First Order Logic

Pros and Cons of Propositional Logic

- Propositional logic is **declarative**: pieces of syntax correspond to facts
- Propositional logic allows partial/disjunctive/negated information (unlike most data structures and databases)
- Propositional logic is **compositional**: meaning of $B_{1,1} \wedge P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$
- Meaning in propositional logic is **context-independent** (unlike natural language, where meaning depends on context)
- Propositional logic has very limited expressive power (unlike natural language)

First-Order Logic

Whereas propositional logic assumes world contains **facts**,
first-order logic (like natural language) assumes the world contains

- Objects: people, houses, numbers, theories, Ronald McDonald, colors, baseball games, wars, centuries ...
- Relations: red, round, bogus, prime, multistoried ...,
  brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, ...
- Functions: father of, best friend, third inning of, one more than, end of ...

Syntax:

Constants	$KingJohn,\ 2,\ UCB, \ldots$
Predicates	$Brother,\ >, \ldots$
Functions	$Sqrt,\ LeftLegOf, \ldots$
Variables	$x,\ y,\ a,\ b, \ldots$
Connectives	$\wedge \quad \vee \quad \neg \quad \Rightarrow \quad \Leftrightarrow$
Equality	$=$
Quantifiers	$\forall \ \exists$

Truth in FOL

Sentences are true with respect to a model and an interpretation

Model contains ≥ 1 objects (domain elements) and relations among them

Interpretation specifies referents for

constant symbols $\rightarrow$ objects

predicate symbols $\rightarrow$ relations

function symbols $\rightarrow$ functional relations

An atomic sentence $predicate(term_1, \ldots, term_n)$ is true
iff the objects referred to by $term_1, \ldots, term_n$
are in the relation referred to by $predicate$

Atomic Sentences

$$\text{Atomic sentence} = predicate(term_1, \ldots, term_n)$$
$$\text{or } term_1 = term_2$$

E(x, y) denote "x = y"

$$\text{Term} = function(term_1, \ldots, term_n)$$
$$\text{or } constant \text{ or } variable$$

E.g., $Brother(KingJohn, RichardTheLionheart)$

$> (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))$

Multiple Quantifiers

∃x ∀ y p(x,y) means "There exists some x such that p (x, y) is true for every y."

Universal Quantification

$\forall \langle variables \rangle \ \ \langle sentence \rangle$

Everyone at Berkeley is smart:
$\forall\, x \ \ At(x, Berkeley) \ \Rightarrow \ Smart(x)$

$\forall\, x \ \ P$ is true in a model m iff P is true with x being **each** possible object in the model

Roughly speaking, equivalent to the conjunction of instantiations of P

$$
\begin{aligned}
& (At(KingJohn, Berkeley) \ \Rightarrow \ Smart(KingJohn)) \\
\wedge \ & (At(Richard, Berkeley) \ \Rightarrow \ Smart(Richard)) \\
\wedge \ & (At(Berkeley, Berkeley) \ \Rightarrow \ Smart(Berkeley)) \\
\wedge \ & \ldots
\end{aligned}
$$

Existential Quantification

$\exists \langle variables \rangle \;\; \langle sentence \rangle$

Someone at Stanford is smart:
$\exists x \;\; At(x, Stanford) \land Smart(x)$

$\exists x \;\; P$ is true in a model m iff P is true with x being **some** possible object in the model

Roughly speaking, equivalent to the disjunction of instantiations of P

$$
\begin{aligned}
& (At(KingJohn, Stanford) \land Smart(KingJohn)) \\
\lor\ & (At(Richard, Stanford) \land Smart(Richard)) \\
\lor\ & (At(Stanford, Stanford) \land Smart(Stanford)) \\
\lor\ & \ldots
\end{aligned}
$$

Negation of Quantified Propositions

Negation of a universally quantified proposition results an existentially quantified proposition, and negation of an existentially quantified proposition results a universally quantified proposition.

- $\forall x\ p(x) \land \exists\ y\ q(y)$
- $\sim.(\forall x\ p(x) \land \exists\ y\ q(y))$

  $\cong \sim\forall\ x\ p(x) \lor \sim\exists y q\ (y)$ $\qquad (\therefore \sim(p \land q) = \sim p \lor \sim q)$

  $\cong \exists\ x \sim p(x) \lor \forall y \sim q(y)$

Well Formed Formula (wff)

Complex sentences are made from atomic sentences using connectives

$$\neg S, \quad S_1 \wedge S_2, \quad S_1 \vee S_2, \quad S_1 \Rightarrow S_2, \quad S_1 \Leftrightarrow S_2$$

E.g. $Sibling(KingJohn, Richard) \Rightarrow Sibling(Richard, KingJohn)$

Well Formed Formula (wff) is a predicate holding any of the following −

- All propositional constants and propositional variables are wffs
- If x is a variable and Y is a wff, $\forall x$Y and $\exists x$Y are also wff
- Truth value and false values are wffs
- Each atomic formula is a wff
- All connectives connecting wffs are wffs

Truth Example

Consider the interpretation in which
$Richard \rightarrow$ Richard the Lionheart
$John \rightarrow$ the evil King John
$Brother \rightarrow$ the brotherhood relation

Under this interpretation, $Brother(Richard, John)$ is true just in case Richard the Lionheart and the evil King John are in the brotherhood relation in the model

Properties of Quantifiers

$\forall x \ \forall y$ is the same as $\forall y \ \forall x$

$\exists x \ \exists y$ is the same as $\exists y \ \exists x$

$\exists x \ \forall y$ is **not** the same as $\forall y \ \exists x$

$\exists x \ \forall y \ Loves(x, y)$
"There is a person who loves everyone in the world"

$\forall y \ \exists x \ Loves(x, y)$
"Everyone in the world is loved by at least one person"

Quantifier duality: each can be expressed using the other

$\forall x \ Likes(x, IceCream) \qquad \neg\exists x \ \neg Likes(x, IceCream)$

$\exists x \ Likes(x, Broccoli) \qquad \neg\forall x \ \neg Likes(x, Broccoli)$

Deducing Hidden Properties

Properties of locations:

$\forall x, t \;\; At(Agent, x, t) \land Smelt(t) \;\Rightarrow\; Smelly(x)$

$\forall x, t \;\; At(Agent, x, t) \land Breeze(t) \;\Rightarrow\; Breezy(x)$

Squares are breezy near a pit:

Diagnostic rule—infer cause from effect

$$\forall y \;\; Breezy(y) \;\Rightarrow\; \exists x \;\; Pit(x) \land Adjacent(x, y)$$

Causal rule—infer effect from cause

$$\forall x, y \;\; Pit(x) \land Adjacent(x, y) \;\Rightarrow\; Breezy(y)$$

Neither of these is complete—e.g., the causal rule doesn't say whether squares far away from pits can be breezy

Definition for the $Breezy$ predicate:

$$\forall y \;\; Breezy(y) \;\Leftrightarrow\; [\exists x \;\; Pit(x) \land Adjacent(x, y)]$$

Inferencing in Predicate Logic

Domain: D

Constant Symbols: M, N, O, P, ...

Variable Symbols: x,y,z,....

Function Symbols: F(x), G(x,y), H(x,y,z)

Predicate Symbols: p(x), q(x,y), r(x,y,z),

Connectors: ~, Λ, V, →, ∃, ∀

Terms:

Well-formed Formula:

Free and Bound Variables:

Interpretation, Valid, Non-Valid, Satisfiable, Unsatisfiable

What is an Interpretation? Assign a domain set D, map constants, functions, predicates suitably. The formula will now have a truth value

Example:

F1: ∀x(g(M, x) → g(L, x))

F2: g(M, S)

G: g(L, S)

Interpretation 1: D = {Akash, Baby, Home, Play, Ratan, Swim}, etc.,

Interpretation 2: D = Set of Integers, etc.,

How many interpretations can there be?

To prove Validity, means (F1 Λ F2) → G) is true under all interpretations

To prove Satisfiability means (F1 Λ F2) → G) is true under at least one interpretation

Resolution in FOL

- Resolution is a theorem proving technique that proceeds by building refutation proofs, i.e., proofs by contradictions.
- Resolution is used, if there are various statements are given, and we need to prove a conclusion of those statements.
- Resolution is a single inference rule which can efficiently operate on the **conjunctive normal form or clausal form**.
- **Clause**: Disjunction of literals (an atomic sentence) is called a **clause.**
- **Conjunctive Normal Form**: A sentence represented as a conjunction of clauses is said to be **conjunctive normal form** or **CNF**

Steps for Resolution:

1.Conversion of facts into first-order logic.
2. Convert FOL statements into CNF
3. Negate the statement which needs to prove (proof by contradiction)
4. Draw resolution graph (unification).

(i) John likes all kind of food.
(ii) Apple and vegetable are food
(iii) Anything anyone eats and not killed is food.
(iv) Anil eats peanuts and still alive
(v) Harry eats everything that Anil eats.

Prove by resolution:
John likes peanuts.

Step 1:

a. $\forall x: \text{food}(x) \rightarrow \text{likes}(\text{John}, x)$

b. $\text{food}(\text{Apple}) \land \text{food}(\text{vegetables})$

c. $\forall x \forall y: \text{eats}(x, y) \land \neg \text{killed}(x) \rightarrow \text{food}(y)$

d. $\text{eats}(\text{Anil}, \text{Peanuts}) \land \text{alive}(\text{Anil})$.

e. $\forall x: \text{eats}(\text{Anil}, x) \rightarrow \text{eats}(\text{Harry}, x)$

f. $\forall x: \neg \text{killed}(x) \rightarrow \text{alive}(x)$ **added predicates.**

g. $\forall x: \text{alive}(x) \rightarrow \neg \text{killed}(x)$

h. $\text{likes}(\text{John}, \text{Peanuts})$

Step-2: Conversion of FOL into CNF

In First order logic resolution, it is required to convert the FOL into CNF

Eliminate all implication (→) and rewrite

∀x ¬ food(x) V likes(John, x)
food(Apple) Λ food(vegetables)
∀x ∀y ¬ [eats(x, y) Λ ¬ killed(x)] V food(y)
eats (Anil, Peanuts) Λ alive(Anil)
∀x ¬ eats(Anil, x) V eats(Harry, x)
∀x¬ [¬ killed(x)] V alive(x)
∀x ¬ alive(x) V ¬ killed(x)
likes(John, Peanuts).

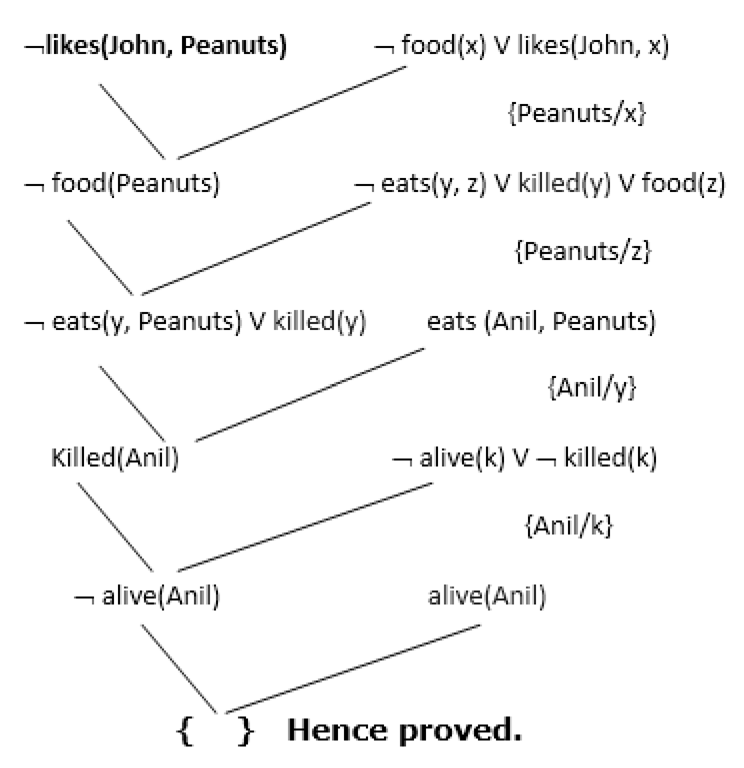
Move negation (¬)inwards and rewrite

∀x ¬ food(x) V likes(John, x)
food(Apple) Λ food(vegetables)
∀x ∀y ¬ eats(x, y) V killed(x) V food(y)
eats (Anil, Peanuts) Λ alive(Anil)
∀x ¬ eats(Anil, x) V eats(Harry, x)
∀x ¬killed(x)] V alive(x)
∀x ¬ alive(x) V ¬ killed(x)
likes(John, Peanuts).

- **Rename variables or standardize variables**
  - ∀x ¬ food(x) V likes(John, x)
  - food(Apple) Λ food(vegetables)
  - ∀y ∀z ¬ eats(y, z) V killed(y) V food(z)
  - eats (Anil, Peanuts) Λ alive(Anil)
  - ∀w¬ eats(Anil, w) V eats(Harry, w)
  - ∀g ¬killed(g) V alive(g)
  - ∀k ¬ alive(k) V ¬ killed(k)
  - likes(John, Peanuts).
- **Eliminate existential instantiation quantifier by elimination.** In this step, we will eliminate existential quantifier ∃, and this process is known as **Skolemization**.
- Here there is no existential quantifier so all the statements will remain same in this step.

- **Drop Universal quantifiers.**
  In this step we will drop all universal quantifier since all the statements are not implicitly quantified so we don't need it.
  - ¬ food(x) V likes(John, x)
  - food(Apple)
  - food(vegetables)
  - ¬ eats(y, z) V killed(y) V food(z)
  - eats (Anil, Peanuts)
  - alive(Anil)
  - ¬ eats(Anil, w) V eats(Harry, w)
  - killed(g) V alive(g)
  - ¬ alive(k) V ¬ killed(k)
  - likes(John, Peanuts).

- **Distribute conjunction ∧ over disjunction ¬.**
  Step-3: Negate the statement to be proved
- In this statement, we will apply negation to the conclusion statements, which will be written as ¬likes(John, Peanuts)
- **Step-4: Draw Resolution graph:**



Resolution Refutation for Propositional Logic

To prove validity of

$F = ((F1 \land F2 \land \ldots \land Fn) \rightarrow G)$

we shall attempt to prove that

$\sim F = (F1 \land F2 \land \ldots \land Fn \land \sim G)$

is unsatisfiable

Steps for Proof by Resolution Refutation:

1. Convert of Clausal Form / Conjunctive Normal Form (CNF, Product of Sums).
2. Generate new clauses using the resolution rule.
3. At the end, either False will be derived if the formula ~F is unsatisfiable implying F is valid.

If Asha is elected VP then Rajat is chosen as G-Sec and Bharati is chosen as Treasurer. Rajat is not chosen as G-Sec. Therefore Asha is not elected VP.

F1: $(a \rightarrow (b \land c)) = (\sim a \lor b) \land (\sim a \lor c)$

F2: $\sim b$, G: $\sim a$, $\sim G$: a

Clauses of Clause Form: $\sim F = (C1 \land C2 \land C3 \land C4)$

where: C1: $(\sim a \lor b)$

C2: $(\sim a \lor c)$

C3: $\sim b$

C4: a

To prove that ~F is False

Let $C1 = a \lor b$ and $C2 = \sim a \lor c$

then a new clause $C3 = b \lor c$ can be derived.

(Proof by showing that $((C1 \land C2) \rightarrow C3)$ is a valid formula).

To prove unsatisfiability use the Resolution Rule repeatedly to reach a situation where we have two contradictory clauses of the form $C1 = a$ and $C2 = \sim a$ from which False can be derived.

If the propositional formula is satisfiable then we will not reach a contradiction and eventually no new clauses will be derivable.

For propositional logic the procedure terminates.

Resolution Rule is Sound and Complete

Applying Resolution Refutation

Let C1 = a V b and C2 = ~a V c then a new clause C3 = b V c can be derived.

(Proof by showing that ((C1 ∧ C2) → C3) is a valid formula).

To prove unsatisfiability use the Resolution Rule repeatedly to reach a situation where we have two contradictory clauses of the form C1 = a and C2 = ~a from which False can be derived.

If the propositional formula is satisfiable then we will not reach a contradiction and eventually no new clauses will be derivable.

For propositional logic the procedure terminates.

Resolution Rule is Sound and Complete

If Asha is elected VP then Rajat is chosen as G-Sec and Bharati is chosen as Treasurer. Rajat is not chosen as G-Sec. Therefore Asha is not elected VP.

F1: (a → (b ∧ c)) = (~a V b) ∧ (~a V c)

F2: ~b

G: ~a

~G: a

Clauses of Clause Form: ~F
= (C1 ∧ C2 ∧ C3 ∧ C4)
where: C1: (~a V b)
C2: (~a V c)
C3: ~b
C4: a
To prove that ~F is False

New Clauses Derived
C5: ~a (Using C1 and C3)
C6: False (using C4 and C5)

Resolution Refutation for Predicate Logic

Given a formula F which we wish to check for validity, we first check if there are any free variables. We then quantify all free variables universally.

Create F' = ~F and check for unsatisfiability of F'

STEPS:

Conversion to Clausal (CNF) Form:

- Handling of Variables and Quantifiers, Ground Instances

Applying the Resolution Rule:

- Concept of Unification
- Principle of Most General Unifier (mgu)
- Repeated application of Resolution Rule using mgu

CONVERSION TO CLAUSAL FORM IN PREDICATE LOGIC

1. Remove implications and other Boolean symbols converting to equivalent forms using ~, V, Λ
2. Move negates (~) inwards as close as possible
3. Standardize (Rename) variables to make them unambiguous
4. Remove Existential Quantifiers by an appropriate new function /constant symbol taking into account the variables dependent on the quantifier (Skolemization)
5. Drop Universal Quantifiers
6. Distribute V over Λ and convert to CNF