CAP 379
Artificial Intelligence
Lab

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

Lab 01
Prolog Fundamentals
Propositional Calculus

Tanzeela Javid Kaloo (32638)

Assistant Professor

System And Architecture

Lovely Professional University

What is Prolog?

- **Definition**:
  - Prolog stands for "Programming in Logic."
  - It is a logic programming language based on formal logic.
- **Features**:
  - Declarative paradigm.
  - Designed for symbolic computation and reasoning.
- **Use Cases**:
  - Artificial Intelligence.
  - Knowledge-based systems.
  - Natural Language Processing.

Prolog's Unique Approach

- **How Prolog Differs**:
  - **Imperative Languages** (e.g., Python, C): Focus on *how* to do tasks.
  - **Prolog**: Focuses on *what* is true or logically consistent.
- **Components**:
  - **Facts**: Represent truths.
  - **Rules**: Define relationships and infer new truths.
  - **Queries**: Ask questions about the facts and rules.

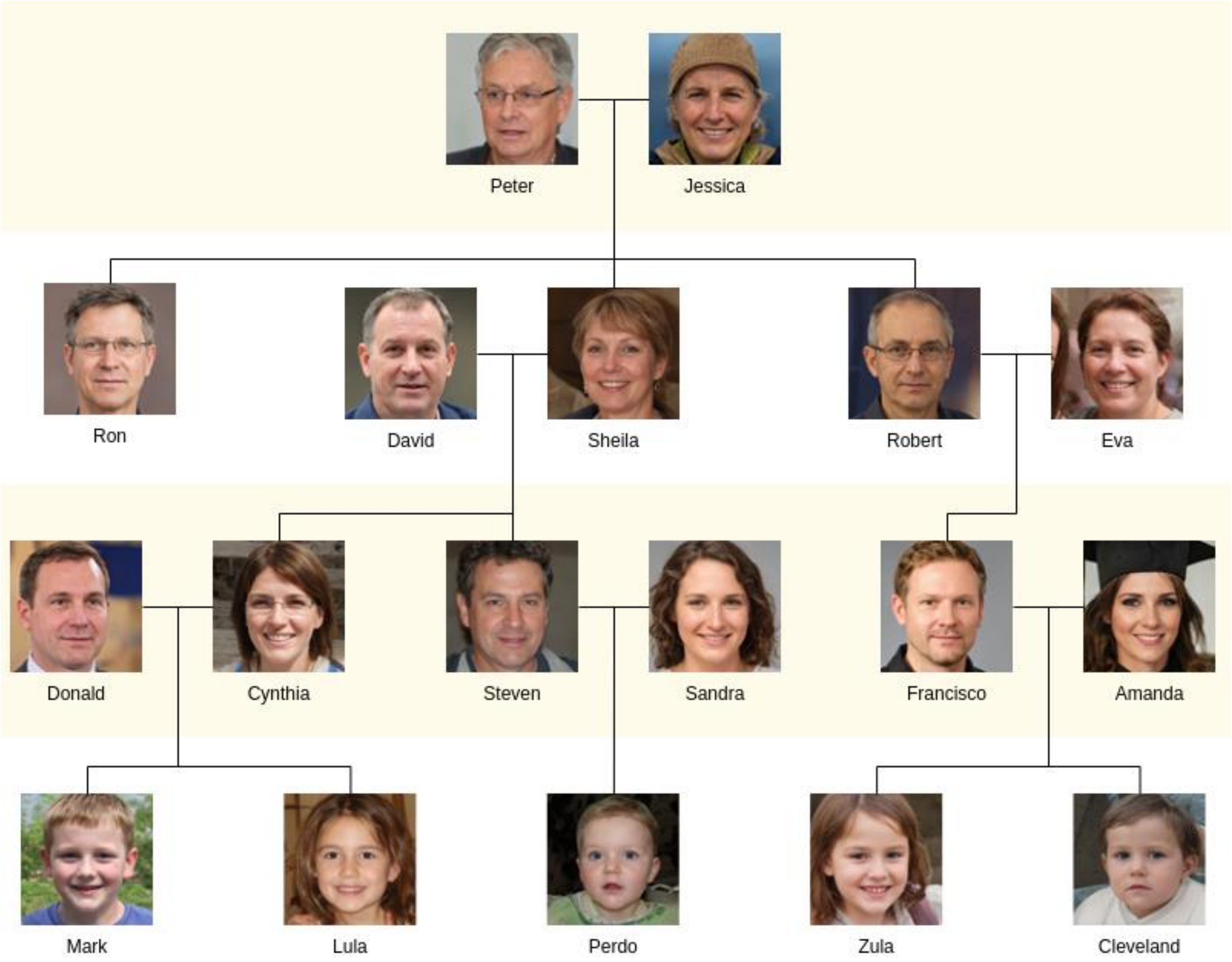
Applications of Prolog

- **AI and Expert Systems**:
  - Building knowledge bases.
  - Implementing reasoning engines.
- **Natural Language Processing**:
  - Syntax analysis.
  - Semantic understanding.
- **Problem Solving**:
  - Solving puzzles, games, and logical problems.
- **Database Systems**:
  - Querying structured information using logic.

Setting Up Prolog

- **Environment Setup**:
  - Recommended Software: SWI-Prolog, GNU Prolog.
- Installation:
  - Download from the official website.
  - Follow the installation instructions.
- **Prolog Console**:
  - `?-` indicates the Prolog prompt.

Propositional Calculus: Family Tree Example in Prolog



Propositional Calculus

- Proposition
  - It is Raining
  - 2+2=6
- Propositional Symbols: P, Q, R, S, ...
- Truth Symbols: true, false

Logical Connectives

- **Negation** (¬): The negation of a proposition P is denoted as ¬P, meaning "not P".
  - Example: ¬P (if P is true, ¬P is false).
- **Conjunction** (∧): Represents "and". P∧Q is true if both P and Q are true.
  - Example: P∧Q (both P and Q must be true).
- **Disjunction** (∨): Represents "or". P∨Q is true if at least one of P or Q is true.
  - Example: P∨Q (either P or Q or both are true).
- **Implication** (→): Represents "if... then...". P→Q is false only when P is true and Q is false.
  - Example: "If it rains, then the ground will be wet" (P→Q).
- **Biconditional** (↔): Represents "if and only if". P↔Q is true when both P and Q are either both true or both false.
  - Example: "You will get an A if and only if you study" (P↔Q).

Propositional Calculus Sentences

- Every propositional and truth symbol is a sentence.
- Negation ($\neg P$), conjunction ($P \wedge Q$), disjunction ($P \vee Q$), implication ($P \rightarrow Q$), and equivalence ($P \vee Q \equiv R$) are all valid sentences.
- Sentences are also called well-formed formulas (WFFs).
- Examples of components:
  - $P \wedge Q$: Conjuncts are P and Q.
  - $P \rightarrow Q$: Premise is P, conclusion is Q.
- Parentheses and brackets control the order of evaluation (e.g., $(P \vee Q) \equiv R$ vs $P \vee (Q \equiv R)$).

Truth Assignment Rules

- **Negation (¬P)**: True if P is false, false if P is true.
- **Conjunction (P ∧ Q)**: True if both P and Q are true; otherwise, false.
- **Disjunction (P ∨ Q)**: False only when both P and Q are false; otherwise, true.
- **Implication (P → Q)**: False only when P is true and Q is false; otherwise, true.
- **Equivalence (P ≡ Q)**: True if both expressions have the same truth value in all interpretations.

Logical Identities

- **Examples of Logical Identities**:
  - $\neg(\neg P) \equiv P$ (Double negation)
  - $P \rightarrow Q \equiv \neg P \vee Q$ (implication is equivalent to "not P or Q").
  - $P \rightarrow Q \equiv \neg Q \rightarrow \neg P$ (contrapositive law)
  - $(\neg P \rightarrow Q) \equiv (P \vee Q)$ (Contrapositive law)
  - de Morgan's Laws: $\neg (P \vee Q) \equiv (\neg P \wedge \neg Q)$, $\neg (P \wedge Q) \equiv (\neg P \vee \neg Q)$
  - Commutative: $(P \wedge Q) \equiv (Q \wedge P)$
  - Associative: $((P \wedge Q) \wedge R) \equiv (P \wedge (Q \wedge R))$
  - Distributive: $P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$

Focuses on *what* is true or logically consistent.

Consistency: A set of propositions is **consistent** if there is at least one interpretation where all the propositions are true

Rules

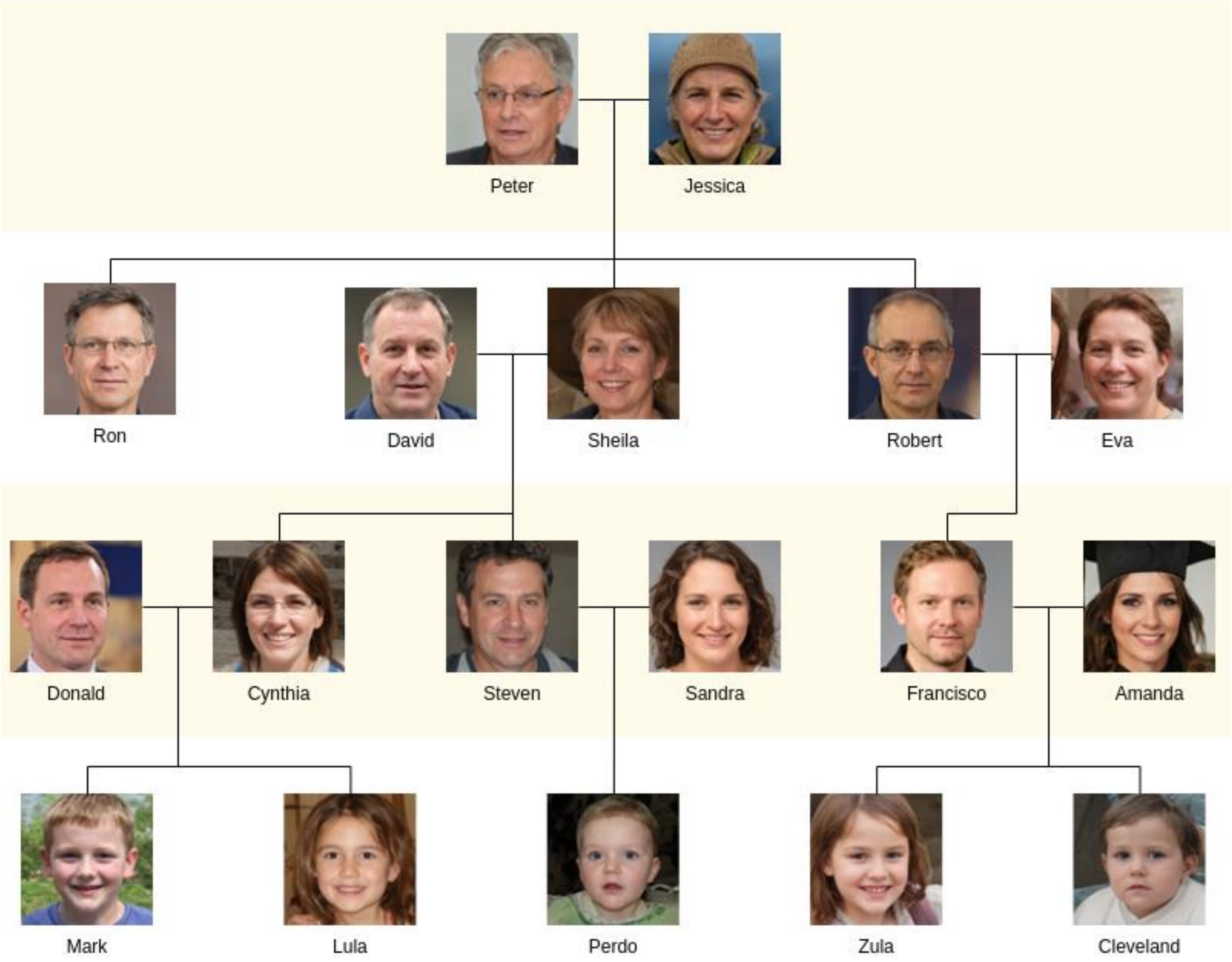
- **Rules of Inference**: Including rules like
  - **Modus Ponens:** From $P \rightarrow Q$ and P, infer Q (if $P \rightarrow Q$ and P then Q) and
  - **Modus Tollens:** $P \rightarrow Q$ and $\neg Q$, infer $\neg P$. (if $P \rightarrow Q$ and $\neg Q$ then $\neg P$).
  - **Disjunctive Syllogism**: From $P \vee Q$ and $\neg P$, infer Q.
  - **Hypothetical Syllogism**: From $P \rightarrow Q$ and $Q \rightarrow R$, infer $P \rightarrow R$.

Validity of Arguments

- An **argument** in propositional logic consists of premises and a conclusion.
- The argument is **valid** if, whenever all the premises are true, the conclusion must be true.

Back to Example

Filename: family-tree_propositional_logic.pl



Propositional Logic Steps

- **Step 1: Define Facts**
  - Use atomic propositions to represent parent-child relationships.
  - Explicitly enumerate all relationships (e.g., parent, sibling, grandparent, cousin).

Propositional Logic Steps

- **Step 2: Encode Relationships**
- **Parent-Child Relationships**: Each parent-child relationship is defined as a separate proposition:
  - `parent_peter_ron`, `parent_peter_david`, `parent_peter_sheila`, etc.
- **Sibling Relationships**:
  - Siblings are defined explicitly for all pairs:
    - `sibling_ron_david`, `sibling_ron_sheila`, etc.
- **Grandparent Relationships**:
  - Grandparents are defined explicitly by extending parent-child facts:
    - `grandparent_peter_mark`, `grandparent_jessica_lula`.
- **Cousins**:
  - Cousins are defined explicitly for all valid pairs:
    - `cousin_mark_lula`, `cousin_zula_cleveland`.

Propositional Logic Steps

- **Step 3: Query Relationships**
  - To find relationships, query explicitly defined propositions.

Step 1: Facts

- Parent-Child Relationships
  - parent_peter_ron.
  - parent_steven_perdo

Step 2: Define Rules

- Derived Relationships
  - Sibling Relationships
    - sibling_ron_david.
  - Grandparent Relationships
    - grandparent_jessica_mark.
  - Cousin Relationships
    - cousin_mark_lula.
    - cousin_mark_perdo.

Step 3: Queries

- Who are Peter’s children?
    - parent_peter_X.
- Are Mark and Lula cousins?
    - cousin_mark_lula.
- Who are Peter’s grandchildren?
    - grandparent_peter_X.

Limitations of Propositional Logic

- **Redundancy**:
  - Every relationship must be explicitly defined (e.g., all sibling and cousin relationships).
- **Scalability**:
  - Adding a new member to the family tree requires updating multiple facts and derived relationships.
- **Lack of Generalization**:
  - Cannot infer relationships dynamically (e.g., "Who are all of Peter's descendants?").
- **Relational Complexity**:
  - Cannot handle indirect or recursive relationships easily.