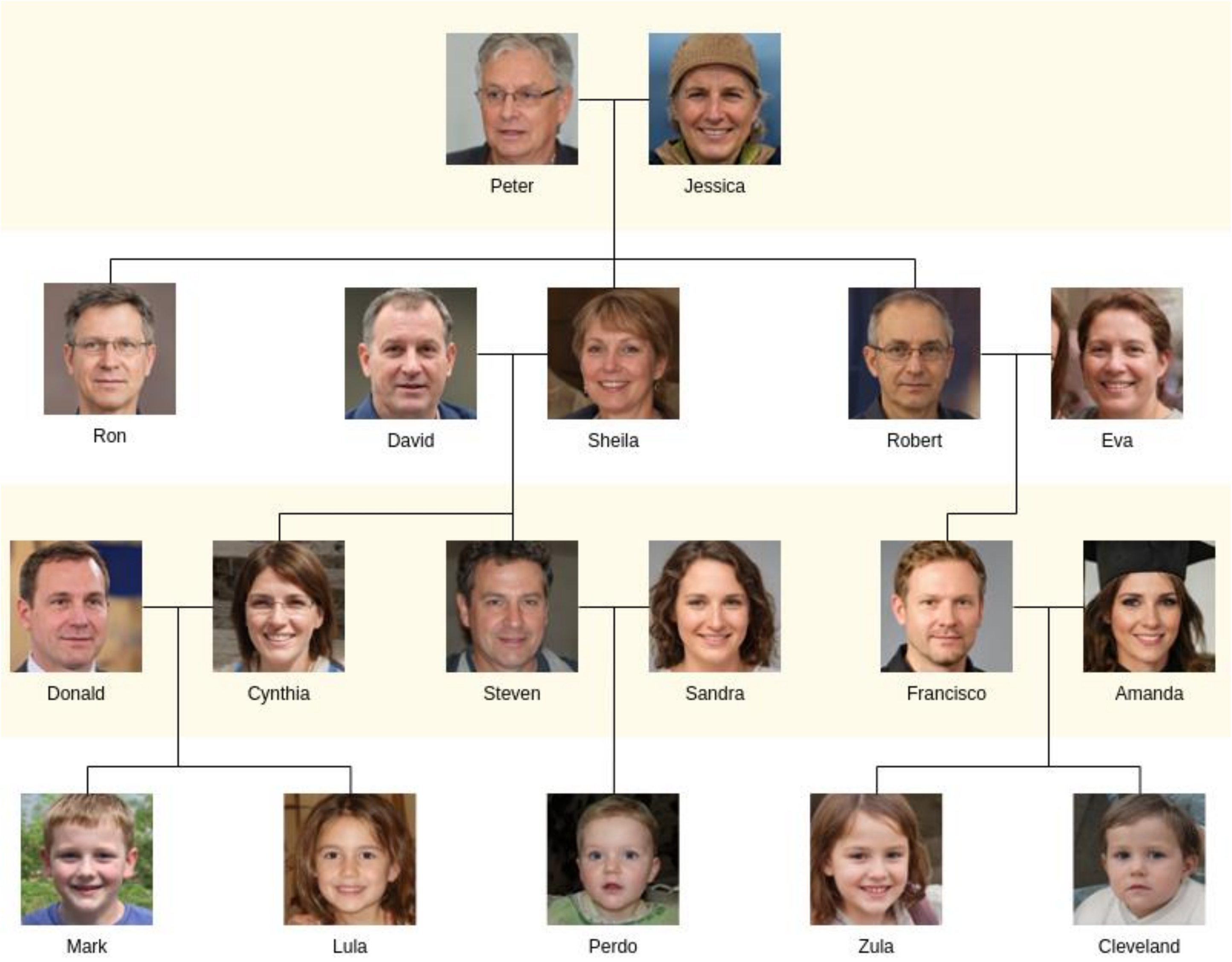
CAP 379
Artificial Intelligence Lab

Tanzeela Javid Kaloo (32638)
Assistant Professor
System And Architecture
Lovely Professional University

Lab 02
Predicate Calculus

Tanzeela Javid Kaloo (32638)
Assistant Professor
System And Architecture
Lovely Professional University

Predicate Calculus: Family Tree Example in Prolog



Limitations of Propositional Logic

- **Static Nature**:
  - Propositional logic requires explicit enumeration of all relationships.
  - Dynamic relationships (e.g., "Find the nth ancestor") are cumbersome.
- **No Variables for Quantifiers**:
  - Statements like "All descendants of Peter" require predicate logic, not propositional logic.
- **Limited Expressiveness**:
  - Cannot generalize rules for complex reasoning (e.g., inheritance of traits or roles).

Predicate Logic

- To overcome these limitations, predicate logic introduces:
  - **Variables** for dynamic queries.
  - **Quantifiers** (`forall`, `exists`) for general rules.
  - **Advanced Reasoning** for more intelligent systems.

Predicate Calculus

- **Propositional Calculus:** Uses atomic symbols (e.g., P, Q) to represent entire propositions. Lacks the ability to describe components of propositions.
- **Predicate Calculus:** Allows representation of relationships between objects and properties, using predicates and variables.

Features

- **Predicates:**
  - Represents a relationship between objects.
  - Example: `weather(tuesday, rain)`.
- **Variables:**
  - Generalize assertions about classes of objects.
  - Example: `∀X (weather (X, rain))` → "It rains every day."
- **Inference Rules:**
  - Access and manipulate predicate calculus expressions.

Syntax of Predicate Calculus

- **Symbols:**
  - Alphabet: Letters (`A-Z, a-z`), Digits (`0-9`), Underscore (`_`).
  - Symbols start with a letter and may contain legal characters.
- **Types of Symbols:**
  - **Constants:** Specific objects, properties (lowercase). Example: george, blue.
  - **Variables:** Represent classes of objects (uppercase). Example: X, Day.
  - **Functions:** Map elements from a domain to a range (lowercase). Example: father(david).
  - **Predicates:** Define relationships. Example: likes(george, kate).
- **Reserved Truth Symbols:**
  - true, false.

Well-Formed Expressions

- **Constants:** Start with a lowercase letter.
  - Example: `blue, rain.`
- **Variables:** Start with an uppercase letter.
  - Example: `X, Day.`
- **Function Expressions:** Function symbol followed by arguments.
  - Example: `father(david), price(bananas).`
- **Atomic Sentences:** Predicate followed by arguments (arity matters).
  - Example: `likes(george, kate), friends(bill, george).`

Evaluation

- Replacing a function with its value is called **evaluation**.
  - **Example**: If `father(david) = george`, then `friends(father(david), allen)` evaluates to `friends(george, allen)`.

Definitions

- **Terms:** Constants, variables, or function expressions.
  - Example: `X, mother(sarah), cat.`
- **Predicates:** Define relations of arity n.
  - Example: `likes(X, Y), friends(bill, george).`

Examples

- **Predicate with constants:** `likes(george, kate).`
- **Predicate with variables:** `friends(X, Y).`
- **Functions as arguments:** `friends(father(david), father(andrew)).`

Core Concepts

- **Predicate Symbols and Atomic Sentences**:
  - Predicate symbols begin with lowercase letters.
  - Predicates have an **arity**, which defines the number of arguments (e.g., `mother(eve, abel)` has arity 2).
  - An **atomic sentence** is formed by applying a predicate to the correct number of terms enclosed in parentheses and separated by commas.
  - **Truth values** `(true, false)` are also considered atomic sentences.
- **Logical Connectives**:
  - The connectives `∧ (and), ∨ (or), ¬ (not), → (implies), and ≡ (equivalent)` are used to combine atomic sentences into more complex expressions.

Core Concepts

- **Quantifiers**:
  - **Universal Quantifier** (∀): Indicates that a sentence applies to all elements in the domain.
    - Example: `∀ X likes(X, ice_cream)` means "everyone likes ice cream."
  - **Existential Quantifier** (∃): Indicates that a sentence applies to at least one element in the domain.
    - Example: `∃ Y friends(Y, peter)` means "someone is a friend of Peter."
- **Defining Complex Sentences**: Sentences in predicate calculus can be built recursively:
  - Base: Atomic sentences.
  - Negation: If s is a sentence, so is ¬s.
  - Conjunction/Disjunction: If `s1` and `s2` are sentences, so are `s1 ∧ s2` and `s1 ∨ s2`.
  - Implication/Equivalence: If `s1` and `s2` are sentences, so are `s1 → s2` and `s1 ≡ s2`.
  - Quantification: If X is a variable and s a sentence, ∀X s and ∃X s are also sentences.

Example: Biblical Genealogy

Using predicates to describe relationships:

- **Atomic predicates:**
  - `mother(eve, abel)`
  - `mother(eve, cain)`
  - `father(adam, abel)`
  - `father(adam, cain)`

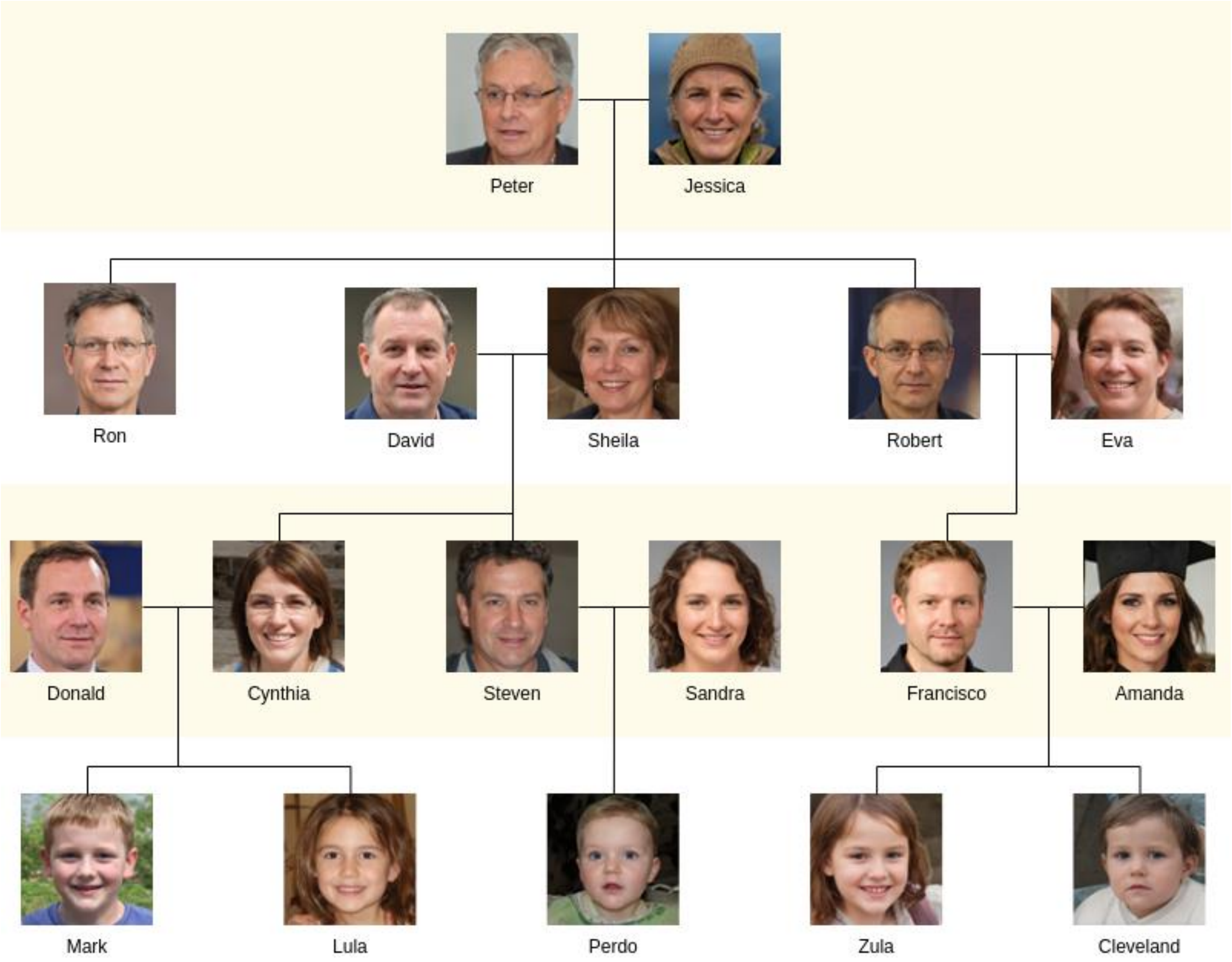
- **Derived relationships:**
  - Parent definition: $\forall X \forall Y (father(X, Y) \lor mother(X, Y) \rightarrow parent(X, Y))$
  - Sibling definition: $\forall X \forall Y \forall Z (parent(X, Y) \land parent(X, Z) \rightarrow sibling(Y, Z))$
  - Inference: From these rules, we can deduce `sibling(cain, abel)`.

Well-formedness and Examples

- **Valid Atomic Sentences**:
  - `equal(plus(two, three), five)`
  - `equal(plus(2, 3), seven)` (though this is false under standard arithmetic, it is still syntactically valid).
- **Complex Sentences**:
  - `∃ X foo(X, two, plus(two, three)) ∧ equal(plus(two, three), five)`
  - `(foo(two, two, plus(two, three))) → (equal(plus(three, two), five) ≡ true)`

Back to Example

Filename: family-tree_predicate_logic.pl



Step 1: Facts

- Define individuals
  - individual(peter).
- Define parent-child relationships
  - parent(peter, ron).
- Define gender
  - male(peter).

Step 2: Define Rules

- Grandparent Relationships
- Sibling Relationships
- Ancestor Relationships
- Descendant Relationships
- Cousin Relationships
- Intelligent Inference

Step 2: Define Rules

- Grandparent Relationships

  grandparent(X, Z) :- parent(X, Y), parent(Y, Z).
- Sibling Relationships

  sibling(X, Y) :- parent(P, X), parent(P, Y), X \= Y.
- Ancestor Relationships

  ancestor(X, Y) :- parent(X, Y).

  ancestor(X, Y) :- parent(X, Z), ancestor(Z, Y).
- Descendant Relationships

  descendant(X, Y) :- ancestor(Y, X)
- Cousin Relationships

  cousin(X,Y) :- parent(P, X), parent(P, Y), X \= Y.

  cousin(X, Y) :- grandparent(Z, X), grandparent(Z, Y), X \= Y.

Step 2: Define Rules

- Intelligent Inference
  - Check if X and Y are related
    - related(X, Y) :- ancestor(Z, X), ancestor(Z, Y).
- Universal Quantifier (∀)
- Existential Quantifier (∃)

Step 3: Queries

- Who are the children of Peter?
  - parent(peter, X).
- Who are the siblings of Ron?
  - sibling(ron, X).
- Who are Peter's grandchildren?
  - grandparent(peter, X).