1 First-Order Logic

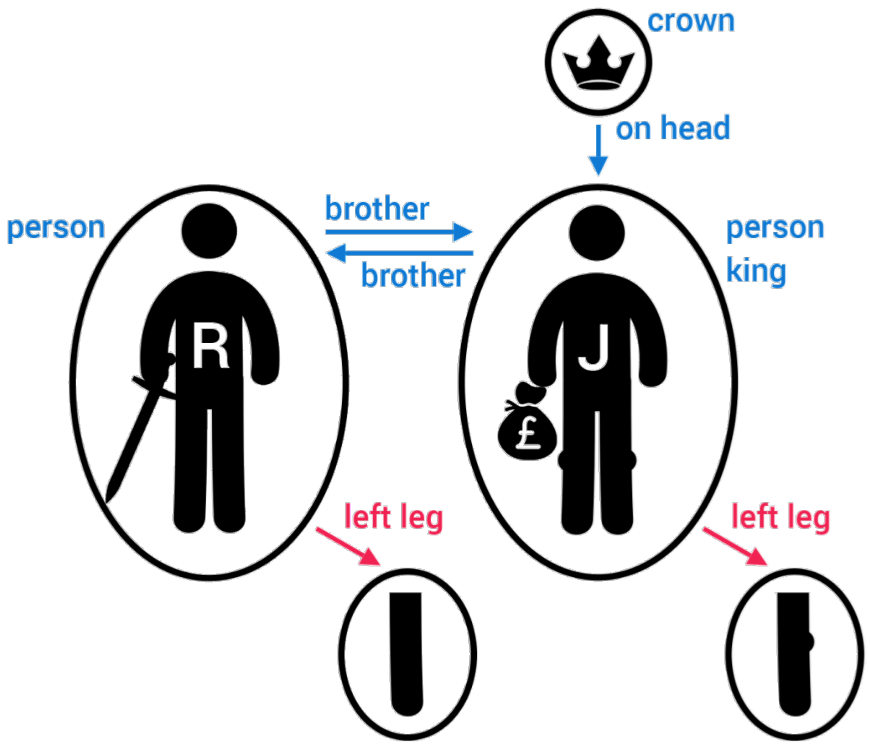
1.1 Limitations of Propositional Logic

Propositional Logic allows us to capture knowledge about the world using atomic facts and logical connectives. However, it suffers from major limitations regarding expressiveness:

- It deals with **atomic facts** only.
- It has no notion of **objects** or **relations** between them.
- Terms have no formal meaning; meaning is only intuitive to the reader (e.g., *RoommateCarryingUmbrella* is just a string "P" to the computer).

1.2 First-Order Logic (FOL)

First-Order Logic extends propositional logic by introducing the notion of objects, variables, and quantifiers. This allows us to generalize rules over entire classes of objects rather than writing rules for every specific instance.



Key Elements of FOL Syntax

- **Objects**: Specific entities in the world. (e.g., $John$, $Earth$, $Umbrella_0$).
- **Relations (Predicates)**: properties that hold true or false.
  - **Unary Relations**: Properties of a single object (e.g., $IsWet(x)$, $Person(y)$).
  - **n-ary Relations**: Relationships between multiple objects (e.g., $Carrying(John, Umbrella_0)$).
- **Functions**: A mapping that refers to an object without needing a specific name. Unlike predicates, functions do **not** return true/false; they return an **object**.
  - Example: $LeftLegOf(John)$ refers to a specific leg object.
  - Functions allow encoding data structures (e.g., $succ(0)$, $succ(succ(0))$ for integers).
- **Equality**: $Roommate(Person_0) = Person_1$.

1.3 Quantifiers

Quantifiers allow us to talk about sets of objects rather than individuals.

1.3.1 Universal Quantifier (∀)

"For all..." or "For every...".

- Syntax: $\forall x : \text{Formula}(x)$.
- Typically used with **Implication** (⇒).
- **Example:** "All lions are cats."

$$\forall x : Lion(x) \Rightarrow Cat(x)$$

- **Common Mistake:** Using AND ($\wedge$) with $\forall$. $\forall x : Lion(x) \wedge Cat(x)$ would mean "Everything in the universe is both a lion and a cat."

1.3.2 Existential Quantifier (∃)

"There exists..." or "For some...".

- Syntax: $\exists x : \text{Formula}(x)$.
- Typically used with **Conjunction** ($\wedge$).
- **Example:** "There is a cat that is not a lion."

$$\exists x : Cat(x) \wedge \neg Lion(x)$$

- **Common Mistake:** Using Implication (⇒) with $\exists$. $\exists x : Lion(x) \Rightarrow Cat(x)$ is trivially true if there is even one object in the universe that is not a lion (because False ⇒ True is valid).

1.3.3 Duality of Quantifiers

Universal and existential quantifiers are related via negation:

- "All x are P" is equivalent to "There is no x that is not P".

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

- "There exists an x that is P" is equivalent to "It is not the case that for all x, P is false".

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

1.3.4 Translating English to FOL

- "John has Jane's umbrella": $Has(John, Umbrella(Jane))$ (Using function).
- "John has an umbrella": $\exists y : (Has(John, y) \wedge IsUmbrella(y))$.
- "Any person who has an umbrella is not wet":

$$\forall x : (IsPerson(x) \Rightarrow ((\exists y : (Has(x, y) \wedge IsUmbrella(y))) \Rightarrow \neg IsWet(x)))$$

1.4 Inference and Substitution

1.4.1 Substitution (SUBST)

To perform inference (reasoning), we need to make general rules apply to specific objects. **Substitution** replaces variables with specific terms (constants or functions).

- Notation: $SUBST(\{x/John\}, IsHealthy(x))$
- Result: $IsHealthy(John)$

1.4.2 Generalized Modus Ponens

This is the workhorse of FOL inference. It combines substitution with the standard Modus Ponens rule.

- **Premise 1:** $\forall x : Loves(John, x)$ (John loves everything).
- **Premise 2:** $\forall y : Loves(y, Jane) \Rightarrow FeelsAppreciatedBy(Jane, y)$.
- **Unification:** We find a substitution that makes the "Loves" parts match.
- Substitution $\theta = \{x/Jane, y/John\}$.
- **Conclusion:** $FeelsAppreciatedBy(Jane, John)$.

1.5 Normal Forms and Skolemization

To use automated theorem proving (like Resolution), sentences must be converted into **Conjunctive Normal Form (CNF)**.

Algorithm: Converting to First-Order CNF

This is a critical procedure for the exam.

1. **Eliminate Implications:** Replace $A \Rightarrow B$ with $\neg A \vee B$.
2. **Move Negations Inwards (NNF):** Apply De Morgan's laws and $\neg \forall x P \equiv \exists x \neg P$. Negations should only appear immediately before predicates.
3. **Standardize Variables:** Rename variables so that no two quantifiers use the same variable name (e.g., $\forall x P(x) \vee \forall x Q(x)$ becomes $\forall x P(x) \vee \forall y Q(y)$).
4. **Skolemization** (Eliminate Existential Quantifiers):
   - If $\exists$ occurs outside all $\forall$: Replace variable with a **Skolem Constant**.
   - If $\exists$ occurs inside a $\forall x$: The existential variable depends on x. Replace it with a **Skolem Function** $f(x)$.
   - *Example:* $\forall x \exists y : Loves(x, y) \rightarrow \forall x : Loves(x, f(x))$. (Everyone loves someone $\rightarrow$ Everyone loves their specific "loved one").
5. **Drop Universal Quantifiers:** Assuming all remaining variables are universally quantified.
6. **Distribute OR over AND:** To get the conjunction of disjunctions.
7. **Create Clauses:** Separate into a set of clauses.

1.6 Resolution

Resolution is a refutation proof procedure. To prove a sentence P, we add $\neg P$ to the knowledge base and try to derive a contradiction (empty clause).

- **Input:** Two clauses containing complementary literals (e.g., $P(x)$ and $\neg P(y)$).
- **Process:**
  1. **Unify** the complementary literals. Find a substitution θ such that $P(x)\theta = P(y)\theta$.
  2. Apply θ to the remaining literals in both clauses.
  3. Combine the remaining literals into a new clause (the resolvent).
- **Completeness:** Resolution is refutation-complete. If a set of clauses is unsatisfiable, resolution will find a contradiction.

1.7 Prolog and Logic Programming

1.7.1 Horn Clauses

Prolog is based on a subset of FOL called **Horn Clauses**. A Horn clause is a disjunction of literals with **at most one positive literal**.

- Can be written as an implication: $Head \leftarrow Body_1, Body_2, \ldots$
- In Prolog syntax: `Head :- Body1, Body2.`
- "Head is true IF Body1 AND Body2 are true."

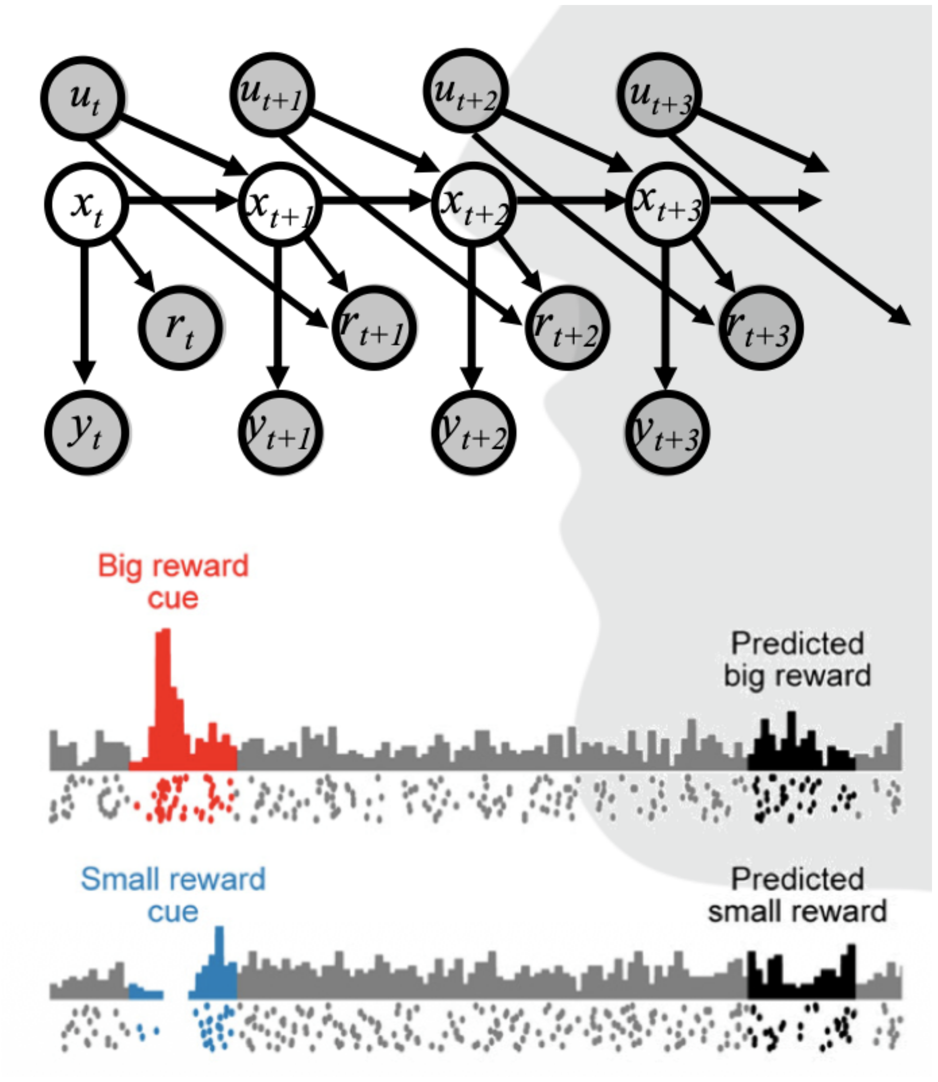
1.7.2 Prolog Syntax

- **Facts**: `eats(sam, dal).` (Note: atoms are lowercase).
- **Variables**: Start with Uppercase (e.g., `Person`, `X`).
- **Rules**: `compatible(P1, P2) :- eats(P1, Food), eats(P2, Food).`

1.7.3 Prolog Inference

Prolog uses **Backward Chaining** with **Depth-First Search (DFS)**.

1. Start with the query (goal).
2. Try to unify the goal with the head of a rule in the database (top-down).
3. If matched, the body of the rule becomes the new sub-goal.
4. If a path fails, **Backtrack** to the last choice point.



1.8 Theoretical Limits

1.8.1 Decidability

- **Semidecidable:** Inference in FOL is semidecidable.
- If a sentence is entailed, an algorithm exists that will eventually prove it.
- If a sentence is **not** entailed, the algorithm might run forever (halting problem). We cannot always conclude "False".

1.8.2 Gödel's Incompleteness Theorem

- FOL is not rich enough to fully model basic arithmetic (specifically mathematical induction) while remaining complete.
- For any consistent formal system capable of expressing arithmetic, there exist true statements that **cannot be proved** within the system.
- Implication: We cannot prove every truth in the universe using a single formal system.

1.8.3 Higher-Order Logic

FOL cannot quantify over predicates or relations.

- FOL: $\forall x P(x)$ (Valid).

- Higher-Order: $\forall P P(John)$ ("John has all properties"). This is not allowed in FOL.

1.9 Neuro-Symbolic AI (Current Trends)

There is a debate on how to combine the reasoning power of logic with the learning power of neural networks.

- **Symbols**: Good for general intelligent action, reasoning, interpretability (Newell & Simon).
- **Neurons**: Learning from data, handling noise, high-dimensional perception (Hinton).
- **Neuro-Symbolic**: Approaches that use neural networks to perceive (e.g., identifying objects in an image) and logic to reason about them (e.g., "If X is close to Y..."").
- **Example Approach:** Differentiable Forward Reasoning, where logic rules are treated as weighted components in a differentiable loss function, allowing backpropagation through logic.