Inference Rules in Propositional logic:-

- In propositional logic, inference rules are formal logic patterns that allows us to derive new propositions (conclusions) from known propositions (premises)

List of inference Rules.

1) Modus Ponens.
2) Modus Tollens
3) And Introduction
4) And Elimination
5) Or Introduction
6) Disjunctive Syllogism.
7) Hypothetical Syllogism.
8) Resolution.
9) Double Negation.

1. Modus Ponens:- (Implication Elimination)

Rule:

If

- $P \rightarrow Q$ (If P then Q)
- P (P is true)

Then.

Q (Q must be true).

Example:-

Premise 1: If it is raining, then the ground is wet. $R \rightarrow W$

Premise 2: It is raining. R.

Conclusion: The ground is wet. W.

- The Modus Ponens let us eliminate the implication & assert the truth of the consequence.

2. Modus Tollens

Rule:

If

$P \rightarrow Q$ (If P then Q)

$\neg Q$ (Q is not true.

Then

$\neg P$ (P is not true.

Example 1:

Premise 1:

If the computer is turned on, the power light will be on.

$C \rightarrow L$.

Premise 2: The light is not on.

$\neg L$

Conclusion:-

The computer is not turned on.

$\neg C$

- The failure of the expected effect is used to conclude the absence of the cause.

3. And Introduction ($\wedge$ Introduction)

If

P is true

Q is true

Then

$P \wedge Q$ is true.

Example:

Premise 1: R: It is raining. (true)
Premise 2: W: It is windy. (true)

Conclusion: It is Raining & Windy $(R \wedge W)$

If both statements are true, then combine them into one conjunctive statement using the logical AND $(\wedge)$.

4. And Elimination ($\wedge$ Elimination)

If
$P \wedge Q$ (if $P \wedge Q$ is true),
Then
P is true
OR
Q is true.

This rule extracts individual components from a conjunction.

Example:

Premise $B \wedge S$: The sky is blue and the sun is bright.

Conclusion: B: The sky is blue.
(You could also conclude
S: The sun is bright).

5. Or Introduction ($\vee$ Introduction)

Rule.

If

P (P is true)

Then

$P \vee Q$ (P or Q is true)

Example:

F: The fan is on.

Conclusion $F \vee A$. (The fan is on or the AC is on)

- Even if we don't know whether the AC is on, the truth of the fan being on is sufficient to conclude the disjunction.

This is because in a disjunction (OR statement) only one part needs to be true for the entire statement to be true.

6. Disjunctive Syllogism :-

If

$P \vee Q$ (P or Q) is true

$\neg P$ (P is not true

Then

Q must be true.

Example:

Premise 1: $S \vee M$: It is sunday or Monday

Premise 2: It is not sunday. ($\neg S$)

Conclusion (M): It is Monday.

If one option is false, the other must be true.

7. Hypothetical Syllogism (Transitivity of Implication)

Rule:

If

$P \rightarrow Q$ (If P then Q).

$Q \rightarrow R$ (If Q then R)

Then

$P \rightarrow R$ (If P then R)

This rule allows you to chain two conditional statements. (implications) together.

It means "If P leads to Q and Q leads to R, then P must lead to R.

Example:

Premise 1: $S \rightarrow P$

If I study, I will Pass.

Premise 2: $P \rightarrow G$

If I pass, I will graduate.

Conclusion: $S \rightarrow G$.

If I study, I will graduate.

8. Resolution:-

It is one of the powerful and fundamental inference rules in propositional logic.

Rule:

If

$A \vee B$ (A OR B)

$\neg B \vee C$ (NOT B OR C)

Then

$A \vee C$ (A OR C)

Resolution eliminates opposite literals (in this case (B and $\neg B$) to produce new clause.

Example:

Premise 1:

$R \vee C$: It is raining or cloudy.

Premise 2: $\neg C \vee K$

It is not cloudy or it is cold.

Conclusion: $R \vee K$

It is raining or it is cold.

9 Double Negation Elimination:

Rule

If $\neg\neg P$ (Not Not P)

Then

P.

This rule states that two negations cancel each other leaving you with the original proposition.

Example1:-

Premise: $\neg\neg H$

It is not the case that it is not hot.

Conclusion: Therefore it is hot. (H)

Example 2:

Premise: It is not true that it is not raining.

Conclusion: It is Raining.

Difference between Propositional logic and First order logic.

(1)

Propositional logic inference	First order logic inference
① The basic unit is a proposition like P or Q. represents a complete or indivisible fact.	① In first order logic the basic unit is a predicate that takes Variables. for example Human(x), means x is a human. and Loves(x, y) means x loves y.
e.g. Each proposition stands for a full statement like "It is raining" or The light is on." The logic doesn't know what the statement is about. The statements can be only true or false.	→ Here, you can describe the parts of the fact, like what kind of thing you are talking about (a human, a cat, a relationship).
② It does not allow reasoning about specific objects or thier properties.	② It can talk about real things, like people animals or objects.
It just deals with basic facts like P is true or Q is false. It doesn't know what P or Q mean.	It can say things like Whiskers is a cat or All cats are animals and then figure out new facts like "Whiskers is an animal.
③ It doesn't support quantifiers like for all or there exists.	③ It supports quantifiers such as ∀ (for all) & ∃ (there exists) to make general & existential claims.

Propositional logic Inference	First order logic Inference
(04) It has low expressive power and cannot represent general rules or relationships.	(04) It is highly expressive and can represent complex rules and relationships involving multiple objects.

It can only represent specific facts, like It is raining (P) or The light is on (Q). It can not talk about patterns or general rules. like All cats are animals. Everyone loves someone.

It allows you write general rules using variables & quantifiers.
e.g. For every x, if x is a cat, then x is an animal

$$\forall x\,(\mathrm{Cat}(x) \longrightarrow \mathrm{Animal}(x))$$

It also handles relationships between objects like Loves(John, Mary) means John loves mary.

Propositional logic Inference	First order logic Inference
(05) Inference is done using truth tables, basic rules like modus ponens. Modus Ponens Rule → 1. If P, then Q. 2. P is true then Q must also be true.	(05) Inference involves advanced techniques like unification, substitution and handling quantifiers.
(06) Sentences are atomic meaning they donot have internal structure.	(06) Sentences are structured, including terms, functions and variables, which provide more detail & flexibility.

Atomic sentence means the sentence can not be broken down any further.

Propositional Logic Inference

07 A typical sentence might be $P \rightarrow Q$ meaning if P then Q.

08 From $P \rightarrow Q$ and P, We can infer Q.

First order logic Inference

07 A typical sentence might be $\forall x (Cat(x) \rightarrow Animal(x))$ meaning "All cats are animals."

08 From Cat(Whiskers) & $\forall x (Cat(x) \rightarrow Animal(x))$ We can infer Animal(Whiskers) i.e. Whiskers is an Animal.

Note:- For Theory exam, You can write the contents which are given in rectangular boxes only.

* Unification & Lifting :-

- Until the 1960's computers tried to convert First Order logic into propositional logic (called propositionalization).
- But this method was slow & inefficient.
- Humans can easily figure out the things like.

$\boxed{\text{"John is a king" + "John is greedy} \rightarrow \text{" John is evil"}}$

computers should also do it easily without unnecessary steps.

e.g.

Imagine we have following facts.

$$\forall x \; (King(x) \wedge Greedy(x)) \longrightarrow Evil(x)$$

(for every x, if x is a king & x is greedy then x is evil)
that means. All geedy kings are evil.

Suppose,
King(John) $\longrightarrow$ John is a king.

Greedy(John) $\longrightarrow$ John is greedy.

We want to prove Evil(John). $\longrightarrow$ John is evil.

When we use propositionalization, the computer blindly creates all possible specific cases.

Imagine we have a general rule:

$$\forall x \; (King(x) \wedge Greedy(x) \longrightarrow Evil(x))$$

(For every x, if x is a king & greedy then x is ~~evil~~ evil.)

Now suppose, the computer knows about many people like Richard, Henry, John, Alice.

Scanned with OKEN Scanner

then it will automatically create statements like,

$King(Richard) \wedge Greedy(Richard) \rightarrow Evil(Richard)$

$King(Henry) \wedge Greedy(Henry) \rightarrow Evil(Henry)$

⋮

$King(Alice) \wedge Greedy(Alice) \rightarrow Evil(Alice)$

Even if we are only interested in John. The computer will create above statements. so It wastes time & memory by generating unnecessary statements for Richard, Henry, Alice etc.

It makes system slow & messy.

So, to solve this, we can find a substitution to match the rule directly with our known facts.

e.g.

Rule: $King(x) \wedge Greedy(x) \rightarrow Evil(x)$

Facts: $King(John)$, $Greedy(John)$.

So, we can substitute $x = John$.

Now Rule becomes,

$King(John) \wedge Greedy(John) \rightarrow Evil(John)$

Since, $King(John)$ & $Greedy(John)$ are true.

We can directly infer $Evil(John)$.

Also,

if, more general statement is given. like.

instead of $Greedy(John)$, we have given

$\forall y \; Greedy(y)$. $\longrightarrow$ Everyone is greedy

$King(John)$ $\longrightarrow$ John is a king.

still we can conclude Evil(John) because:

John is a King (given)

John is greedy (because everyone is greedy).

we substitute:

x = John (for King(x))

y = John (for greedy(y)).

Now Everything matches. & we can infer Evil(John).

* Regular Modus Ponens Rule:-

In propositional logic, Modus Ponens rule is,

If p is true and $p \rightarrow q$ is true then q must be true.

Example:-

p : "It is raining."

$p \rightarrow q$: "If it is raining then the ground is wet."

Therefore q : "The ground is wet.

This is very simple because in propositional logic we don't have variables.

So, in First order logic, facts and rules involves variables.

Example:

Fact 1: King(John) (John is king)

Fact 2: Greedy(John) (John is Greedy)

Rule: (King(x) ∧ Greedy(y)) ⇒ Evil(x).

So in FOL rule variables are used so we can't directly apply modus Ponens because the rule is general.

Hence we need to use Generalized Modus Ponens.

Generalized Modus Ponens :-

→ Generalized Modus Ponens is powerful version of modus Ponenus.

→ It is designed for F.O.L. where variables are used.

In Generalized Modus Ponens,

facts are represented as.

$p1', p2', p3' \ldots pn'$

- These are known atomic facts)

Rule is represented as:

$(p1 \wedge p2 \wedge \ldots \wedge pn) \rightarrow q.$

and if there is a substitution θ such that, After applying θ to each pi(rule), you match each known fact Pi'! Then you can conclude SUBST(θ, q) — apply same substitution to conclusi q.

... Evil(John) ...

Scanned with OKEN Scanner

Reasoning in Artificial Intelligence

Example:- step 1 (Find Facts & Rule)

Fact P_1' : king (John)

Fact P_2' : Greedy (John)

Rule: $(\text{King}(x) \wedge \text{Greedy}(y)) \rightarrow \text{Evil}(x)$.

$P_1 \wedge P_2 \rightarrow q$.

$P_2 \rightarrow$ given.

Step 2: Find substitution θ.

Find a θ that makes the premises match the facts.

Premises means starting statement or assumption that has variables.

Here we need P_1 and P_2 to match P_1' & P_2' after substitution.

P_1 is king(x), P_1' is King(John).

so. θ : $x \rightarrow$ John or $\{x / \text{John}\}$.

P_2 is Greedy(y), P_2' is Greedy (John)

so θ : $y \rightarrow$ John. or $\{y / \text{John}\}$.

Thus, the substitution θ is

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

Step3: Apply substitution θ to the conclusion.

conclusion q is Evil(x)

Applying θ gives.

$$\boxed{\text{SUBST}(\theta, q) \Rightarrow \text{Evil}(\text{John}).}$$

* Forward Chaining:-

- In forward chaining, we start with known facts and use rules to derive new facts until no more facts can be derived.
- It uses Modus Ponens, for example,

If $A \rightarrow B$ and
A is true.
then
B must also be true.

This process continues until:
- The goal is reached or
- No new facts can be inferred.

* Forward Chaining with First order Definite clause:-

- The first order definite clauses are logical rules or facts.

Each rule can be written as
If condition1 AND condition2 AND ...... ⇒ conclusion.

e.g -
$King(x) \wedge Greedy(x) \rightarrow Evil(x)$.

These clauses can have variables like (x, y) which are considered universally quantified (i.e. true for all values of x/y that make the rule apply).

All rules must have only one conclusion on the right side. (this is what makes them definite clause)

Let's take an example - The Missile Example.

The law says, It is a crime for an american to sell weapons to hostile countries.
↳ unfriendly

Nono is a hostile country and has missiles.
All missiles were sold to Nono by Colonel West, who is American.

Goal: Prove that West is a Criminal.

$\Rightarrow$

Law (Main Rule)

$American(x) \wedge weapon(y) \wedge sells(x, y, z) \wedge Hostile(z) \rightarrow criminal(x)$.

If x is american, and x sells a weapon y to hostile country z then x is a criminal.

We will prove that West is a criminal.

First, we need to represent these facts into first order definite clauses.

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

$American(x) \wedge weapon(y) \wedge sells(x, y, z) \wedge Hostile(z) \rightarrow Crimininal(x)$. — ①

Nono has some missiles.

$\exists x$ owns(Nono, x) $\wedge$ Missile(x). is transformed into two definite clauses by Existential Instantiation introducing a new constant M1.

Owns(Nono, M1) — ②

Missile(M1). — ③

All of it's missiles were sold to it by colonel West

Missile(x) $\wedge$ owns(Nono, x) $\Rightarrow$ ~~West~~ Sell (West, x, Nono) — ④

We will also need to know that missiles are weapons

Missile(x) $\rightarrow$ Weapon(x) — ⑤

and We must know that an enemy of America counted as hostile.

Enemy (x, America) → hostile(x). —— ⑥

West, who is an American.

American (West). —— ⑦

The country nono, an enemy of America.

Enemy (Nono, America) —— ⑧

This knowledge base contains no function symbol hence it is called as Datalog. Knowledge bases.

80

From ⑧ we can infer that

Enemy (Nono, America) → Hostile (Nono). — ⑨

From ② & ③

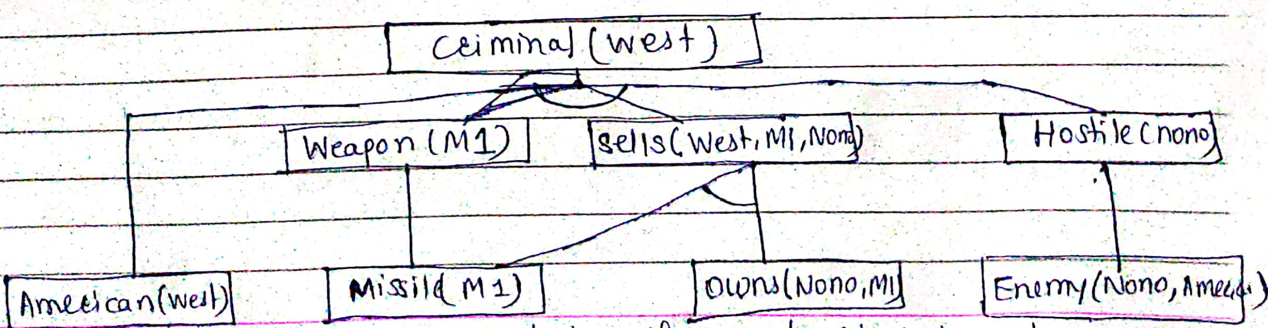
Missile (M1) ∧ owns (Nono, M1) → sells(West, M1, Nono) — ⑩

From ⑧

Missile (M1) → Weapon (M1) — ⑪

From ⑦, ⑪, ⑩, ⑨ we can infer that,

$\boxed{\text{Criminal (West).}}$



* Proof Generated by forward Chaining *

- The initial facts appear at the bottom level, facts inferred on the first iteration in the middle level and facts inferred on the second iteration at the top level.

Advantages:-

1. Forward chaining works incrementally as new information arrives.
2. It is useful for expert systems, rule based systems.
3. It is easy to implement.
4. It is sound & complete.

Sound: It derives only entailed (logically valid) conclusions.

Complete: For definite clause knowledge bases it can derive anything that is entailed.

Disadvantages:-

1. It is inefficient because forward chaining algorithm is data driven not goal driven.
   It may infer many irrelevant facts before reaching the desired conclusion.

2. It can generate useless facts.

* Backward chaining :-

- Backward chaining algorithm is a recursive depth first search algorithm that works by starting from the goal and looking backward to find rules that could produce the goal.

- It then tries to prove the premises of those rules recursively, continuing until it reaches known facts.

example:-

Prove West is a Criminal

Criminal (West)

Given knowledge base.

(I) Rules (Definite clauses):

1. Law.

$American(x) \wedge weapon(y) \wedge sells(x, y, z) \wedge Hostile(z) \rightarrow Criminal(x)$

2. Missile ownership & sellers.

$Missile(x) \wedge owns(Nono, x) \longrightarrow sells(West, x, Nono)$

3. Missiles are Weapons.

$Missiles(x) \rightarrow Weapon(x)$

4. Enemies are hostile.

$Enemy(x, America) \rightarrow hostile(x)$

(II) Facts :-

5. American (West)
6. Enemy (Nono, America)
7. Owns (Nono, M1)
8. Missile (M1)

* Step by step backward chaining

Prove: Criminal(West)

1st Rule:

American(x) ∧ weapon(y) ∧ sells(x, y, z) ∧ Hostile(z)
⟹ criminal(x)

To satisfy criminal(West), we must prove all of these:

American(West)
Weapon(y)
sells(West, y, z)
Hostile(z)

Let's initialize x = west & try to find values y & z

Step 1: Prove American(West)
- given in fact (statement / fact 5)
Hence proven.

Step 2:
We look for a rule that concludes sells(West, x, Nono)

Rule 2 matches with conclusion.

Missile(x) ∧ owns(Nono, x) → sells(West, x, Nono)

If we can find some x such that
Missile(x)
owns(Nono, x)
Then we can conclude sells(West, x, Nono)
from fact 8 & 7
Missile(M1) → fact 8
owns(Nono, M1) → fact 7

so we can conclude

Sells (West , M1 , Nono) (Proven)

Now,

$x = West$

$y = M1$

$z = Nono$.

Step3: Prove Weapon M1.

we know the rule.

Missile(x) $\rightarrow$ Weapon (x)

$x = M1$.

Weapon (M1) ——— Proved.

Step4: Prove Hostile (Nono)

Rule:

Enemy (X, America) $\rightarrow$ Hostile(x)

We have

Enemy (Nono, America) fact 6.

$\rightarrow$ hostile (Nono).

$\therefore$

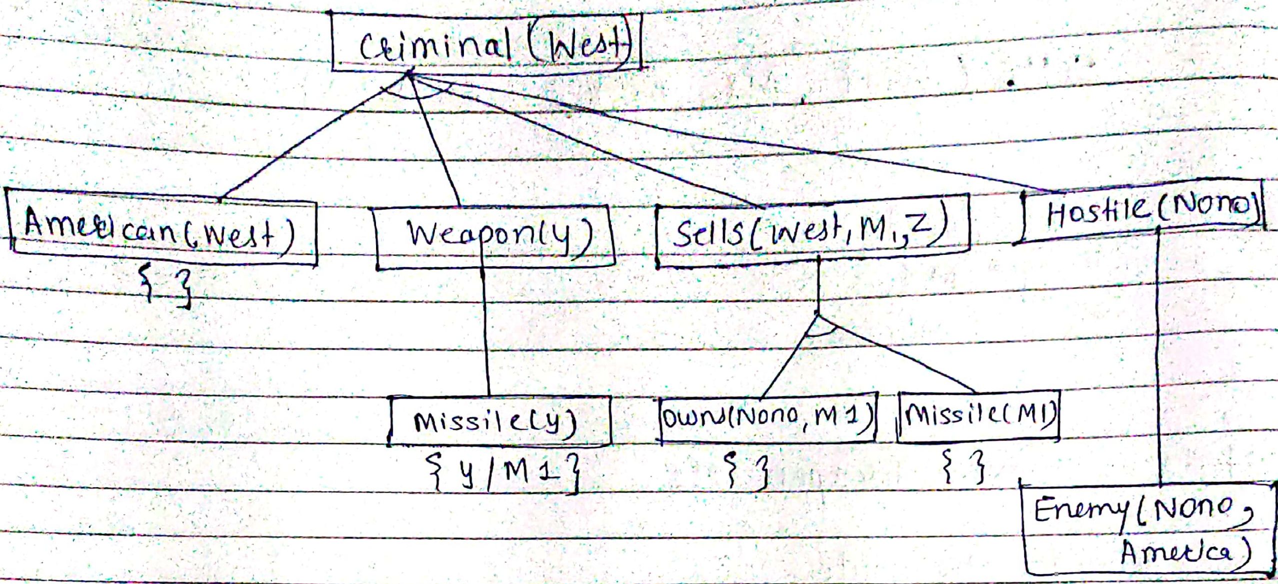
Proved

We have successfully proven all the conditions for

American(West) $\wedge$ Weapon(M1) $\wedge$ Sells(West, M1, Nono) $\wedge$ Hostile (Nono) $\rightarrow$ Criminal (West)

Therefore we conclude.

Criminal (West)



* Proof tree *

- Proof tree constructed by backward chaining to prove that West is a criminal.
- The tree should be read depth first, left to right
- To prove Criminal (West), we have to prove for conjucts below it. Some of these are in the knowledge base and other requires further backward chaining.

Advantages:-

1. It focuses only on facts relevant to the specific query.
2. It is goal directed, it avoids exploring irrelevant rules or data.
3. It can work efficiently with large KB's.
4. It requires few inferences as it only derives facts needed to answer the goal.

Disadvantages:-

1. It may loop infinetly.
2. It can mis useful facts as it ignores facts that are not directly connected to goal.
3. It requires complete goal specification.