Yachay Tech University

Mathematical and Computational Logic

Prolog Lab 2: Building a Simple Expert System

1. Lab Objective

In this lab, students will design and implement a simple expert system in Prolog. The system will ask questions (yes/no or multiple-choice), and based on answers, it will infer a conclusion (diagnosis, recommendation, or classification).

Example domains include:
- Animal identification
- Tech troubleshooting (e.g., diagnosing a computer problem)
- Food recommendation system

This lab uses animal identification as the guiding example. However, each person should take a different topic.

2. Background Concepts

- Knowledge Base: Facts and rules that describe the domain.
- Inference: Prolog's ability to use backtracking to derive answers.
- User Interaction: Using write/1, nl/0, and read/1 for simple input/output.

3. Step-by-Step Lab Tasks

Step 1: Knowledge Base

Define facts about animals and their properties:

```
has_fur(cat).
has_fur(dog).
lays_eggs(chicken).
lays_eggs(duck).
barks(dog).
meows(cat).
```

Step 2: Rules for Classification

Add rules to deduce what an animal could be:

```
is_mammal(X) :- has_fur(X).
is_bird(X) :- lays_eggs(X).
```

Query examples:

```
?- is_mammal(cat).
true.

?- is_bird(dog).
false.
```

Step 3: Interactive Questions

Ask the user questions and record answers:

```
ask(Question, Answer) :-
  write(Question), write(' (yes/no): '), nl,
  read(Answer).
```

Step 4: Inference Engine

Use input to guess the animal:

```
identify_animal(Animal) :-
  ask('Does it have fur?', Fur),
  (Fur == yes ->
    ask('Does it bark?', Bark),
    (Bark == yes -> Animal = dog ; Animal = cat)
  ;
    ask('Does it lay eggs?', Eggs),
    (Eggs == yes -> Animal = chicken ; Animal = duck)
  ),
  write('I think the animal is: '), write(Animal), nl.
```

Step 5: Extend the System

Students should expand the knowledge base and rules, here we show the examples in the case of the animals but find the corresponding extensions for your system:

- Add more animals (horse, eagle, penguin).
- Add more properties (flies, swims, domesticated).
- Handle ambiguity (multiple possible answers).
- Improve interaction (give the user a list of candidates instead of a single guess).

4. Exercises

1. Basic Run: Test the program with the default animals (dog, cat, chicken, duck).
2. Knowledge Expansion: Add at least 5 more animals and their properties.
3. Generalization: Create can_fly/1, can_swim/1, etc. and rewrite some rules.
4. Ambiguity Handling: Modify the system so it can suggest multiple possible animals if facts are insufficient.
   Example:
   possible_animals(List) :- findall(A, matches(A), List).
5. Challenge: Add a recursive rule for "ancestor" type reasoning (e.g., classification tree: vertebrate → mammal → dog).

6. Deliverables

- A .pl file with:
  - Facts for at least 10 animals
  - Rules for classification
  - An interactive procedure (identify_animal/1)
- Example queries and their results

All deliverables should be stored in the corresponding directory on your GitHub account.