Intelligent Agents

Propositional and First Order Logic

Curtis Larsen

Utah Tech University—Computing

Fall 2025

Review: Why Logic?

- Provides a precise language for representing facts and rules
- Enables rigorous inference: deriving conclusions from premises
- Foundation for knowledge-based agents in AI
- Bridges human reasoning and machine reasoning

Representing Knowledge in AI

- Knowledge representation = encoding information about the world
- Propositional logic: facts as true/false statements
- First-order logic: extends with objects, relations, and quantifiers
- Central to building systems that *know*, *reason*, and *act*

Role in Reasoning and Decision-Making

- Logic enables **inference**: deducing new facts from known ones
- Supports planning: selecting actions consistent with goals
- Handles uncertainty via extensions (probabilistic logics, Bayesian nets)
- Critical for autonomous decision-making in intelligent agents

Operators of Propositional Logic

- **Negation** ($\neg p$): true when p is false
- **Conjunction** ($p \wedge q$): true when both p and q are true
- **Disjunction** ($p \vee q$): true when at least one of p or q is true
- **Implication** ($p \rightarrow q$): false only if p is true and q is false
- **Biconditional** ($p \leftrightarrow q$): true when p and q have the same truth value

Truth Tables for Propositional Logic

Operators: $\neg$ (not), $\wedge$ (and), $\vee$ (or), $\rightarrow$ (implies), $\leftrightarrow$ (iff)

p	$\neg p$
T	F
F	T

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

Satisfiability

- A propositional sentence is **satisfiable** if there exists at least one assignment of truth values that makes it true.
- Example:

$$(P \vee Q) \wedge \neg P$$

  is satisfiable (set $P =$ False, $Q =$ True).
- A sentence is **unsatisfiable** if no assignment makes it true (i.e., always false).
- A sentence is **valid** if all assignments make it true (i.e., a tautology).

Entailment

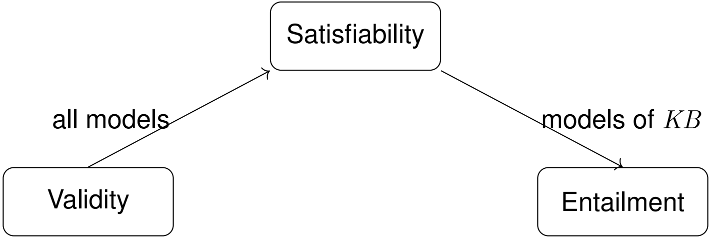
- Knowledge base KB **entails** a sentence α (written $KB \models \alpha$) if every model of KB is also a model of α.
- Equivalently: Whenever KB is true, α must also be true.
- Example:

$$KB = \{P \to Q, P\}, \quad \alpha = Q$$

  Then $KB \models \alpha$.
- Entailment is the semantic foundation of logical inference.

Relationship of Validity, Satisfiability, Entailment

- α is **valid** $\iff$ $\neg\alpha$ is unsatisfiable.
- $KB \models \alpha \iff (KB \wedge \neg\alpha)$ is unsatisfiable.
- Entailment connects satisfiability with inference: Checking $KB \models \alpha$ reduces to checking unsatisfiability.



Motivating Example: Course Advisor

- Student: **Alice**
- Transcript: CS 1030, CS 1400, CS 1410
- Course prerequisites (subset):
  - CS 1400 $\rightarrow$ CS 1410
  - CS 1410 $\rightarrow$ CS 2100, CS 2420
  - CS 2420, CS 2810, CS 3005 $\rightarrow$ CS 3400
- Course offerings:
  - CS 2100, 2420 offered in Fall
  - CS 3400 offered in Spring
- Question: What can Alice take next term?

Propositional Encoding

- Symbols for Alice:
  - $P_{1030}, P_{1400}, P_{1410}$ = true
  - $P_{2100}, P_{2420}, P_{3400}$ = not yet
  - $O_{2100,F}$, $O_{2420,F}$, $O_{3400,S}$ = offerings
- Rules:
  - $P_{1400} \rightarrow E_{1410}$
  - $P_{1410} \wedge O_{2100,F} \rightarrow E_{2100,F}$
  - $P_{1410} \wedge O_{2420,F} \rightarrow E_{2420,F}$
  - $(P_{2420} \wedge P_{2810} \wedge P_{3005}) \wedge O_{3400,S} \rightarrow E_{3400,S}$
- **TELL:** $P_{1030}, P_{1400}, P_{1410}$: Alice has taken these courses.
- **ASK:** $E_{2420,F}$: Is Alice eligible for CS 2420 (Fall)?

Reasoning in Propositional Logic

- From P_{1410} and $O_{2420,F}$: infer $E_{2420,F}$ $\rightarrow$ Alice can take CS 2420 this Fall.
- From P_{1410} and $O_{2100,F}$: infer $E_{2100,F}$ $\rightarrow$ Alice can take CS 2100 this Fall.
- Cannot infer $E_{3400,S}$ (prereqs not yet satisfied).

Motivating Example: Wumpus World

- Classic AI testbed: an agent explores a cave of rooms (grid world)
- Hazards: **Wumpus** (monster) and **pits**
- Percepts:
  - Breeze $\Rightarrow$ a pit is adjacent
  - Stench $\Rightarrow$ the Wumpus is adjacent
- Goal: safely navigate, find the gold, avoid the Wumpus and pits
- Example scenario:
  - Agent starts at $(1, 1)$, perceives a Breeze but no Stench
  - Should it move to $(2, 1)$ or $(1, 2)$?

Propositional Logic Encoding

- Propositions:
  - $B_{i,j}$: Breeze in cell (i, j)
  - $S_{i,j}$: Stench in cell (i, j)
  - $P_{i,j}$: Pit in cell (i, j)
  - $W_{i,j}$: Wumpus in cell (i, j)
- Physics rules (Horn clauses):
  - $B_{i,j} \leftrightarrow (P_{i+1,j} \vee P_{i-1,j} \vee P_{i,j+1} \vee P_{i,j-1})$
  - $S_{i,j} \leftrightarrow (W_{i+1,j} \vee W_{i-1,j} \vee W_{i,j+1} \vee W_{i,j-1})$
- Example facts (TELL):
  - $B_{1,1}$ is true, $S_{1,1}$ is false
- Example queries (ASK):
  - Is $P_{2,1}$ possible?
  - Is $(2, 2)$ safe? ($\neg P_{2,2} \wedge \neg W_{2,2}$)

Inference by Truth Tables

- **Idea:** To check if a conclusion follows from premises:
  1. List all possible truth assignments to the propositional symbols
  2. Mark which rows satisfy the premises
  3. If the conclusion is true in every row where the premises are true, the inference is valid
- **Example:**
  - Premises: $P \to Q$, P
  - Conclusion: Q
  - Truth table shows: whenever premises are true, Q is also true
- Pros: sound, complete
- Cons: exponential in number of symbols

Inference by Truth Tables — Example (Modus Ponens)

Premises: $(P \rightarrow Q)$ and P Conclusion: Q

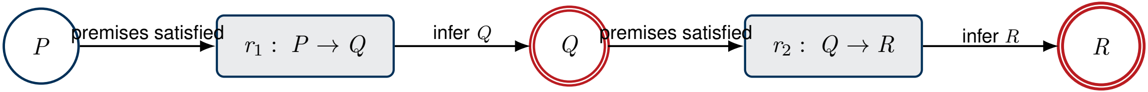
P	Q	$P \rightarrow Q$	Premises $(P \land (P \rightarrow Q))$	Conclusion Q
T	T	T	**T**	**T**
T	F	F	F	F
F	T	T	F	T
F	F	T	F	F

- The premises are true only in the first row; there, Q is also true.
- Therefore, $(P \rightarrow Q), P \models Q$ is a **valid** inference.

Forward Chaining

- Works with knowledge bases of Horn clauses (rules of form $A_1 \wedge \cdots \wedge A_k \rightarrow B$)
- **Algorithm:**
  1. Start with known facts (KB)
  2. If premises of a rule are satisfied, infer its conclusion and add it to the KB
  3. Repeat until no new inferences can be made or goal is found
- **Example:**
  - Facts: P, $P \rightarrow Q$, $Q \rightarrow R$
  - Inference: from P infer Q, then from Q infer R
- Pros: sound, complete for Horn clauses; efficient

Forward Chaining (Data-Driven) — Diagram

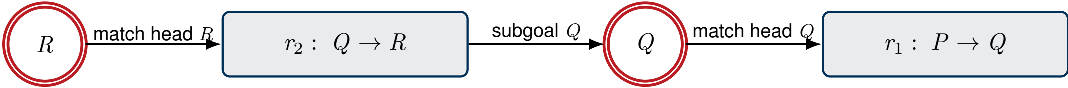


Idea: Start from known facts and fire any rule whose premises are all known, adding new conclusions until no change or the goal is derived.

Backward Chaining

- Starts from the **query/goal** and works backward
- **Algorithm:**
  1. To prove a goal G, check if G is in KB
  2. If not, look for rules with G as the conclusion
  3. Recursively try to prove each premise of those rules
- **Example:**
  - Goal: R
  - Rule: $Q \to R$
  - Sub-goal: prove Q
  - Rule: $P \to Q$
  - Sub-goal: prove P (fact in KB) $\to$ success
- Pros: focuses search on query; avoids irrelevant facts

Backward Chaining (Goal-Driven) — Diagram



all subgoals discharged $\Rightarrow$ R proved

Idea: Start from the goal, find rules whose *conclusion* matches it, and recursively prove each *premise* as a subgoal until you hit known facts.

Limitations of Propositional Logic

- **Expressiveness is limited:**
  - Cannot easily refer to individuals (e.g., "Socrates")
  - Cannot represent relations between individuals (e.g., "is a teacher of")
  - Cannot handle quantification (e.g., "for all students," "there exists a person")
- Treats whole statements as indivisible symbols
- Lacks structure needed for general knowledge representation
- General solutions have $O(2^n)$ complexity

First-Order Logic (FOL) Introduction

- Extends propositional logic with **objects**, **relations**, and **quantification.**
- Enables more expressive knowledge representation:
  - Talk about individuals, their properties, and relationships.
  - Capture general rules, not just facts.
- Foundation for knowledge-based agents.

Syntax of FOL

- **Constants:** denote specific objects (e.g., `John`, `2`).
- **Predicates:** describe relations or properties (e.g., `Loves(John, Mary)`).
- **Functions:** map objects to objects (e.g., `MotherOf(John)`).
- **Variables:** range over objects in the domain (e.g., `x`, `y`).
- **Quantifiers:**
  - Universal: $\forall x \; P(x)$ — "for all x"
  - Existential: $\exists x \; P(x)$ — "there exists an x"

Semantics of FOL

- **Interpretation:** assigns meaning to symbols
  - Constants $\rightarrow$ objects
  - Predicates $\rightarrow$ relations over objects
  - Functions $\rightarrow$ mappings between objects
- **Model:** an interpretation in which all sentences are true.
- Truth of a FOL sentence is always defined with respect to a model.

Translating Natural Language to FOL

- Step 1: Identify objects, relations, and quantifiers.
- Step 2: Write formal FOL expressions.
- Examples:
  - "All humans are mortal."
    $\forall x\ (Human(x) \rightarrow Mortal(x))$
  - "There exists a student enrolled in CS4300."
    $\exists x\ (Student(x) \wedge Enrolled(x, CS4300))$
  - "John loves Mary."
    $Loves(John, Mary)$

First-Order Logic Encoding

- Predicates:
  - $\texttt{Took}(s, c)$, $\texttt{Prereq}(p, c)$
  - $\texttt{OfferedIn}(c, t)$, $\texttt{Eligible}(s, c, t)$
- Rules:
  - $\forall s, c, p : \texttt{Took}(s, p) \wedge \texttt{Prereq}(p, c) \wedge \texttt{OfferedIn}(c, t) \rightarrow \texttt{Eligible}(s, c, t)$
  - For multiple prereqs:
    $\forall s, c : (\bigwedge_{p \in \text{Prereqs}(c)} \texttt{Took}(s, p)) \wedge \texttt{OfferedIn}(c, t) \rightarrow \texttt{Eligible}(s, c, t)$
- **TELL:**
  - $\texttt{Took}(Alice, CS1030)$, $\texttt{Took}(Alice, CS1400)$, $\texttt{Took}(Alice, CS1410)$
  - $\texttt{Prereq}(CS1410, CS2420)$, $\texttt{OfferedIn}(CS2420, Fall)$
- **ASK:** $\texttt{Eligible}(Alice, CS2420, Fall)$?

Knowledge Base Representation

- KB Facts:
  - $\texttt{Took}(Alice, CS1030)$, $\texttt{Took}(Alice, CS1400)$, $\texttt{Took}(Alice, CS1410)$
  - $\texttt{Prereq}(CS1410, CS2420)$
  - $\texttt{OfferedIn}(CS2420, Fall)$
- KB Rules:
  - $\texttt{Completed}(s, c) \leftarrow \texttt{Took}(s, c)$
  - $\texttt{AllReq}(s, c) \leftarrow \bigwedge_{p \in \texttt{Prereqs}(c)} \texttt{Completed}(s, p)$
  - $\texttt{Eligible}(s, c, t) \leftarrow \texttt{AllReq}(s, c) \wedge \texttt{OfferedIn}(c, t)$
- **ASK:** $\texttt{Eligible}(Alice, CS2420, Fall) \rightarrow$ True

First-Order Logic Encoding

- Predicates:
  - $\mathrm{Breeze}(x, y)$, $\mathrm{Pit}(x, y)$
  - $\mathrm{Stench}(x, y)$, $\mathrm{Wumpus}(x, y)$
  - $\mathrm{Adjacent}((x, y), (u, v))$
- General rules:
  - $\forall x, y \; \big(\mathrm{Breeze}(x, y) \leftrightarrow \exists u, v \; (\mathrm{Adjacent}((x, y), (u, v)) \wedge \mathrm{Pit}(u, v))\big)$
  - $\forall x, y \; \big(\mathrm{Stench}(x, y) \leftrightarrow \exists u, v \; (\mathrm{Adjacent}((x, y), (u, v)) \wedge \mathrm{Wumpus}(u, v))\big)$
- Facts (TELL):
  - $\mathrm{Breeze}(1, 1)$, $\neg\mathrm{Stench}(1, 1)$
- Example queries (ASK):
  - $\mathrm{Safe}(2, 2) \equiv \neg\mathrm{Pit}(2, 2) \wedge \neg\mathrm{Wumpus}(2, 2)$?

Knowledge-Based Agent Architecture