CT331 Programming Paradigms Week 9 – Lecture 2

Dr. Finlay Smith
Room 430, IT Building
Finlay.smith@UniversityofGalway.ie

Introduction to Prolog

Prolog

- Used to solve problems involving objects, and relationships between objects.

- The basics:
  - Facts
  - Questions
  - Variables
  - Conjunctions
  - Rules

- Declaring Facts about objects and their relationships.
- Defining Rules about objects and their relationships.
- Asking Questions about objects and their relationships.

Prolog Syntax

- A program is a list/database of *clauses*.
- These clauses can be:
  - Facts
  - Relations
  - Rules
- The logic database is queried.

- **Program** can be thought of as a storehouse of facts and rules.
- **Conversational Language**: The user can ask questions about the set of facts and rules in the Prolog program.

Prolog Abstractions

- **Data Structure***:* List
- **Control***:*

    Recursion.

    Ordering of clauses.

    Built-in control facilities:

        cut operator, - should be avoided

        fail, not.

Features of Prolog (Clocksin and Mellish)

- Several dialects
- Syntax relatively easy but writing efficient Prolog programs is not so easy
- Prolog performs a task in response to a question (query) from the programmer
- A question provides a *conjunction of goals* to be satisfied
- Prolog uses known clauses in the database to try satisfy the goals.

Facts and Backtracking

- A fact can cause a goal to be satisfied immediately whereas a rule can only reduce the task to that of satisfying a conjunction of sub goals.
- A clause can only be used if it matches the goal under consideration.
- If a goal cannot be satisfied, backtracking will occur.
- Backtracking consists of reviewing what has been done and attempting to re-satisfy the goals by finding the *alternative way* to satisfying them.
- Prolog attempts to satisfy the goals in a conjunction in a left to right order/top down manner.

Simple Facts

- Statements which are true in a given knowledge base
- In Prolog we can make some statements by using facts.
  - Particular item
  - A relation between items.

- We can represent the fact that it is sunny by writing the program:

```
sunny.
```

- We can now ask a query of Prolog by asking

```
?- sunny.
```

- **?-** is the Prolog prompt.
  - To this query, Prolog will answer yes. sunny is **true**
  - Prolog matches it in its database of facts.

Syntax Rules for Facts

- Begin with a **lowercase** letter.
- End with a full stop.
- Any letter or number combination
  - …and underscore _ character.
- Names containing the characters -, +, *, /, or other mathematical operators should be avoided.

Logic database 1:
sunny.
happy.
this_is_fun.

Examples

```
joe_gymnast.            /* Joe is a gymnast */
mary_is_confused.       /* Mary is confused*/
2happy_today.           /* Incorrect syntax for fact */
foggy.                  /* It is foggy */
Ed_is_lost!!.           /* Incorrect syntax for fact*/
```

- ?- raining.
- no

- ?- foggy.
- yes

Facts with Arguments

- More complicated facts consist of a **relation** and the items that this refers to.
  - These items are called **arguments**.
  - Facts can have arbitrary number of arguments (zero upwards)
- A general model is shown below:

```
relation(<argument1>,<argument2>,....,<argumentN> ).
```

- Relation names must begin with a **lowercase** letter
  - likes(bill, cake).

Facts with Arguments

owns(joe,computer).

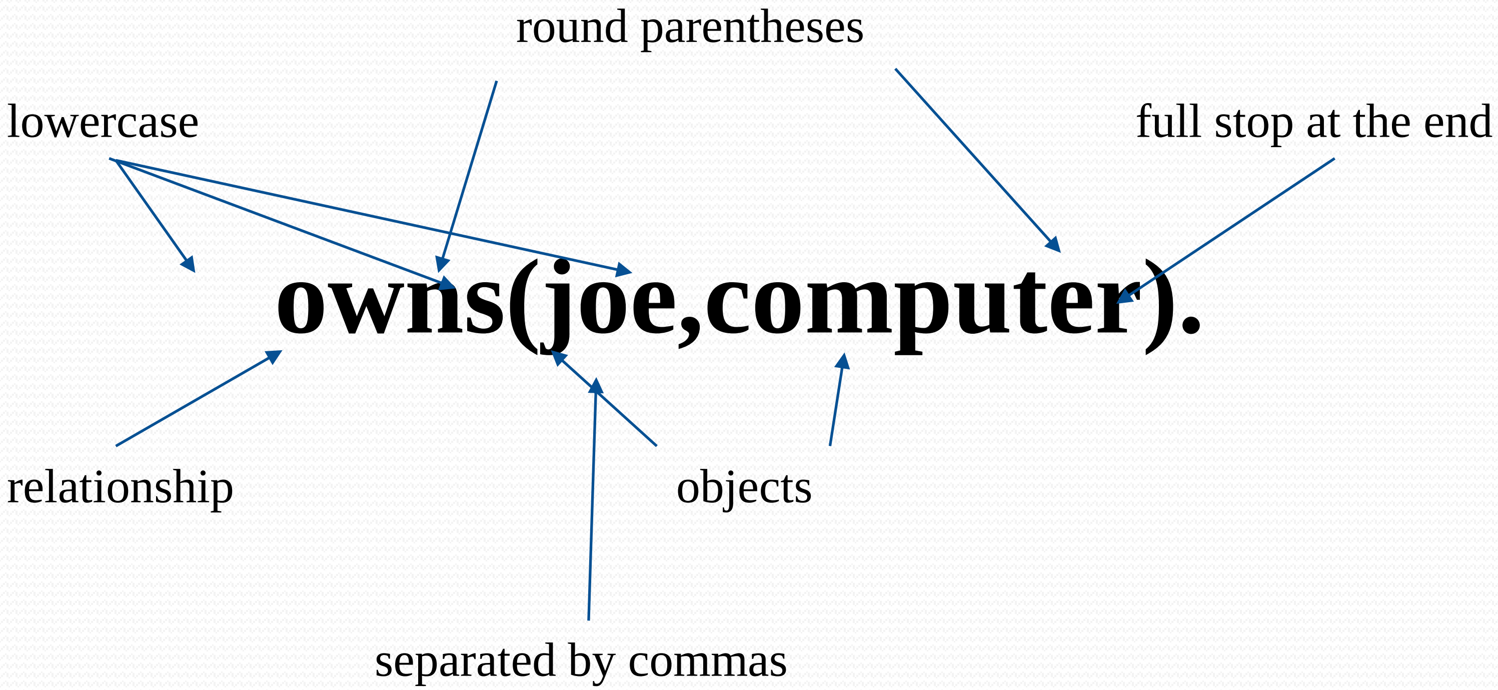
- "*Joe owns the computer*"

- **Relationship**: ownership
- **Objects**: Joe, Computer

- **Directional**: Joe owns the computer

  Not: The computer owns Joe

Parts of a Fact



round parentheses

lowercase

full stop at the end

owns(joe,computer).

relationship

objects

separated by commas

Relations

- Used to declare “world” of relations
- Arguments may be
  - Instantiated variable
  - Un-instantiated variables

Database 2:

```
happy(ted).

sunny.

likes(ted, sun).

likes(ted, beer).

likes(ted, beach).
```

Querying the Database

- The database contains the facts from which the questions are answered.

- A Question can look exactly like a fact:

  likes(ted,sun).

- The difference is in which **mode** one is in…

Querying the Database

- *Interactive question mode* is indicated by:
  - Question mark and dash **?-**
- Question example: ?- likes(ted,sun).
- Meaning:
  - If ted is interpreted as a person called Ted, and sun is interpreted as the sun, then:
  - ?- likes(ted,sun). means: Does Ted like the sun?

Variables and Unification

- When querying a database:
  - In order to match arguments we must use a **variable**.
  - The process of matching items with variables is known as **unification**.
  - Variables are distinguished by starting with a capital letter.

- Examples;
  - X /* Begins with capital letter */
  - VaRiAbLe /* Can be made up of either case of letters */
  - My_name /* we can link words together via '_') */

Querying Database 2

- ?- likes(ted, sun).
  - yes
- ?- likes(ted, holidays).
  - no
- What does ted like?
  - ?- likes(ted, X). /* X is an **un-instantiated** variable */
  - What will the result be?

Database 2:

```
happy(ted).
sunny.
likes(ted, sun).
likes(ted, beer).
likes(ted, beach).
```

Results

- X = sun ;
- X = beer ;
- X = beach ;
- no

Why did we receive these results?

?- likes(ted, X).

- 2nd argument, *X*, is un-instantiated and may match anything, provided *ted* is first argument.

Database searched from top to bottom.

- First match in database is *X* = *sun*, i.e. likes(ted, sun).
- *sun* is output
- The place of this clause in the database is marked so that X won't be instantiated to *sun* again on subsequent searches
- Backtracking occurs when we ask Prolog to keep searching (;) to see if there are any more matches ..
- X is un-instantiated again and the search begins from after the marking in database … thus next match is *X* = *beer* . *beer* is output and database is marked.
- Backtracking occurs when we ask Prolog to keep searching (;) to see if there are any more matches ..
- X is un-instantiated again and the search begins from after the marking in database … thus next match is *X* = *beach* . *beach* is output and database is marked.
- Backtracking occurs when we ask Prolog to keep searching (;) to see if there are any more matches ..
- No more matches and have checked everything so Prolog outputs *no*

Place Marker

- The first match is found: X=sun.
- The user acknowledges.
- From that place on the next match is found
  - The search continues.
- If at the place of the last instantiation no more match is found: The answer will be: **no**.

- Try:

  ?- likes(Y, beach).

Database
happy(ted).
sunny.
likes(mary,beach).
likes(ted,beer).
likes(ted,beach).

- Write out all steps as in previous example

Conjunction

- A conjunction between the two terms will result in the whole expression to evaluate to *true* if both terms evaluate to true. If either or both terms in the expression evaluate to false, the whole expression evaluates to false.
- The word "*conjunction*" is used mainly in the context of logic and logic programming
    - It is equivalent to an "AND" in Java, C++..

Conjunction Example

Database

```
likes(mary,food).
likes(mary,wine).
likes(john,wine).
likes(john,mary).
```

- In Prolog, a comma means a conjunction:

  `?- likes(john,mary), likes(mary,john).`

- Answer: no
- A match for likes(john,mary)
- No match for likes(mary,john)

Conjunction using Variables

- Is there anything that both mary and john like?
- Find out what Mary likes and then see if John likes it:

```
?- likes(mary,X), likes(john,X).
```

Classification of Prolog

- A language for programming in Logic
- Relational
- Descriptive
- Declarative