Logic

Intelligent Systems II

Mário Antunes

February 11, 2026

Universidade de Aveiro

Logic Revision

Prelude: A Basic Example of Boolean Logic i

Before diving into complex inference, let's recall the basics of **Boolean Logic** using a simple circuit metaphor.

- **Scenario**: A lamp (L) turns on only if *Switch A* (A) is ON **AND** *Switch B* (B) is ON.
- **Representation**:
  - 1 = True (On/High)
  - 0 = False (Off/Low)
- **Formula**: $L = A \wedge B$

Prelude: A Basic Example of Boolean Logic ii

Truth Table:

Switch A	Switch B	Lamp (L)
0	0	0
0	1	0
1	0	0
1	**1**	**1**

- **Key Concept**: Logic allows us to formalize the state of the world (A, B) and derive consequences (L) without physically checking the lamp every time.

Logic Systems Components

A logic system is defined by three main components:

1. **Syntax**: Describes the valid set of formulas that can be written.
    - *Note*: These are called **Well Formed Formulas (WFF)**.
2. **Semantics**: Connects the formulas to the facts or rules they represent in the real world.
    - *Example*: In Propositional Logic, semantics are defined via **Truth Tables**.
3. **Inference Rules**: Algorithms that allow deriving *new* formulas from *known* formulas.

Propositional Logic i

Based on **propositions** (elementary formulas that can be True or False).

- **Proposition**: An elementary formula (e.g., “Snow is white”, “Sugar is a hydrocarbon”).
- **Propositional Variable**: A variable (e.g., P, Q) that takes the truth value of a proposition.
- **Connectives**:
  - Negation ($\neg$)
  - Conjunction ($\wedge$)
  - Disjunction ($\vee$)
  - Implication ($\Rightarrow$)
  - Biconditional ($\Leftrightarrow$)

Propositional Logic ii

Example:

- P: “It is raining”
- Q : “The ground is wet”
- Formula: $P \Rightarrow Q$ (“If it is raining, then the ground is wet”)

First-Order Logic (FOL)

Propositional logic is limited (it cannot handle objects or relations easily). FOL extends this by introducing:

- **Objects (Constants)**: Specific entities.
  - *Examples*: `1215`, `Aveiro`, `Alice`.
- **Functions**: Return an object.
  - *Examples*: `Power(4,3)`, `FatherOf(Paul)`.
  - *Note*: Constants can be seen as functions with arity 0.
- **Predicates (Relations)**: Return True/False.
  - *Examples*: `Father(Rui, Paulo)`, `Brother(Paul, Rose)`.
- **Variables**: Placeholders for objects (x, y, z).

FOL Grammar and Syntax i

A well-formed formula in FOL is constructed recursively:

- **Terms**:
  - Constant | Variable | Function(Term, ...)
- **Atomic Formula**:
  - Predicate(Term, ...) | Term = Term
- **Complex Formula**:
  - $\neg$ Formula
  - Formula $\wedge$ Formula
  - Formula $\vee$ Formula
  - Formula $\Rightarrow$ Formula
  - Quantifier Variable Formula

FOL Grammar and Syntax ii

Convention:

- **Predicates/Constants/Functions**: Start with Uppercase (e.g., `Near`, `Alice`).
- **Variables**: Start with lowercase (e.g., `x`, `y`).

Quantifiers: Universal $(\forall)$

"For all x, $P(x)$ is true."

Example: "All students in Oxford are smart."

- **Correct**: $\forall x(\text{Studies}(x, \text{Oxford}) \Rightarrow \text{Smart}(x))$
- **Common Mistake**: Using $\wedge$ instead of $\Rightarrow$.
  - *Incorrect*: $\forall x(\text{Studies}(x, \text{Oxford}) \wedge \text{Smart}(x))$
  - *Why?* This means "Everyone in the universe studies at Oxford AND is smart."

Quantifiers: Existential $(\exists)$

"There exists an x such that $P(x)$ is true."

Example: "Some student in Oxford is smart."

- **Correct**: $\exists x(\text{Studies}(x, \text{Oxford}) \land \text{Smart}(x))$
- **Common Mistake**: Using $\Rightarrow$ instead of $\land$.
  - *Incorrect*: $\exists x(\text{Studies}(x, \text{Oxford}) \Rightarrow \text{Smart}(x))$
  - *Why?* The implication $A \Rightarrow B$ is true if A is false. If there is *anyone* who does not study at Oxford, the statement becomes true vacuously, even if no smart students exist.

Interpretations and Models

- **Interpretation (Prop Logic)**: An assignment of truth values to all variables.
- **Interpretation (FOL)**: A mapping of constants to objects, predicates to relations, and functions to functional mappings in the domain.
- **Model**: An interpretation that satisfies a formula (evaluates to True).
- **Satisfiability**: A formula is satisfiable if there exists at least one model for it.
- **Tautology (Validity)**: A formula is true in **all** possible interpretations.
- **Entailment (⊨)**: $KB \models \alpha$ means α is true in all models where KB is true.

Logic Rewrite Rules (Equivalences)

Useful for simplifying formulas and converting to Normal Forms.

- **De Morgan's Laws**:
  - $\neg(A \wedge B) \equiv \neg A \vee \neg B$
  - $\neg(A \vee B) \equiv \neg A \wedge \neg B$
- **Implication**:
  - $A \Rightarrow B \equiv \neg A \vee B$
- **Double Negation**:
  - $\neg\neg A \equiv A$
- **Quantifier Negation (Generalized De Morgan)**:
  - $\neg\forall x P(x) \equiv \exists x \neg P(x)$
  - $\neg\exists x P(x) \equiv \forall x \neg P(x)$

Conjunctive Normal Form (CNF)

Standard format required for the **Resolution** algorithm.

1. **Literal**: An atomic formula (P) or its negation ($\neg P$).
2. **Clause**: A disjunction ($\vee$) of literals.
    - Example: $(A \vee B \vee \neg C)$
3. **CNF**: A conjunction ($\wedge$) of clauses.
    - Example: $(A \vee B) \wedge (\neg B \vee C \vee D) \wedge (\neg A)$

Clausal Form: Often represented as a set of sets:
$\{\{A, B\}, \{\neg B, C, D\}, \{\neg A\}\}$

Proofs and Logical Consequence

- **Logical Consequence**: $\Delta \models A$ (A follows from set Δ).
- **Deduction Theorem**: $\{A_1, \dots, A_n\} \models B$ iff $(A_1 \wedge \cdots \wedge A_n) \Rightarrow B$ is a valid (tautology).

Proof by Refutation (Reductio ad Absurdum) To prove $\Delta \models A$:

1. Assume the negation of the goal: $\Delta \cup \{\neg A\}$.
2. Show that this set is **unsatisfiable** (leads to a contradiction/false).
3. If $\Delta \wedge \neg A$ is impossible, then $\Delta \Rightarrow A$ must be true.

Inference: The Resolution Rule

Resolution is a sound inference rule that generalizes “Modus Ponens”.

Propositional Resolution:

$$\frac{A \vee B, \quad \neg B \vee C}{\vdash} A \vee C$$

- **Logic**: If B is True, C must be True (from the second clause). If B is False, A must be True (from the first clause). Therefore, either A or C is True.
- **Step**: Resolving $(A \vee B)$ and $(\neg B \vee C)$ cancels out the complementary literals B and $\neg B$.

Completeness: Resolution is **refutation-complete**. It may not derive every true sentence directly, but it can always confirm a contradiction if the KB is unsatisfiable.

Unification i

In FOL, clauses contain variables. To apply resolution, we must match predicates by finding a substitution θ that makes terms identical.

- **Substitution**: $\theta = \{x/\text{Alice}, y/\text{Bob}\}$
- **Unifiable**: Two terms T_1, T_2 are unifiable if there exists θ such that $T_1\theta = T_2\theta$.
- **MGU (Most General Unifier)**: The simplest substitution that solves the unification.

Unification ii

Examples:

1. Unify `Knows(John, x)` and `Knows(y, Mother(y))`?
   - $\theta = \{y/\text{John}, x/\text{Mother(John)}\}$
   - Result: `Knows(John, Mother(John))`
2. Unify `King(x)` and `Greedy(y)`?
   - **Fail**: Predicates differ.

Combines Propositional Resolution with Unification.

Rule: Given clauses C_1 and C_2:

- $C_1 = l_1 \vee \cdots \vee l_k$
- $C_2 = m_1 \vee \cdots \vee m_n$
- If l_i and $\neg m_j$ unify with substitution θ:

$$\text{Resolvent} = SUBST(\theta, (C_1 - \{l_i\}) \cup (C_2 - \{m_j\}))$$

Example:

1. C_1: $\neg$Man(x) $\vee$ Mortal(x) (All men are mortal)
2. C_2: Man$($Socrates$)$
3. Unify $\neg$Man(x) and Man$($Socrates$)$:
   $\theta = \{x/$Socrates$\}$.
4. Resolvent: Mortal$($Socrates$)$.

Horn Clauses

A restricted subset of logic used in **Logic Programming (Prolog)** allowing efficient linear-time inference.

- **Definition**: A clause with **at most one positive literal**.
- **Forms**:
  1. **Rule**: $\neg P_1 \lor \neg P_2 \lor H \equiv (P_1 \land P_2) \Rightarrow H$.
     - Prolog: `H :- P1, P2.`
  2. **Fact**: H (no negative literals).
     - Prolog: `H.`
  3. **Goal**: $\neg P_1 \lor \neg P_2$ (no positive literal).
     - Prolog: `:- P1, P2.`
- **Inference**: Supports **Forward Chaining** (Data-driven) and **Backward Chaining** (Goal-driven).