Islamic University of Technology

Artificial Intelligence Lab

CSE 4712

Lab 3

Author:
Ishmam Tashdeed
CSE, IUT

Contents

1 Introduction to Prolog

In this lab, you will be introduced to Prolog, a *declarative logic programming language* commonly used in AI for modeling relationships, search problems, and constraint-satisfaction problems (CSPs). Unlike Python, where you tell the computer what to do step by step, in Prolog you describe what is true, and Prolog searches for values that satisfy those truths. Think of Prolog as a smart database and a search engine. We will use an online compiler for writing Prolog.

1.1 Facts

Facts in Prolog are simple statements that are `true`. These are used to model anything in the world.

```
animal(cat).
animal(dog).
animal(mouse).
animal(butterfly).

plant(cactus).
plant(rosemary).
plant(mango).
plant(pumpkin).
```

In this example, I tried modeling some animals and plants. You can have complicated facts such as:

```
age(max, 25).
age(norris, 21).
age(oscar, 22).
```

Showing a fact regarding a person and their age. **Note:** In Prolog, variables start with a **capital letter** *e.g.* Age, X, Xy.

1.2 Queries

Queries in Prolog are a way of asking questions based on the facts that have been already defined. Asking:

```
age(max, 25).
```

Should return `true`. You could query to return all the facts that match a certain pattern by using a variable:

```
age(X, 25).
```

This will list the names whose age is 25 and the output should be `X=max`.

1.3 Rules

Rules express conditional truth having a head and a body. Rules are like defining Python functions, but without return statements, as if it only returns `true` or `false`.

```
% Fact: John likes pizza
likes(john, pizza).

% Rule: Someone is happy if they like pizza
happy(X) :- likes(X, pizza).
```

The body consists of one or more goals (predicates) separated by commas (,) for conjunction (AND) or semicolons (;) for disjunction (OR). Each goal in the body must be proven true for the head to be true. Every Prolog clause (fact or rule) must end with a full stop. Rules often employ recursion to define relationships that involve repeated application of a condition, such as defining ancestor relationships or list processing.

2 Tasks

2.1 Task 1

Model a small world using atomic facts. Create a Prolog knowledge base containing facts about family relationships:

```
parent(alice, bob).
parent(bob, charlie).
parent(alice, diana).
parent(diana, eric).
```

Tasks:

- Add more facts of your own following the given relationships such as:
  - `sibling`: Show who are siblings from the given relationships
  - `gender`: Define the gender of each person from the relationships
  - `age`: Define the age of each person from the relationships

- Query the facts:
  - Who are Alice's children?
  - Who are Bob's parents?
  - List all people with the age of 30.
  - List all the men.

2.2 Task 2

Use rules to define relationships that aren't explicitly stored such as `grandparent`, `grandson`, `granddaughter`. Then query:

- Who are Charlie's grandparents?
- Is Alice a grandparent of Eric?
- List the grandsons of Alice.

2.3 Task 3

Add the following facts to a file:

```
color(red).
color(green).
color(blue).
```

Define a rule where you will check if the two colors that are being tested are different or not. Using that rule, list all combinations of two distinct colors from the given set of colors.

2.4 Task 4

Introduce a recursive definition named `count` to find the total number of elements in a given list. **Sample Input:**

```
count([1, 2, 4, 1, 3], N)
```

Sample Output:

```
N=5
```