Knowledge & Reasoning

Knowledge-Based Agents · Logic · FOL · Syntax-Semantics · Inference · Resolution

Dr. Gopal Chandra Jana

Dept. of CSE, Sharda University

https://www.gcjana.in/courses/shardauniversity/2501/CSE472/

April 30, 2026

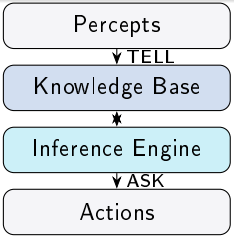
Course Outline

1. Knowledge-Based Agents
2. Propositional Logic
3. First-Order Logic (FOL)
4. Syntax in FOL
5. Semantics in FOL
6. Simple Usage of FOL
7. Inference Procedures
8. Inference in FOL
9. Reduction to Propositional Logic
10. Inference Rules
11. Forward Chaining
12. Backward Chaining
13. Resolution

What is a Knowledge-Based Agent?

Definition

A **Knowledge-Based Agent (KBA)** is an AI agent with an internal **Knowledge Base (KB)** of sentences about the world and an **inference engine** that derives conclusions to guide actions.



Two core components:

- **Knowledge Base (KB)**: set of *sentences* (facts + rules) in a formal language
- **Inference Engine**: derives new sentences from KB

Agent operations:

- `TELL(KB, sentence)`: add new knowledge

KBA — Generic Algorithm

KB-Agent Pseudocode

function KB-AGENT(*percept*):
  persistent: *KB* (knowledge base), t (time counter, init 0)
  TELL(*KB*, MAKE-PERCEPT-SENTENCE(*percept*, t))
  action $\leftarrow$ ASK(*KB*, MAKE-ACTION-QUERY(t))
  TELL(*KB*, MAKE-ACTION-SENTENCE(*action*, t))
  $t \leftarrow t + 1$
  return *action*

Declarative approach:

- Build KB by telling it facts and rules
- Query KB to determine what action to take
- Knowledge is explicit and inspectable
- Can update KB as world changes

Wumpus World Example:

- Agent perceives: Breeze, Stench, Glitter
- TELL: "There is a breeze at [1,2]"
- KB derives: "Pit is adjacent to [1,2]"
- ASK: "Which square is safe to move to?"

Newell's Knowledge Level

An agent is described at the knowledge level by *what it knows*, independent of how knowledge is stored.

Level	Description	Example
Knowledge Level	Goals and facts (semantic)	"It rains in Mumbai in July"
Symbol Level	Logical sentences (syntactic)	$Rain(Mumbai, July)$
Implementation	Data structures	Hash tables, trees

Ontological Commitment

What kinds of things exist?

- Objects (people, places)
- Properties (color, weight)
- Relations (parent, friend)
- Events, time, beliefs

Epistemological Commitment

What an agent can believe:

- True / False (classical logic)
- Degrees of belief (probability)
- Fuzzy truth values

Why Logic?

Purpose of Logic in AI

Logic provides a **formal language** with precise **syntax** (what sentences look like) and **semantics** (what sentences mean) for representing knowledge and reasoning.

Properties of a good logic:

- **Soundness**: only derive true conclusions
- **Completeness**: derive ALL true conclusions
- **Expressiveness**: represent real-world knowledge
- **Efficiency**: tractable computation

Types of logic in AI:

- Propositional Logic (PL)
- First-Order Logic (FOL)
- Temporal Logic, Fuzzy Logic
- Description Logic, Modal Logic

This Lecture Focuses On

Propositional Logic (PL) and First-Order Logic (FOL) — the foundations of classical AI reasoning.

Logic vs. English

"All cats have tails" $\rightarrow$
$\forall x\ Cat(x) \Rightarrow HasTail(x)$

English is ambiguous; logic is precise.

Propositional Logic (PL)

Uses **propositional symbols** ($P, Q, R, \ldots$) that are either True or False, combined with **logical connectives**.

Symbol	Name	Meaning
$\neg P$	Negation	"not P" (complement)
$P \wedge Q$	Conjunction	"P and Q" (both true)
$P \vee Q$	Disjunction	"P or Q" (at least one)
$P \Rightarrow Q$	Implication	"if P then Q"
$P \Leftrightarrow Q$	Biconditional	"P if and only if Q"

Example

$P =$ "It is raining", $Q =$ "Ground is wet"
$P \Rightarrow Q$: "If it is raining, then ground is wet."
$\neg P$: "It is not raining."
$P \wedge Q$: "It is raining AND ground is wet."

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

Key Equivalences

- $P \Rightarrow Q \equiv \neg P \vee Q$
- $P \Leftrightarrow Q \equiv (P \Rightarrow Q) \wedge (Q \Rightarrow P)$
- *De Morgan*: $\neg(P \wedge Q) \equiv \neg P \vee \neg Q$
- *De Morgan*: $\neg(P \vee Q) \equiv \neg P \wedge \neg Q$

More Equivalences

- Contrapositive: $P \Rightarrow Q \equiv \neg Q \Rightarrow \neg P$
- Double Negation: $\neg\neg P \equiv P$
- Commutativity: $P \wedge Q \equiv Q \wedge P$
- Associativity: $(P \wedge Q) \wedge R \equiv P \wedge (Q \wedge R)$

Limitations of Propositional Logic

Core Problem

PL cannot represent *general statements* about objects, properties, and relations without creating one symbol per object-property combination.

Classic Syllogism in English:

- All humans are mortal.
- Socrates is a human.
- $\therefore$ Socrates is mortal.

In PL (no generalization):

- *HumanSocrates*, *MortalSocrates*
- *HumanPlato*, *MortalPlato*
- *HumanAristotle*, *MortalAristotle*
- Thousands of separate rules!

What we actually need:

- Variables: $\forall x$, $\exists y$
- Predicates: $Human(x)$, $Mortal(x)$
- One rule: $\forall x\ Human(x) \Rightarrow Mortal(x)$
- Functions: $MotherOf(x)$, $FatherOf(x)$

Solution: First-Order Logic!

FOL provides variables, predicates, functions, and quantifiers.

First-Order Logic (FOL) = Predicate Logic

FOL extends PL by allowing quantification over **objects** in a domain and expressing **properties** and **relations** among them.

Building blocks:

- **Constants**: specific objects — *John*, 3, *Paris*
- **Variables**: stand for any object — x, y, z
- **Predicates**: properties/relations — $Human(x)$, $Loves(x, y)$
- **Functions**: map objects to objects — $MotherOf(x)$
- **Quantifiers**: $\forall$ (for all), $\exists$ (there exists)
- **Connectives**: same as PL ($\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$)

Example FOL Sentences

- $Human(Socrates)$
  "Socrates is human"
- $\forall x\ Human(x) \Rightarrow Mortal(x)$
  "All humans are mortal"
- $\exists x\ Loves(John, x)$
  "John loves someone"
- $\forall x\ \exists y\ Loves(x, y)$
  "Everyone loves someone"

Terms

A **term** is a logical expression that refers to an *object*. Terms are used as arguments to predicates and functions.

Three kinds of terms:

1. **Constant symbols**: refer to specific objects
   Examples: *John*, *Paris*, 2, *India*
2. **Variable symbols**: refer to any object in the domain
   Examples: x, y, z, *person*
3. **Function terms**: $f(t_1, \ldots, t_n)$ where f is a function symbol
   Examples: $MotherOf(John)$, $LeftLegOf(x)$, $Plus(2, 3)$

Examples with Nesting

- *John* — constant term
- x — variable term
- $MotherOf(John)$ — function applied to constant
- $MotherOf(FatherOf(x))$ — nested functions ("maternal grandfather of x")

Atomic Sentence

An **atomic sentence** is either:

- $Predicate(t_1, t_2, \ldots, t_n)$ — n-ary predicate applied to n terms
- $t_1 = t_2$ — equality of two terms

Atomic sentences are either *true* or *false* under an interpretation.

Examples by arity:

- *Unary* (property):
  $Human(Socrates)$ — "Socrates is human"
  $Prime(7)$ — "7 is prime"
- *Binary* (relation):
  $Loves(John, Mary)$ — "John loves Mary"
  $Parent(George, Philip)$
- *Ternary*:
  $Sells(West, Missiles, Nono)$
  $Between(Delhi, Agra, Jaipur)$

Predicate vs. Function

- **Predicate**: returns *true/false*
  $Human(x)$, $Loves(x, y)$
- **Function**: returns an *object*
  $MotherOf(x)$, $Plus(x, y)$

Equality

$MotherOf(Luke) = Padme$

Universal Quantifier $\forall$

$\forall x\ \phi(x)$: "For *every* object x in the domain, $\phi(x)$ holds."

Typically paired with $\Rightarrow$:
$\forall x\ Cat(x) \Rightarrow HasTail(x)$
"All cats have tails"

Existential Quantifier $\exists$

$\exists x\ \phi(x)$: "There *exists* at least one object x such that $\phi(x)$."

Typically paired with $\wedge$:
$\exists x\ Cat(x) \wedge Black(x)$
"Some cat is black"

Scope and binding:

- $\forall x\ [P(x) \Rightarrow Q(x)]$: x is *bound*
- $P(y)$: y is *free* (must be given value)
- A sentence with no free variables is a *closed formula*

Common Mistakes

Wrong: $\forall x\ Cat(x) \wedge HasTail(x)$
(Claims everything is a cat!)

Wrong: $\exists x\ Cat(x) \Rightarrow Black(x)$
(True even if no cats exist!)

Order of Quantifiers Matters!

Switching $\forall$ and $\exists$ changes meaning fundamentally.

Sentence	Meaning
$\forall x\ \forall y\ Loves(x, y)$	Everyone loves everyone
$\forall x\ \exists y\ Loves(x, y)$	Everyone loves *someone* (possibly different)
$\exists x\ \forall y\ Loves(x, y)$	*Someone* loves everyone (one specific person)
$\exists x\ \exists y\ Loves(x, y)$	At least one love pair exists

Real-world Example

$\forall x\ Student(x) \Rightarrow \exists y\ Teacher(y) \wedge Teaches(y, x)$
"Every student has some teacher."

$\exists y\ Teacher(y) \wedge \forall x\ Student(x) \Rightarrow Teaches(y, x)$
"There exists a teacher who teaches every student."

FOL Syntax — Formal Grammar (BNF)

BNF Grammar of FOL

$$
\begin{aligned}
\textit{Sentence} &::= \textit{AtomicSentence} \mid \neg \textit{Sentence} \\
&\mid \textit{Sentence} \wedge \textit{Sentence} \mid \textit{Sentence} \vee \textit{Sentence} \\
&\mid \textit{Sentence} \Rightarrow \textit{Sentence} \mid \textit{Sentence} \Leftrightarrow \textit{Sentence} \\
&\mid \forall \textit{Variable Sentence} \mid \exists \textit{Variable Sentence} \\
\textit{AtomicSentence} &::= \textit{Predicate}(\textit{Term}_1, \ldots, \textit{Term}_n) \mid \textit{Term}_1 = \textit{Term}_2 \\
\textit{Term} &::= \textit{Constant} \mid \textit{Variable} \mid \textit{Function}(\textit{Term}_1, \ldots, \textit{Term}_n) \\
\textit{Constant} &::= \textit{John} \mid \textit{Paris} \mid 3 \mid \ldots \\
\textit{Variable} &::= x \mid y \mid z \mid \ldots \\
\textit{Predicate} &::= \textit{Human} \mid \textit{Loves} \mid \textit{Parent} \mid \ldots \\
\textit{Function} &::= \textit{MotherOf} \mid \textit{Plus} \mid \ldots
\end{aligned}
$$

Well-formed Sentences

$\forall x \ (\textit{Dog}(x) \Rightarrow \textit{Animal}(x))$ ✓

$\exists x \ \exists y \ (\textit{Parent}(x, y) \wedge \textit{Parent}(y, \textit{John}))$ ✓

Interpretation / Model

An **interpretation** $\mathcal{I}$ assigns meaning to all symbols:

- A non-empty **domain** D (the set of all objects in the world)
- Each *constant* $c \rightarrow$ a specific element $c^{\mathcal{I}} \in D$
- Each *n-ary function* $f \rightarrow$ a function $f^{\mathcal{I}} : D^n \rightarrow D$
- Each *n-ary predicate* $P \rightarrow$ a relation $P^{\mathcal{I}} \subseteq D^n$

Concrete Example

Domain: $D = \{John, Mary, Bob\}$
Constants: $John^{\mathcal{I}} = John$, $Mary^{\mathcal{I}} = Mary$
Predicates: $Human^{\mathcal{I}} = \{John, Mary, Bob\}$, $Mortal^{\mathcal{I}} = \{John, Mary, Bob\}$
Relations: $Loves^{\mathcal{I}} = \{(John, Mary), (Mary, Bob)\}$

Under this interpretation:
$\forall x\ Human(x) \Rightarrow Mortal(x)$ is **TRUE**
$\forall x\ Loves(x, Mary)$ is **FALSE** (Bob doesn't love Mary)

Key Semantic Concepts

- **Model**: interpretation making a sentence true
- **Valid/Tautology**: true in *all* interpretations
  e.g. $\forall x\ P(x) \vee \neg P(x)$
- **Satisfiable**: true in *some* interpretation
- **Unsatisfiable/Contradiction**: true in *no* interpretation
  e.g. $P(a) \wedge \neg P(a)$
- **Entailment** ($\alpha \models \beta$): every model of α is also a model of β

Classic Entailment Example

KB:
$\{Human(Socrates),$
$\forall x\ Human(x) \Rightarrow Mortal(x)\}$

Query: $Mortal(Socrates)$?

Every model of KB makes $Mortal(Socrates)$ true.
$\therefore KB \models Mortal(Socrates)$ ✓

Inference Goal

Find a *sound and complete* procedure to derive all entailed sentences.

De Morgan's Laws for Quantifiers

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

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

In English:

- $\neg\forall x\ Human(x)$
  "Not everyone is human"
  $\equiv \exists x\ \neg Human(x)$ (someone is not human)
- $\neg\exists x\ Perfect(x)$
  "No one is perfect"
  $\equiv \forall x\ \neg Perfect(x)$ (everyone is imperfect)

Substitution: Variable Assignment

To evaluate $\forall x\ P(x)$ under interpretation $\mathcal{I}$:
Must be true for *every* value of x in D.

To evaluate $\exists x\ P(x)$:
Must be true for *at least one* value of x in D.

Domain: Family relationships in the British royal family

Facts (Ground Atoms)

$Parent(George, Philip)$, $Parent(Philip, Charles)$, $Parent(Philip, Anne)$
$Male(George)$, $Male(Philip)$, $Male(Charles)$, $Female(Anne)$

Rules (Universal Sentences)

- $\forall x \forall y\ Parent(x, y) \land Male(x) \Rightarrow Father(x, y)$
- $\forall x \forall y\ Parent(x, y) \land Female(x) \Rightarrow Mother(x, y)$
- $\forall x \forall y \forall z\ Parent(x, y) \land Parent(y, z) \Rightarrow Grandparent(x, z)$
- $\forall x \forall y\ Parent(x, y) \Rightarrow Ancestor(x, y)$
- $\forall x \forall y \forall z\ Ancestor(x, y) \land Parent(y, z) \Rightarrow Ancestor(x, z)$

Derived Facts by Inference

$Father(George, Philip)$, $Father(Philip, Charles)$, $Father(Philip, Anne)$
$Grandparent(George, Charles)$, $Ancestor(George, Charles)$

Agent percepts and KB rules:

Causal Rules

$\forall s\ Breezy(s) \Leftrightarrow$
$\quad \exists r\ Adjacent(r, s) \wedge Pit(r)$

"A square is breezy iff a pit is adjacent"

$\forall s\ Smelly(s) \Leftrightarrow$
$\quad \exists r\ Adjacent(r, s) \wedge Wumpus(r)$

$\forall s\ Glitter(s) \Leftrightarrow Gold(s)$

Geometry Rules

$\forall x \forall y\ Adjacent([x, y], [x + 1, y])$
$\forall x \forall y\ Adjacent([x, y], [x, y + 1])$
$\forall s\ Adjacent(s, r) \Rightarrow Adjacent(r, s)$

Query

Given $Breezy([1, 2])$, derive:
$\exists r\ Pit(r) \wedge Adjacent(r, [1, 2])$

$\Rightarrow$ Pit at $[1, 3]$ or $[2, 2]$!

Peano Arithmetic:

$NatNum(0)$
$\forall n\ NatNum(n) \Rightarrow NatNum(S(n))$
$\forall n\ S(n) \neq 0$
$Plus(0, x) = x$
$Plus(S(x), y) = S(Plus(x, y))$
$Times(0, x) = 0$
$Times(S(x), y) = Plus(y, Times(x, y))$

Set Theory:

$\forall x\ \neg Member(x, EmptySet)$

$\forall x \forall s\ Member(x, AddTo(x, s))$

$\forall x \forall y \forall s\ Member(x, AddTo(y, s))$
$\quad \Leftrightarrow (x = y \vee Member(x, s))$

$\forall x \forall s\ Member(x, s) \Rightarrow$
$\quad \neg Member(x, Remove(x, s))$

Inference — Core Concepts

Definition

Inference is the process of deriving new sentences from existing KB sentences using rules, such that derived sentences are *entailed* by the KB.

Soundness and Completeness

- **Sound** (correct): only derives entailed sentences

$$\text{If } KB \vdash_i \alpha \Rightarrow KB \models \alpha$$

- **Complete**: derives all entailed sentences

$$\text{If } KB \models \alpha \Rightarrow KB \vdash_i \alpha$$

Notation

$KB \models \alpha$: semantic entailment

$KB \vdash_i \alpha$: derived by procedure i

Main Inference Methods:

1. **Model Checking**
   Enumerate all models; check each
2. **Theorem Proving**
   - Apply inference rules
   - Forward Chaining
   - Backward Chaining
   - Resolution

Inference Rules — Modus Ponens and Tollens

Modus Ponens (MP)

$$\frac{\alpha \Rightarrow \beta \quad \alpha}{\beta}$$

"If P implies Q and P is true, then Q is true."

Modus Tollens (MT)

$$\frac{\alpha \Rightarrow \beta \quad \neg\beta}{\neg\alpha}$$

"If P implies Q and Q is false, then P is false."

MP Example

$Rain \Rightarrow Wet$
$Rain$ (fact)
$\therefore Wet$ ✓

MT Example

$Rain \Rightarrow Wet$
$\neg Wet$ (ground is dry)
$\therefore \neg Rain$ ✓

And-Elimination

$$\frac{\alpha \wedge \beta}{\alpha} \qquad \frac{\alpha \wedge \beta}{\beta}$$

Extract a conjunct from a conjunction.

And-Introduction

$$\frac{\alpha \quad \beta}{\alpha \wedge \beta}$$

Or-Introduction

$$\frac{\alpha}{\alpha \vee \beta}$$

Hypothetical Syllogism

$$\frac{\alpha \Rightarrow \beta \quad \beta \Rightarrow \gamma}{\alpha \Rightarrow \gamma}$$

Disjunctive Syllogism

$$\frac{\alpha \vee \beta \quad \neg\alpha}{\beta}$$

Chaining Example

$P \Rightarrow Q$, $Q \Rightarrow R$, P
By HS: $P \Rightarrow R$
By MP: R ✓

Universal Instantiation

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

where g is any **ground term** (a term containing no variables).
"We may substitute any ground term for a universally quantified variable."

Example with Multiple Instantiations

Sentence: $\forall x\ Human(x) \Rightarrow Mortal(x)$

Substitution $\{x/Socrates\}$:	$Human(Socrates) \Rightarrow Mortal(Socrates)$
Substitution $\{x/Plato\}$:	$Human(Plato) \Rightarrow Mortal(Plato)$
Substitution $\{x/MotherOf(John)\}$:	$Human(MotherOf(John)) \Rightarrow Mortal(MotherOf(John))$

Note

UI can be applied any number of times. Combined with known facts and Modus Ponens, it drives forward and backward chaining.

Existential Instantiation

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

where k is a **Skolem constant** — a brand-new constant not appearing anywhere in the KB.

Example

Sentence: $\exists x\ Crown(x) \wedge OnHead(x, John)$
Introduce Skolem constant C_1:
$Crown(C_1) \wedge OnHead(C_1, John)$

We don't know *which* crown, but there is one, so we give it a name.

Key Difference from UI

- UI: can instantiate *multiple* times with different terms
- EI: instantiate *once* with a new constant; cannot substitute the new constant back

Unification

Unification finds a substitution θ that makes two expressions identical:
$\text{UNIFY}(\alpha, \beta) = \theta$ such that $\text{SUBST}(\theta, \alpha) = \text{SUBST}(\theta, \beta)$

Expression 1	Expression 2	Unifier θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, Bill)$	$\{y/John, x/Bill\}$
$Knows(John, x)$	$Knows(y, MotherOf(y))$	$\{y/John, x/MotherOf(John)\}$
$Knows(John, x)$	$Knows(x, Elizabeth)$	FAIL (variable conflict)
$Animal(x)$	$Animal(Cat)$	$\{x/Cat\}$
$Loves(x, x)$	$Loves(John, Mary)$	FAIL (x can't be both)

Most General Unifier (MGU)

The **MGU** is the *least constraining* substitution. Use **standardizing apart** (rename variables) to avoid variable conflicts.

Generalized Modus Ponens (GMP)

Generalized Modus Ponens

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

where $\mathrm{SUBST}(\theta, p_i') = \mathrm{SUBST}(\theta, p_i)$ for all i.

Example

KB:

- Rule: $King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
- Fact 1: $King(John)$
- Fact 2: $Greedy(John)$

Apply GMP with $\theta = \{x/John\}$:
$p_1' = King(John)$, $p_2' = Greedy(John)$
Conclusion: $\mathrm{SUBST}(\{x/John\}, Evil(x)) = Evil(John)$ ✓

Idea: Ground Instantiation

Convert FOL to propositional logic by substituting every possible ground term from the KB for universally quantified variables, then apply propositional inference.

Steps:

1. For $\forall x\ \alpha$, substitute every ground term
2. Replace $\exists x$ with Skolem constants/functions
3. Apply propositional resolution or truth tables

Herbrand's Theorem (1930)

If α is entailed by a FOL KB, there is a *finite* set of ground instances that propositionally entail α.

Example

KB: $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
Constants: $\{John, Richard\}$

Grounds:

- $King(John) \wedge Greedy(John) \Rightarrow Evil(John)$
- $King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$

Problem

Infinite domains $\rightarrow$ infinitely many ground instances!

Skolemization — Eliminating Existential Quantifiers

Replace each $\exists x$ with a **Skolem function** of the enclosing universally quantified variables. If no enclosing $\forall$, use a *Skolem constant*.

Original FOL	After Skolemization	Note
$\exists x\ P(x)$	$P(A)$	A = new Skolem constant
$\forall x\ \exists y\ P(x, y)$	$\forall x\ P(x, f(x))$	f = Skolem function of x
$\forall x \forall y\ \exists z\ P(x, y, z)$	$\forall x \forall y\ P(x, y, g(x, y))$	g depends on x, y

Example: "Everyone has a mother"

Original: $\forall x\ \exists y\ \mathit{MotherOf}(y, x)$
Skolemized: $\forall x\ \mathit{MotherOf}(m(x), x)$ where m is a Skolem function.

Note: $m(x)$ denotes "the mother of x" without naming a specific individual.

Converting to Conjunctive Normal Form (CNF)

Steps to Convert FOL Sentence to CNF

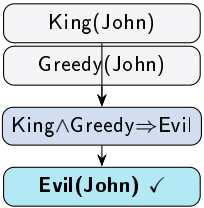
1. **Eliminate biconditionals**: $\alpha \Leftrightarrow \beta \rightarrow (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$
2. **Eliminate implications**: $\alpha \Rightarrow \beta \rightarrow \neg\alpha \vee \beta$
3. **Move $\neg$ inward**: De Morgan's laws; quantifier duality
4. **Standardize variables**: rename so each quantifier uses a unique variable
5. **Skolemize**: eliminate all $\exists$ quantifiers
6. **Drop universal quantifiers**: all remaining variables are universally quantified
7. **Distribute $\vee$ over $\wedge$**: get conjunction of disjunctions (clauses)

Example

$\forall x\ Animal(x) \Rightarrow \exists y\ Loves(x, y)$
Step 2: $\forall x\ \neg Animal(x) \vee \exists y\ Loves(x, y)$
Step 4-5: $\forall x\ \neg Animal(x) \vee Loves(x, f(x))$
Step 6-7 (CNF clause): $\neg Animal(x) \vee Loves(x, f(x))$

Forward Chaining (Data-Driven / Bottom-Up)

Start from **known facts** in the KB, apply inference rules to derive new facts, and repeat until the query is proved or no new facts can be derived.



Algorithm idea:

repeat until query proved or no change:
  for each rule $p_1 \wedge \cdots \wedge p_n \Rightarrow q$:
    find all θ s.t. SUBST$(\theta, \{p_i\}) \subseteq$ KB
    add SUBST(θ, q) to KB

Properties:

- **Sound** and **Complete** for Datalog (function-free

Forward Chaining — Knowledge Base (Criminal Example)

KB — Horn Clauses

1. $American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow Criminal(x)$
2. $Enemy(x, America) \Rightarrow Hostile(x)$
3. $Missile(x) \Rightarrow Weapon(x)$
4. $Missile(x) \land Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$

Known Facts

$Owns(Nono, M_1)$, $Missile(M_1)$, $American(West)$, $Enemy(Nono, America)$

Goal

Prove: $Criminal(West)$

Forward Chaining — Step-by-Step Trace

Step	Rule Applied	Substitution	New Fact
1	Rule 3: $Missile(x) \Rightarrow Weapon(x)$	$\{x/M_1\}$	$Weapon(M_1)$
2	Rule 2: $Enemy(x, America) \Rightarrow Hostile(x)$	$\{x/Nono\}$	$Hostile(Nono)$
3	Rule 4 + $Missile(M_1)$ + $Owns(Nono, M_1)$	$\{x/M_1\}$	$Sells(West, M_1, Nono)$
4	Rule 1 + Step 1,2,3 + $American(West)$	$\{x/West,$ $y/M_1, z/Nono\}$	$Criminal(West)$ ✓

Conclusion

Forward chaining derived $Criminal(West)$ in **4 steps** by data-driven reasoning.

Observation

All derived facts were *relevant* here, but in large KBs, many irrelevant facts may be generated. → Motivates **Backward Chaining**.

Forward Chaining — FOL-FC-ASK Algorithm

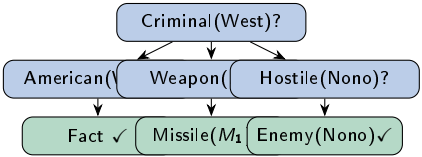
FOL-FC-ASK(KB, α)

```
new ← {}
loop:
  for each rule (l_1 ∧ ··· ∧ l_n ⇒ l) in KB:
    for each θ s.t. UNIFY(l_i, k_i) succeeds ∀i:
      q ← SUBST(θ, l)
      if q ∉ KB∪ new:  add q to new
      if UNIFY(q, α) succeeds:  return θ
  if new = ∅:  return false
  KB ← KB∪ new
```

- Returns unifier θ proving query, or `false` if not provable
- **Complete** for Datalog (function-free definite clauses)
- May not terminate for full FOL with function symbols

Backward Chaining (Goal-Driven / Top-Down)

Start from the **goal** (query), find rules whose *head* matches the goal, then recursively prove each condition in the rule's *body*.



Algorithm sketch:

```
BC(goal, KB):
  if goal ∈ KB (fact):  return SUCCESS
  for each rule head ⇐ b_1 ∧ ··· ∧ b_n in KB:
    θ ← UNIFY(goal, head)
    if θ fails:  continue
    prove each SUBST(θ, b_i) recursively
```

Properties:

- Focuses only on *goal-relevant* subproblems

Backward Chaining — Detailed Trace

Same KB as Forward Chaining. Goal: $Criminal(West)$

AND-OR Tree Expansion

1. To prove $Criminal(West)$, match head of Rule 1 with $\theta = \{x/West\}$.
   Need to prove: $American(West)$, $Weapon(y)$, $Sells(West, y, z)$, $Hostile(z)$
2. $American(West)$: **fact in KB** ✓
3. $Weapon(y)$: match Rule 3 ($Missile(y) \Rightarrow Weapon(y)$). Try $y = M_1$.
   Need: $Missile(M_1)$ — **fact in KB** ✓ $\rightarrow$ $Weapon(M_1)$ ✓
4. $Sells(West, M_1, z)$: match Rule 4. Need $Missile(M_1)$ ✓ and $Owns(Nono, M_1)$ ✓
   $\Rightarrow z = Nono$, $Sells(West, M_1, Nono)$ ✓
5. $Hostile(Nono)$: match Rule 2. Need $Enemy(Nono, America)$ — **fact in KB** ✓

Result

All sub-goals proved! $\Rightarrow Criminal(West)$ with $\theta = \{x/West, y/M_1, z/Nono\}$ ✓

Backward Chaining — FOL-BC-ASK Algorithm

FOL-BC-ASK(KB, goals, θ)

```
if goals = []:  yield θ; return
q ← SUBST(θ, FIRST(goals))
for each clause (lhs ⇒ rhs) in KB:
  (l, rhs') ← STANDARDIZE-APART(lhs ⇒ rhs)
  θ' ← UNIFY(q, l); if FAIL: continue
  for each θ'' in FOL-BC-ASK(KB, rhs'∘REST(goals), COMPOSE(θ', θ)):
    yield θ''
```

Properties

- Sound and complete for definite clauses
- Uses depth-first search
- STANDARDIZE-APART avoids variable name collisions
- COMPOSE combines substitutions

Potential Problem: Loops

KB: $P(x) \Rightarrow Q(x)$, $Q(x) \Rightarrow P(x)$
Query: $P(a)$?
BC loops: $P(a) \rightarrow Q(a) \rightarrow P(a) \rightarrow \ldots$
Solution: Keep track of goals already being proved (tabling/memoization).

Forward vs. Backward Chaining — Comparison

Property	Forward Chaining	Backward Chaining
Direction	Data $\rightarrow$ Goal	Goal $\rightarrow$ Data
Driven by	Known facts	Query/goal
Efficiency	Can generate irrelevant facts	Focuses on query
Termination	Terminates (Datalog)	May loop
Completeness	Complete (Horn/Datalog)	Complete (Horn/Datalog)
Loop handling	Naturally avoids	Needs explicit check
Paradigm	Production systems	Prolog
Best for	Monitoring, all conclusions	Specific queries

Rule of Thumb

Forward: when you want all consequences of new data (e.g., sensor triggers, database updates).
Backward: when you have a specific question and want to avoid irrelevant work (e.g., medical diagnosis).

Why Resolution?

Modus Ponens and chaining work only for **Horn clauses** (at most one positive literal). **Resolution** is a single, powerful inference rule that is *refutation-complete* for full FOL.

Core idea (propositional):

$$\frac{\ell_1 \vee \cdots \vee \ell_k \quad m_1 \vee \cdots \vee m_n}{\ell_1 \vee \cdots \vee \ell_{k-1} \vee m_1 \vee \cdots \vee m_{n-1}}$$

where ℓ_k and m_n are **complementary literals**: $\ell_k = \neg m_n$.

The result is called the **resolvent**.

Simple Example

Clause 1: $(A \vee B)$
Clause 2: $(\neg B \vee C)$
Resolve on B and $\neg B$:
Resolvent: $(A \vee C)$

Proof Strategy

Prove $KB \models \alpha$ by showing $KB \wedge \neg\alpha$ is *unsatisfiable*.

Propositional Resolution — Examples

Example 1: Direct Resolution

Clause 1: $(P \vee Q \vee R)$
Clause 2: $(\neg Q \vee S)$
Resolve on $Q/\neg Q$: **Resolvent** $= (P \vee R \vee S)$

Example 2: Refutation Proof

KB: $\{P \Rightarrow Q,\ P\}$ **Query**: Q?

Convert to CNF:
(1) $\neg P \vee Q$ (2) P (3) $\neg Q$ (negated query)

Step 1: Resolve (1) and (2) on $P/\neg P$: get Q
Step 2: Resolve Q and (3) $\neg Q$: get $\square$ (empty clause)

Contradiction found! $\Rightarrow Q$ is proved. ✓

Empty Clause $\square$

The **empty clause** ($\square$) represents *false* (contradiction). Deriving $\square$ proves the query by refutation.

FOL Resolution Rule

FOL Resolution (with Unification)

$$\frac{\ell_1 \vee \cdots \vee \ell_k \quad m_1 \vee \cdots \vee m_n}{\text{SUBST}(\theta,\ \ell_1 \vee \cdots \vee \ell_{k-1} \vee m_1 \vee \cdots \vee m_{n-1})}$$

where $\theta = \text{UNIFY}(\ell_k, \neg m_n)$ (MGU)

FOL Resolution Example

Clause 1: $\neg Human(x) \vee Mortal(x)$
Clause 2: $Human(Socrates)$

Unify $Human(x)$ with $Human(Socrates)$: $\theta = \{x/Socrates\}$
Resolve on $Human(x)$ / $\neg Human(x)$:
Resolvent: $\text{SUBST}(\{x/Socrates\}, Mortal(x)) = Mortal(Socrates)$ ✓

Factoring

If two literals in the same clause unify, keep only one copy:
$P(x) \vee P(f(y)) \vee Q(x)$ with $\{x/f(y)\}$: factored to $P(f(y)) \vee Q(f(y))$

Resolution Refutation — Full Procedure

Resolution Refutation Algorithm

To prove $KB \models \alpha$:

1. Convert $KB \cup \{\neg\alpha\}$ to CNF
2. Build set of clauses S
3. **Repeat**:
   - Select two clauses $C_i, C_j \in S$ with complementary literals
   - Compute resolvent $r = \text{RESOLVE}(C_i, C_j)$
   - If $r = \square$ (empty clause): **return TRUE** (proved)
   - If $r \notin S$: add r to S
4. If no new resolvents: **return FALSE** (not provable)

Theoretical Basis

$KB \models \alpha \iff KB \wedge \neg\alpha$ is *unsatisfiable*.

Robinson's Theorem (1965): Resolution is *refutation-complete* for full FOL.

KB converted to CNF clauses:

1. $\neg American(x) \vee \neg Weapon(y) \vee \neg Sells(x, y, z) \vee \neg Hostile(z) \vee Criminal(x)$
2. $\neg Enemy(x, America) \vee Hostile(x)$
3. $\neg Missile(x) \vee Weapon(x)$
4. $\neg Missile(x) \vee \neg Owns(Nono, x) \vee Sells(West, x, Nono)$
5. $American(West)$
6. $Missile(M_1)$
7. $Owns(Nono, M_1)$
8. $Enemy(Nono, America)$
9. $\neg Criminal(West)$ *(negated query)*

Derivation Steps

Step	From Clauses	Unifier	Result
R1	3 + 6	$\{x/M_1\}$	$Weapon(M_1)$
R2	2 + 8	$\{x/Nono\}$	$Hostile(Nono)$
R3	4 + 6 + 7	$\{x/M_1\}$	$Sells(West, M_1, Nono)$
R4	1 + 5 + R1 + R3 + R2	$\{x/West, y/M_1, z/Nono\}$	$Criminal(West)$
R5	R4 + 9	—	$\square$ (empty clause)

Conclusion

Derived the empty clause $\square \Rightarrow$ Contradiction reached $\Rightarrow$ $Criminal(West)$ is TRUE ✓

Strategies to Control Search

- **Unit resolution**: always resolve with a unit clause (single literal). Fast but incomplete in general.
- **Set of support**: at least one parent must come from (or trace to) the negated query set. Complete and focused.
- **Input resolution**: one parent must always be an original KB clause. Incomplete for full FOL.
- **Subsumption**: delete any clause C if another clause C' is more general (subsumes C). Reduces search space.
- **Linear resolution**: a generalization of input resolution; complete.

Completeness Summary

Strategy	Complete?
General res.	Yes
Unit res.	No (general), Yes (Horn)
Set of support	Yes
Input res.	No (general), Yes (Horn)
Linear res.	Yes

Semi-Decidability

FOL is *undecidable* in general. Resolution is *semi-decidable*: if α is provable, resolution will find the proof; if not, it may run forever.

Summary — Comparison of Inference Methods

Method	Logic	Complete?	Strategy	Use Case
Truth tables	PL	Yes	Enumerate models	Small PL KBs
Modus Ponens	Horn PL/FOL	Yes (HC)	Rule application	Simple rules
Forward Chaining	Horn FOL	Yes (HC)	Data-driven	Monitoring
Backward Chaining	Horn FOL	Yes (HC)	Goal-driven	Query answering
Resolution	Full PL/FOL	Refutation-complete	Refutation	General

Key Takeaways

- **Horn clauses**: FC and BC are efficient and complete
- **Full FOL**: Resolution is the general-purpose method
- **Prolog**: implements BC with depth-first search + unification
- **Soundness always**: we never want to derive false conclusions

Lecture Summary

Knowledge Representation:

- KBA: KB + Inference Engine
- PL: limited expressiveness
- FOL: rich representation with quantifiers, predicates, functions

FOL Syntax:

- Terms: constants, variables, functions
- Atoms: predicates, equality
- Quantifiers: $\forall$ (with $\Rightarrow$), $\exists$ (with $\wedge$)

FOL Semantics:

- Interpretation, domain, model
- Entailment $KB \models \alpha$

Inference in FOL:

- UI, EI, Unification, GMP
- Skolemization, CNF conversion

Chaining:

- Forward: data-driven, all conclusions
- Backward: goal-driven, Prolog-style
- Both complete for Horn clauses

Resolution:

- Refutation-complete for full FOL
- Requires CNF conversion
- Derives $\square \Rightarrow$ query proved

Core Principle

Sound + Complete inference = Intelligent reasoning

Thank you for your attention!

Post your doubts:

https://www.gcjana.in/courses/shardauniversity/2501/CSE472/#Post-doubts

Dr. Gopal Chandra Jana · Dept. of CSE, Sharda University