x \mid s \mid \cdots \\
Predicate & \rightarrow & Before \mid HasColor \mid Raining \mid \cdots \\
Function & \rightarrow & Mother \mid LeftLeg \mid \cdots
\end{array}
$$

A term is a logical expression that refers to an object. A complex term is formed by a function symbol followed by a parenthesized list of terms as arguments to the function symbol.

Atomic sentence is formed from a predicate symbol followed by a parenthesized list of terms. An atomic sentence is true in a given model under a given interpretation, if the relation referred to by the predicate symbol holds among the objects referred to by the arguments.

Logical connectives are used to construct more complex sentences.

Quantifiers allow us to express properties of entire collections of objects. First order logic contains two standard quantifiers called universal and existential.

Universal quantifier:

$$\forall x \;\; King(x) \;\Rightarrow\; Person(x)$$

where it is read as – For all x, if x is a king, then x is a person. Symbol x is called a variable. A variable is a term all by itself. A term with no variables is called a ground term. By convention, variables are lowercase letters.

Existential quantification:

A statement can be made about some object in the universe without naming it by using existential quantifier.

$$\exists x \;\; Crown(x) \wedge OnHead(x, John)$$

where it is read as there exists a crown on the head of King John.

CHAPTER – 9 Inference In First Order Logic

Simple inference rules can be applied to sentences with quantifiers to obtain sentences without quantifiers. These rules lead to the idea that first order inference can be done by converting the knowledge base to propositional logic and using propositional inference.

Inference rules for quantifiers-

a) Rule of Universal Instantiation: any sentences can be inferred by substituting a ground term (a term without variables) for the variable.
SUBST (ɵ,α) denotes the result of applying the substitution ɵ to sentence α Then the rule is written as –

$$\frac{\forall v \ \ \alpha}{\text{SUBST}(\{v/g\}, \alpha)}$$

for any variable v and ground term g.

b) Rule of Existential Instantiation: For any sentence α, variable v and constant symbol k that does not appear elsewhere in the knowledge base,

$$\frac{\exists v \ \ \alpha}{\text{SUBST}(\{v/k\}, \alpha)}$$

Universal Instantiation can be applied many times to produce many different consequences, Existential Instantiation can be applied once and then the existentially quantified sentence can be discarded.

Unification and Lifting:

Unification is a procedure that compares two literals & discovers whether there exists a set of substitutions that makes them identified. Any substitution that makes 2 or more expressions equal is called a unifier for the expressions. Unification can sometimes be applied to literals within the same single clause. When an mgu exists such that 2 or more literals with in a clause are unified, the clause remaining after deletion of all but one of the unified literals is called a factor of the original clause.

The basic idea of unification is very simple - to attempt to unify 2 literals, we first check if their initial predicate symbols are the same, if so we can proceed otherwise there is no way they can be unified, regardless of their arguments.

Unification has deep mathematical roots and is a useful in many AI programs. for ex: theorem proving and natural language parser.

First order inference rule-

If there is some substitution ɵ that makes the premise of the implication identical to sentences already in the knowledge base, then conclusion of the implication can be asserted after applying ɵ.

This inference rule can be written as a single inference rule called as Generalized Modus Ponens.

For atomic sentences p_i, p_i` and q, where there is a substitution ɵ such that

SUBST (ɵ, p_i') = SUBST(ɵ, p_i') for all i,

$$\frac{p_1', \quad p_2', \quad \ldots, \quad p_n', \quad (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{\text{SUBST}(\theta, q)}$$

There is n+ 1 premise to this rule – the n atomic sentences p_i' and the one implication. The conclusion is the result of applying the substitution ɵ to the consequent q.

Generalized Modus Ponens is a lifted version of Modus Ponens – it raises Modus Ponens from propositional to first order logic. The key advantage of lifted inference rules over propositionalization is that they make only those substitutions which are required to allow particular inferences to proceed. Disadvantage is that Modus Ponens allows any single α on the left hand side of the implication while Generalized Modus Ponens requires a special format for the sentence.

Unification: Lifted inference rules require substitutions that make different logical expressions look identical. This process is called unification and is a key component of all first order inference algorithms. The UNIFY algorithm takes two sentences and returns a unifier for them if one exists.

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

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 (optional, defaults to empty)

  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

function UNIFY-VAR(var, x, θ) returns a substitution
  inputs: var, a variable
          x, any expression
          θ, the substitution built up so far

  if {var/val} ∈ θ then return UNIFY(val, x, θ)
  else if {x/val} ∈ θ then return UNIFY(var, val, θ)
  else if OCCUR-CHECK?(var, x) then return failure
  else return add {var/x} to θ
```

Forward chaining:

In the forward chaining algorithm – start with atomic sentences in the knowledge base and apply Modus Ponens in the forward direction adding new atomic sentences until no further inferences can be made.

First order definite clauses are disjunctions of literals of which exactly one is positive. A definite clause either is atomic or is an implication whose antecedent is a conjunction of positive literals and whose consequent is a single positive literal. First order literals can include variables in which case those variables are assumed to be universally quantified. Every knowledge base cannot be converted into a set of definite clauses because of the single positive literal restriction.

Ex: The law says that it is crime for an American to sell weapons to hostile nations. The country AB, an enemy of America, has some missiles and all of its missiles were sold to it by Colonel West, who is an American.

Goal will be prove that west is the Criminal. Facts have to be represented as first – order definite clauses. Then problem has to be solved with forward chaining algorithm.

"..It is a crime for an American to sell weapons to hostile nations"

$$American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow Criminal(x)$$

AB has some missiles. All of its missiles were sold to it by Colonel West.

Missile(x) ^ Owns(AB, x) » Sells(West, x, AB)

Missiles can also be called weapons -- Missile(x) »Weapon(x)

Enemy of America is counted as hostile – Enemy (x, America) »Hostile(x)

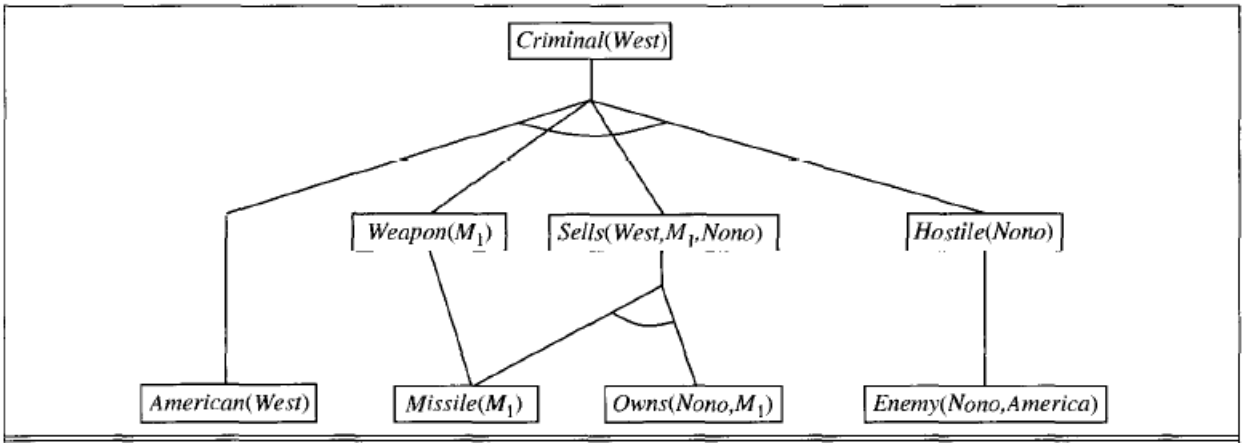
West is an American – American (West)

The country AB is an enemy of America – Enemy (AB, America)

Backward Chaining –

Logical inference algorithms using backward chaining approach work backward from the goal, chaining through rules to find known facts that support the proof. Backward chaining algorithm is called with a list of goals containing a single element, the original query and returns the set of all substitutions satisfying the query. The list of goals can be thought of as a stack waiting to be worked on.

Proof tree constructed by backward chaining to prove that West is a criminal.



Resolution-

The resolution procedure is a simple iterative process. At each step two clauses, called the parent clause are compared (resolved) yielding a new clause that has been inferred from them. The new clause represents ways that the 2 parent's clauses interact with each other.

Resolution work on the principle of identity complementary literals in 2 clauses and deleting them there by forming a new literal . the process is simple and straight forward when one has identical literals . In other words for clauses containing no variables resolution is easy. When there are variables the problem becomes complicated and the necessities one to make proper substitutions.

Propositional resolution is a complete inference procedure for propositional logic.

Conjunctive normal form for first order logic –

First order resolution requires that sentences be in conjunctive normal form that is conjunction of clauses where each clause is a disjunction of literals. Literals can contain variables which are assumed to be universally quantified.

For example –

$$\forall x \;\; American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \;\Rightarrow\; Criminal(x)$$

becomes in CNF,

$$\neg American(x) \vee \neg Weapon(y) \vee \neg Sells(x, y, z) \vee \neg Hostile(z) \vee Criminal(x)$$

Every sentence of first order logic can be converted into an inferentially equivalent CNF sentence.

Procedure for conversion to CNF –

"Everyone who loves all animals is loved by someone".

(or)

$$\forall x \ [\forall y \ Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists y \ Loves(y, x)]$$

Steps are –

a) Eliminate implications

$$\forall x \ [\neg \forall y \ \neg Animal(y) \vee Loves(x, y)] \vee [\exists y \ Loves(y, x)]$$

b) Move ¬ inwards

$\neg \forall x \ p$ becomes $\exists x \ \neg p$

$\neg \exists x \ p$ becomes $\forall x \ \neg p$

The sentence given goes through the following transformations

$$\forall x \ [\exists y \ \neg(\neg Animal(y) \vee Loves(x, y))] \vee [\exists y \ Lovcs(y, x)]$$
$$\forall x \ [\exists y \ \neg\neg Animal(y) \wedge \neg Loves(x, y)] \vee [\exists y \ Loves(y, x)]$$
$$\forall x \ [\exists y \ Animal(y) \wedge \neg Loves(x, y)] \vee [\exists y \ Loves(y, x)] \ .$$

c) Standardize variables – changing the name of one of the variables.

$$\forall x \ [\exists y \ Animal(y) \wedge \neg Loves(x, y)] \vee [\exists z \ Loves(z, x)]$$

d) Skolemize – Skolemization is the process of removing existential quantifiers by elimination.

$$\forall x \ [Animal(A) \wedge \neg Loves(x, A)] \vee Loves(B, x)$$

which conveys a wrong meaning entirely. Thus, we want Skolem entities to depend on x.

$$\forall x \ [Animal(F(x)) \wedge \neg Loves(x, F(x))] \vee Loves(G(x), x)$$

F and G are skolem functions. Arguments of Skolem function are all universally quantified variables in whose scope the existential quantifier appears. Skolemized sentence is satisfiable exactly when original sentence is satisfiable.

e) Drop universal quantifiers – Universally quantify all remaining variables.

$$[Animal(F(x)) \wedge \neg Loves(x, F(x))] \vee Loves(G(x), x)$$

f) Distribute ^ over ˇ

$$[Animal(F(x)) \vee Loves(G(x), x)] \wedge [\neg Loves(x, F(x)) \vee Loves(G(x), x)]$$

The sentence is now in CNF and consists of two clauses.

Resolution inference rule –

Two clauses which are assumed to be standardized apart so that they share no variables can be resolved if they contain complementary literals. Propositional literals are complementary if one is negation of the other, first order literals are complementary if one unifies with negation of the other. So, we have

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{\text{SUBST}(\theta, \ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)}$$

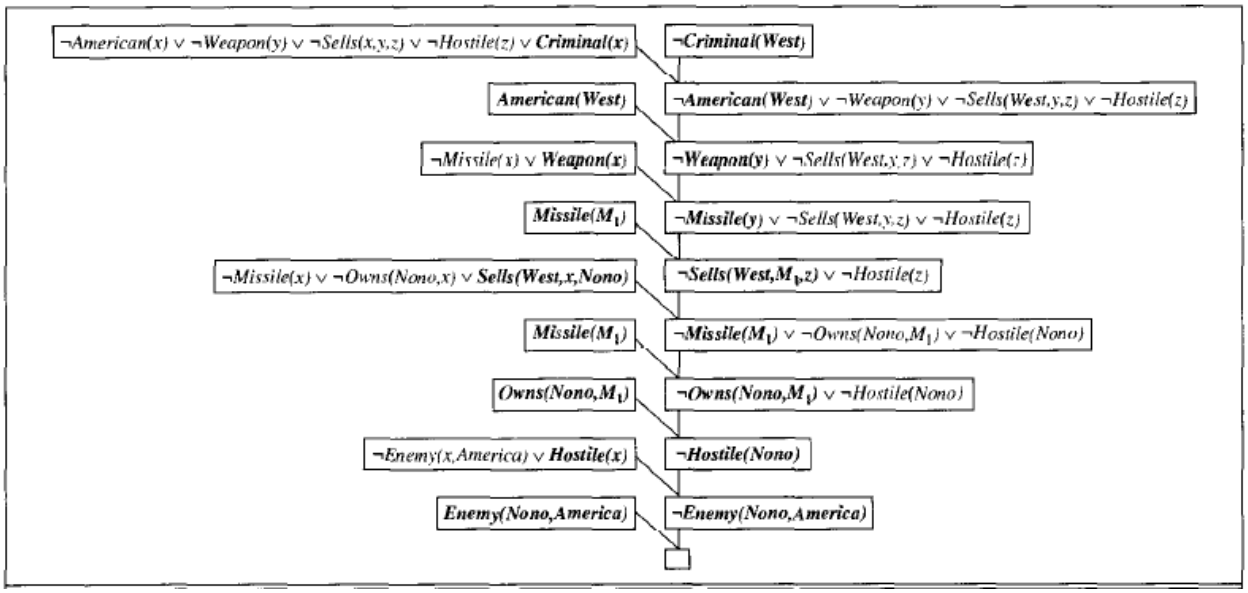
Where UNIFY (l_i, $\neg m_j$)=ɵ

Example proof –

Sentences in CNF are –

$$\neg American(x) \vee \neg Weapon(y) \vee \neg Sells(x, y, z) \vee \neg Hostile(z) \vee Criminal(x)$$
$$\neg Missile(x) \vee \neg Owns(Nono, x) \vee Sells(West, x, Nono)\ .$$
$$\neg Enemy(x, America) \vee Hostile(x)\ .$$
$$\neg Missile(x) \vee Weapon(x)\ .$$
$$Owns(Nono, M_1)\ . \qquad Missile(M_1)\ .$$
$$American(West)\ . \qquad Enemy(Nono, America)\ .$$

Resolution proof is -



Spine beginning with a goal clause resolves against clauses from the knowledge base until empty clause is generated. Clauses along main spine correspond exactly to consecutive values of goals variable in backward chaining algorithm.

Completeness of resolution –

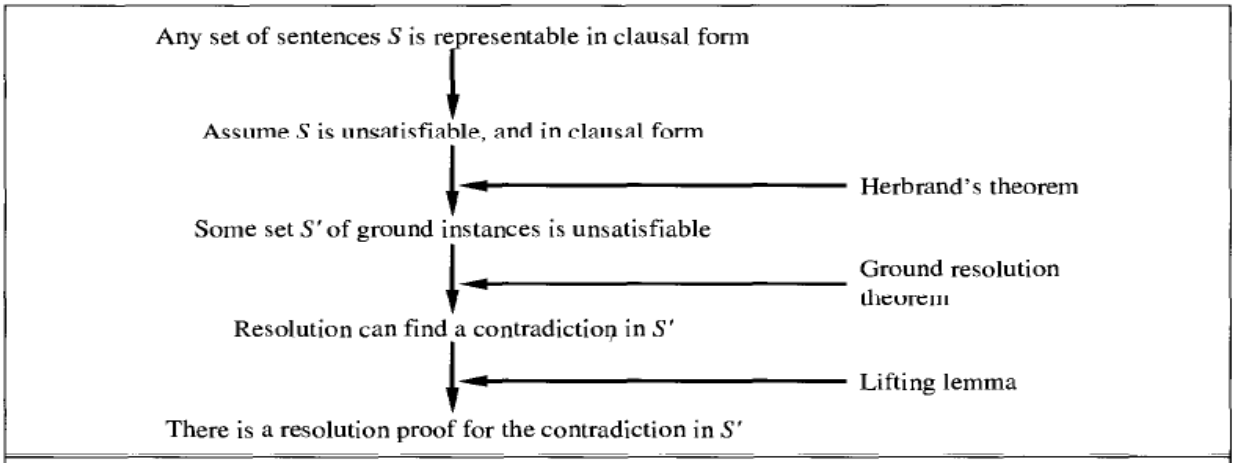
Resolution is refutation complete which means that if a set of sentences is unsatisfiable, then resolution will always be able to derive a contradiction. Resolution cannot be used to generate all logical sequences of a set of sentences but it can be used to establish that a given sentence is entailed by a set of sentences. Hence it can be used to find all answers to a given question using the negated goal method.

Given that any sentence in first order logic (without equality) can be rewritten as a set of clauses in CNF. If S is an unsatisfiable set of clauses, then the application of a finite number of resolution steps to S will yield a contradiction.

The basic structure of proof is as follows –

a) First, observe that if S is unsatisfiable, then there exists a particular set of ground instances of clauses of S such that this set is also unsatisfiable.
b) Use ground resolution theorem which states that propositional resolution is complete for ground sentences.
c) Use a lifting lemma to show that for any propositional resolution proof using the set of ground sentences, there is a corresponding first order resolution proof using the first order sentences from which the ground sentences were obtained.

Structure of completeness proof for resolution -



To carry out the first step, we need 3 new concepts:

a) Herbrand universe – If S is a set of clauses, then H_s, the herbrand universe of S is the set of all ground terms constructible from the following –
    1. the function symbols in S if any
    2. the constant symbol in S, if any, if none, then the constant symbol A.
b) Saturation – If S is a set of clauses, and P is a set of ground terms, then P(S), the saturation of S with respect to P is the set of all ground clauses obtained by applying all possible consistent substitutions of ground terms in P with variables in S.
c) Herbrand base – The saturation of a set S of clauses with respect to its Herbrand universe is called the Herbrand base of S, written as $H_s(S)$.

Resolution strategies –

Repeated applications of the resolution inference rule will find a proof if one exists. Strategies that help find proofs efficiently are –

a) Unit proof – This strategy prefers to do resolutions where one of the sentences if a single literal (also known as unit clause). The idea behind the strategy is that we are trying to produce an empty clause. Resolving a unit sentence with any other sentence always yields a clause that is shorter than the other clause. Unit preference by itself does not reduce the branching factor in medium sized problems enough to make them solvable by resolution. Unit resolution is a restricted form of resolution in which every resolution step must involve a unit clause.

b) Set of support – This strategy starts by identifying a subset of sentences called the set of support. Every resolution combines a sentence from the set of support with another sentence and adds the resolvent into the set of support. If the set of support is small relative to the whole knowledge base, the search space will be reduced drastically.

c) Input Resolution – In this strategy, every resolution combines one of the input sentences (from the KB or query) with some other sentence. In Horn Knowledge bases, Modus Ponens is a kind of input resolution strategy because it combines an implication from the original KB with some other sentences.

d) Subsumption – The subsumption method eliminates all sentences that are subsumed by an existing sentence in KB. Subsumption helps keep the KB small and thus helps keep the search space small.