Inference

- Deriving conclusions from evidences
- Evidences are assumptions and on basis of assumptions we are trying to get some conclusions.
- Rules of inference are the templates for constructing valid arguments

Inference Rules: Deducing logical outcomes

If it rains, I will get wet.

Its raining

∴ I will get wet

$$\left.\begin{array}{l} R \rightarrow W \\ R \end{array}\right\} \text{ Premises}$$

$$\therefore W \text{ - Conclusion}$$

- Rules of inference is taking a set of premises and getting to a conclusion.

$$((R \rightarrow W) \land R) \rightarrow W$$

With truth table this would be a tautology. So all rules of inference of this form are going to form tautology

Inference

- A set of premises P_1, P_2, $P_{3,}$.....,P_n prove some conclusion Q in an argument.

$$(P_1 \land P_2 \land P_3 \land \ldots\ldots \land P_n) \rightarrow Q$$

- An argument is valid if the premises logically involve the conclusion.
- All inference rules take us from the premises to the conclusion.

Resolution

- Its an inference rule used in both Propositional as well as First-order Predicate Logic in different ways.
- It is also called **Proof by Refutation**
- It is basically used for proving the satisfiability of a sentence.
- Proof by Refutation technique is used to prove the given statement. It's a iterative process :
  1. Select two clauses that contain conflicting terms.
  2. Combine those two clauses and cancel out the conflicting terms, yielding a new clause that has been inferred from them.
- The key idea is to use the knowledge base and negated goal to obtain null clause(which indicates contradiction).
- Since the knowledge base itself is consistent, the contradiction must be introduced by a negated goal. As a result, we have to conclude that the original goal is true.

Normal forms

- To solve a complex problem which is represented using set of well formed formulas(wff) we need to reduce wff into set of clauses.
- It's a technique for representing a complex sentence into simple sentences so that resolution can be applied

Steps for Converting a Sentence into Clauses for reduction to normal form

Step- I

- Elimination of biconditional connective : $\leftrightarrow$

$$a \leftrightarrow b \equiv (a \rightarrow b) \land (b \rightarrow a)$$

Step-2 (Replacing 'if-then' operator by negation and OR operator)

- Elimination of Implication connective: $\rightarrow$

$$a \rightarrow b \equiv \neg a \lor b \qquad (\neg a \lor b) \land (\neg b \lor a)$$

Step-3: Move ¬ inward

- Reduce scope of each ¬ to single term or literal.
- Move all negations inward such that, in the end, negations only appear as part of literals .

$\neg(a \land b) \equiv \neg a \lor \neg b$

$\neg(a \lor b) \equiv \neg a \land \neg b$

$\neg(\forall x\ a) \equiv \exists x\ \neg a$

$\neg(\exists x\ a) \equiv \forall x\ \neg a$

$\neg(\land) \equiv \lor$

$\neg(\lor) \equiv \land$

$\neg\neg a \equiv a$

Step-4

- Standardize variables so that each quantifier binds a unique variable
- Renaming the variable within the scope of quantifiers

$\forall x: P(x) \lor \forall x: Q(x)$

$\forall x: P(x) \lor \forall y: Q(y)$

Step-5

- Moving of quantifiers in the front of the expression
- Moving all quantifiers to the front of the formula without changing their order.

$\forall x: P(x) \lor \forall y: Q(y)$

$\forall x \, \forall y: P(x) \lor Q(y)$

This is also known as prenex normal form.

Settings

Step-6 (Skolemization)

- Elimination of existential quantifier ($\exists$)
- Replacing existential quantifier as Skolem constant or Skolem function

  $\exists x: Rich(x)$
- Replacing x with skolem constant

  Rich(Ansh)

  Here we replaced x with actual name of the person
- Replacing **y** with skolem function

  $\forall x: \exists y \, P(x,y) \equiv \forall x: P(x,f(x))$

  Here value of **y** that satisfies '**P**' depends on particular value of **x**

Settings

Step-7

- Drop Universal Quantifier (∀)

  $\forall x : Person(x) \equiv Person(x)$

Step-8 (Conjunction of Disjunction)

- **Apply distributive law**

$$P \vee (Q \wedge R) \equiv \underbrace{(P \vee Q) \wedge (P \vee R)}_{\text{Conjunctive form}}$$

$2(x + y) \quad 2x + 2y$

Example #1

- $\forall x\, P(x) \rightarrow \exists x\, Q(x)$

  (Eliminate implication by Replacing → connective by negation and OR operator)

  $\neg \forall x\ P(x) \vee \exists x\, Q(x)$

  (Move negation inward ($\neg(\forall x\, a) \equiv \exists x\, \neg a$)

  $\exists x \neg P(x) \vee \exists x\, Q(x)$

  (using reverse distributive law take ∃ out)

  $\exists x (\neg P(x) \vee Q(x))$

  Above statement is in prenex normal form.

Example #2

- $\forall x\ \forall y\ [\exists z\ (\ P(x,z) \land P(y,z)) \rightarrow \exists u\ Q(x,y,u)]$

$\forall x\ \forall y\ [\exists z\ (P(x,z) \land P(y,z)) \rightarrow \exists u\ Q(x,y,u)]$

- Elimination of implication connective:

$\forall x\ \forall y\ [\neg \exists z\ (P(x,z) \land P(y,z)) \lor \exists u\ Q(x,y,u)]$

- Moving ¬ inward:

$\forall x\ \forall y\ [\forall z\ \{\neg(P(x,z) \land P(y,z))\} \lor \exists u\ Q(x,y,u)]$

- Using De morgans law

$\forall x\ \forall y\ [\forall z\ (\neg P(x,z) \lor \neg P(y,z)) \lor \exists u\ Q(x,y,u)]$

$\forall x\ \forall y\ [\forall z\ (\neg P(x,z) \lor \neg P(y,z)) \lor \exists u\ Q(x,y,u)]$
can be written as

$$\underbrace{\forall x\ \forall y\ \forall z\ \exists u}_{\text{Prefix}}\ [\ \underbrace{\neg P(x,z) \lor \neg P(y,z) \lor Q(x,y,u)}_{\text{Matrix}}]$$

- Prefix containing all quantifiers and matrix without any quantifiers.

Example

Conversion of Facts/Statements into FOL

1. Sunil likes all kind of food.
2. Apple and vegetable are food
3. Anything anyone eats and not killed is food.
4. Anil eats peanuts and still alive
5. Sohan eats everything that Anil eats
6. Sunil likes peanuts.

1. ∀x : food(x)→likes(Sunil, x)
2. food(apple) Λ food(vegetable)
3. ∀x ∀y : eats(x , y) Λ ¬ killed(x)→food(y)
4. eats(Anil , Peanut) Λ alive(Anil)
5. ∀x: eats(Anil, x) → eats(Sohan , x)
6. Likes(Sunil, Peanut)

- ∀x : ¬ killed(x)→alive(x)
- ∀x : alive (x)→¬killed (x)

(added predicates)

Conversion of FOL into CNF: Elimination of →

1. ∀x : food(x)→likes(Sunil, x)
2. food(apple) Λ food(vegetable)
3. ∀x ∀y : eats(x , y) Λ ¬ killed(x)→food(y)
4. eats(Anil , Peanut) Λ alive(Anil)
5. ∀x: eats(Anil, x) → eats(Sohan , x)
6. ∀x : ¬ killed(x)→alive(x)
7. ∀x : alive (x)→¬killed (x)
8. Likes(Sunil, Peanut)

1. ∀x ¬ food(x) V likes(Sunil, x)
2. food(Apple) Λ food(vegetables)
3. ∀x ∀y ¬ [eats(x, y) Λ ¬ killed(x)] V food(y)
4. eats (Anil, Peanuts) Λ alive(Anil)
5. ∀x ¬ eats(Anil, x) V eats(Sohan, x)
6. ∀x¬ [¬ killed(x)] V alive(x)
7. ∀x ¬ alive(x) V ¬ killed(x)
8. likes(Sunil, Peanuts)

Move negation (¬) inwards and rewrite

1. ∀x ¬ food(x) V likes(Sunil, x)
2. food(Apple) Λ food(vegetables)
3. ∀x ∀y ¬ [eats(x, y) Λ ¬ killed(x)] V food(y)
4. eats (Anil, Peanuts) Λ alive(Anil)
5. ∀x ¬ eats(Anil, x) V eats(Sohan, x)
6. ∀x¬ [¬ killed(x)] V alive(x)
7. ∀x ¬ alive(x) V ¬ killed(x)
8. likes(Sunil, Peanuts).

1. ∀x ¬ food(x) V likes(Sunil, x)
2. food(Apple) Λ food(vegetables)
3. ∀x ∀y ¬ eats(x, y) V killed(x) V food(y)
4. eats (Anil, Peanuts) Λ alive(Anil)
5. ∀x ¬ eats(Anil, x) V eats(Sohan, x)
6. ∀x killed(x) V alive(x)
7. ∀x ¬ alive(x) V ¬ killed(x)
8. likes(Sunil, Peanuts).

Rename variables or standardize variables

1. ∀x ¬ food(x) V likes(Sunil, x)
2. food(Apple) Λ food(vegetables)
3. ∀x ∀y ¬ eats(x, y) V killed(x) V food(y)
4. eats (Anil, Peanuts) Λ alive(Anil)
5. ∀x ¬ eats(Anil, x) V eats(Sohan, x)
6. ∀x killed(x) V alive(x)
7. ∀x ¬ alive(x) V ¬ killed(x)
8. likes(Sunil, Peanuts)

1. ∀x ¬ food(x) V likes(Sunil, x)
2. food(Apple) Λ food(vegetables)
3. ∀y ∀z ¬ eats(y, z) V killed(y) V food(z)
4. eats (Anil, Peanuts) Λ alive(Anil)
5. ∀w¬ eats(Anil, w) V eats(Sohan, w)
6. ∀g killed(g) V alive(g)
7. ∀k ¬ alive(k) V ¬ killed(k)
8. likes(Sunil, Peanuts)

Drop Universal quantifiers

1. ∀x ¬ food(x) V likes(Sunil, x)
2. food(Apple) Λ food(vegetables)
3. ∀y ∀z ¬ eats(y, z) V killed(y) V food(z)
4. eats (Anil, Peanuts) Λ alive(Anil)
5. ∀w¬ eats(Anil, w) V eats(Sohan, w)
6. ∀g killed(g) V alive(g)
7. ∀k ¬ alive(k) V ¬ killed(k)
8. likes(Sunil, Peanuts).

1. ¬ food(x) V likes(Sunil, x)
2. food(Apple)
3. food(vegetables)
4. ¬ eats(y, z) V killed(y) V food(z)
5. eats (Anil, Peanuts)
6. alive(Anil)
7. ¬ eats(Anil, w) V eats(Sohan, w)
8. killed(g) V alive(g)
9. ¬ alive(k) V ¬ killed(k)
10. likes(Sunil, Peanuts)

Statements "food(Apple) Λ food(vegetables)" and "eats (Anil, Peanuts) Λ alive(Anil)" can be written in two separate statements.

Negate the statement to be proved

- Prove by resolution that: **Sunil likes peanuts.**

  likes(Sunil, Peanuts).

  ¬likes(Sunil, Peanuts)

Resolution graph