COMP210: Artificial Intelligence

Lecture 3. Prolog: An AI programming language

Trevor Bench-Capon
Room 215, Ashton Building

http://www.csc.liv.ac.uk/~tbc/COMP210

Introduction to Prolog

- Last Time
  - Overview of AI applications, techniques, state of the art and brief history
- Today
  - a brief introduction to Prolog and its uses;
  - describe the use of facts and rules in Prolog programs;
  - explain how Prolog evaluates queries.

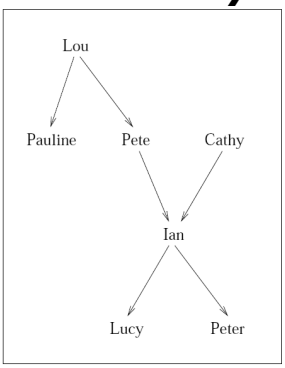
Prolog

- short for Programming in logic;
- based on an idea to use logic as a programming language (we'll see the connection later);
- developed in the 1970's in Edinburgh and Marseilles;
- often thought of as an AI programming language
- useful for solving problems involving objects and relations between them;
- not suitable for problems involving a lot of numeric calculation, or GUI.
- "Deductive Database"

Books

- *Prolog Programming for Artificial Intelligence,* Ivan Bratko, third edition, Addison Wesley.
- *The art of Prolog: advanced programming techniques*, Leon Sterling, Ehud Shapiro, MIT Press 1986
- *Introduction to programming in Prolog* Danny Crookes Prentice-Hall,1988
- SWI Prolog Users Manual available at
  - http://www.swiprolog.org/pldoc/index.html
- SWI Prolog web site
  - http://www.swi-prolog.org/index.txt

Example: A Family Tree



Facts

- To represent the above family tree in Prolog we use *facts.*
  - parent(pete,ian).
  - parent(ian,peter).
  - parent(ian,lucy).
  - etc.
- Note the full stop at the end of every line
- Relation name starts with a lower case letter as do its arguments(constants or atoms). Upper Case is for variables.
- Compare a database record: table *parent*, with attributes for *parent* and *child*

More Facts

- We could store information about the gender of people in the family tree as follows.
  - female(cathy).
  - female(lucy).
  - male(ian).
  - etc.
- Any kind of Facts we want:
  - lives_in(lucy,walford).
- Any number of arguments:
  - lived_from(lou,1890,1992).

The Family Tree Program

```
/* Family Tree */

parent(pete,ian).        % Pete is a parent of Ian
parent(ian,peter).
parent(ian,lucy).
parent(lou,pete).
parent(lou,pauline).
parent(cathy,ian).

female(cathy).           % Cathy is female.
female(lucy).
female(pauline).
female(lou).

male(ian).               % Ian is male.
male(pete).
male(peter).
```

Comment symbols

Relations grouped together

Comments

- As with all programming languages it is a good idea to
- include comments.
- In Prolog there are two ways to do this.
  - % Ian is male.
  - % is a comment on this line only.
- Extended comment

  /*
  Family Tree
  Describes family relationships
  */
  Where
  - /* denotes the beginning of the comment and
  - */ denotes the end of the comment.

Querying the Family Tree Program

- Assume the above facts have been stored as a Prolog program and loaded into a Prolog interpreter. A Prolog query is a clause, just like the facts. Each query is terminated by a full stop.
  - Is Ian a parent of Lucy?

    ?- parent(ian,lucy).

    yes
  - Is Ian a parent of Pauline?

    ?- parent(ian,pauline).

    no
  - Is Cathy a parent of Peggy? *Note Peggy does not appear at all in our program.*

    ?- parent(cathy,peggy).

    no

Querying the Family Tree Program

- Can use variables as well as checking facts
- Who is Lucy's parent?

  ?- parent(X,lucy).

  X = ian

  yes
- Note variables begin with upper case letters (here X).

Querying the Family Tree Program

- If there are several answers, the next solution can be requested by typing a semi-colon. When no further solutions exist Prolog answers 'no'.
- Who are Lou's children?

  ?- parent(lou,X).

  X = pete ? ;

  X = pauline ? ;

  no

Querying a program: summary

- Arguments to relations can be. . .
  - concrete objects, known as *atoms eg* ian, or cathy (starting with a lower case letter),
  - or *variables eg* Father or Child (starting with an upper case letter).
- Questions to the system consist of one or more *goals.*
  - If a positive answer to the question is found we say the goal was satisfiable and the goal succeeded.
  - If we get a negative answer the goal was unsatisfiable and it failed.

Rules

- Rules specify things that are true if some condition is satisfied.
- We can use the parent and female relations to define more interesting relationships.
- Mothers are female parents.
  - mother(X,Y):- parent(X,Y),female(X).
- For all X and Y, X is the mother of Y, if X is the parent of Y and X is female.
- The left side of the rule (the conclusion part) is known as the head of the clause. The right side of the rule (the condition part) is known as the body of the clause.

Evaluating Rules

- Is Cathy the mother of Ian?

  ?- mother(cathy,ian).

  yes
- There are no facts about mothers in the program so we must use the rule about mothers.
  - mother(X,Y):- parent(X,Y),female(X).

How It Works

```
/* Family Tree */

parent(pete,ian).
parent(ian,peter).
parent(ian,lucy).
parent(lou,pete).
parent(lou,pauline).
parent(cathy,ian).

female(cathy).
female(lucy).
female(pauline).
female(lou).

male(ian).
male(pete).
male(peter).
```

mother(X,Y):-parent(X,Y),female(X).

Considers the goals in order: First parent(X,Y)

Finds a matching clause, binding variables if necessary

X= pete, Y= ian

Binding is applied to other clauses **mother(pete,ian):- parent(pete,ian),female(pete)**. Then tries other clauses

female(pete) fails.

Backtracks: resatisifes first clause

female(ian) fails twice

female(lou) succeeds

mother(X,Y) succeeds with X = lou, Y= pete

More queries

```
/* Family Tree */

parent(pete,ian).
parent(ian,peter).
parent(ian,lucy).
parent(lou,pete).
parent(lou,pauline).
parent(cathy,ian).

female(cathy).
female(lucy).
female(pauline).
female(lou).

male(ian).
male(pete).
male(peter).
```

Can instantiate a variable:
?- mother(cathy,Child)

Will succeed:
Matches parent(cathy,ian)
Binds Child to ian and matches female(cathy)
Child = ian

?- mother(pete,Y) fails: binds Y to ian, but female(pete) fails

?- mother(lou,pauline) succeeds

?- mother(lucy,Y) fails: cannot match parent (lucy,Y).

?- mother(Parent,ian) succeeds with Parent = cathy
(but will try pete first)

More Rules

- Grandparents are parents of parents.
  - grandparent(X,Z):- parent(X,Y),parent(Y,Z).
- Siblings are people with the same parent.
  - sibling(X,Y):- parent(Z,X),parent(Z,Y).
  - sister(X,Y):-
    female(Y), parent(Z,X),parent(Z,Y).

  But: sibling(ian,ian) will succeed! distinct variables may bind to same atom. Need to add X \= Y (X not equal to Y).
- Can use relations defined in rules in other rules:
  - grandmother(X,Z):- mother(X,Y),parent(Y,Z).
  - sister(X,Y):-sibling(X,Y),female(Y).

Questions

- What relationship does the following describe?
  - unknown(X,Y):- parent(X,Y),female(Y).
- How would we define brother?
- What does the question grandparent(X,Y) ask?
  - What answers will Prolog find?

Running Prolog

- On a unix/linux system enter the word pl

```
$ pl
Welcome to SWI-Prolog (Multi-threaded, 32 bits, Version
   5.7.11)
Copyright (c) 1990-2009 University of Amsterdam.
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This
   is free software, and you are welcome to redistribute it
   under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?-
```

Loading programs

- Create the program (facts and rules) using any text editor
- If the program is in the same directory, type

```
?- ['family.tree.pl'].
%family.tree.pl compiled 0.00 sec, 1,920 bytes
Yes
?-
```

Now you can type in your queries

To exit the progam type

```
halt.
```

Summary

- A Prolog program is constructed from facts and rules.
- Facts describe things that are true without conditions
  - like data in a database.
- Rules describe things that hold depending on certain conditions. The conditions are defined in rules and facts
  - Ultimately all rules unfold to facts.
- Prolog programs can be queried using questions.
- Prolog clauses are facts, rules and questions.
- Queries are answered by solving each goal in turn by matching rules and facts, instantiating variables, where needed

Next Time

- We will go back to AI
- We will look at Search problems