Declarative Programming
CS-205
Part II: Logic Programming (Prolog)

Monika Seisenberger, Ulrich Berger

Department of Computer Science
College of Science
Swansea University

CS-205 - Week 7

Course Aims (Reminder)

1. Students will be able to specify and write programs in functional and logic programming languages.
2. They will be able to develop solutions to simple algorithmic problems using declarative rather than procedural concepts.

Introduction

This module provides an introduction to functional and logic programming paradigms and gives students the opportunity to gain practical experience in using both.

Syllabus:

- Logic Programming in Prolog:
    - The essence of logic programming.
    - Pattern matching, recursion, backtracking (and resolution).
    - Database programming
    - (Extralogical aspects of Prolog.)
    - Data structure terms and lists.

Learn Prolog Now!

The text for this part of module is *Learn Prolog Now!*

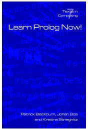


Figure 1: Learn Prolog Now!

Will expect cover each week chapters:

Week 7	Chapt 1: *Facts, Rules, and Queries*
Week 8	Chapt 2: *Matching and Proof Search*
Week 9	Chapt 3: *Recursion*, Chapt 4: *Lists*, Chapt 5: *Arithmetic*
Week 10	Advanced Topics.
Week 11	Revision

Book available online (`www.learnprolognow.org`); but worth buying.

Use either SWI Prolog or Sicstus Prolog. Both is available in the labs.

Declarative Programming:

- Say *what* you want done; not *how* you want it done
- Contrast Imperative/Procedural Programming (C++, Java)

Two main forms of declarative programming:

- Logic Programming – main language Prolog
  - Set of rules and facts to be satisfied
- Functional programming – LISP, Scheme, Racket, ML and Haskell
  - Program defined by a set of function definitions
- Both paradigms based on sound mathematical foundations
  - Subset of first order logic and recursive function theory (resp.)

Introduction

There are three basic constructs in Prolog:

- Facts
- Rules
- Queries

”A collection of facts and rules is called a knowledge base (or a database) and Prolog Programming is all about writing knowledge bases. Prolog programs simply are knowledge bases, collections of facts and rules which describe some collection of relationships that we find interesting.”

Other concepts,

- the role of logic
- unification with the help of variables
- Some Prolog syntax defining terms, atoms, and variables

First example: Knowledge base 1

```
%kb1 - this is a comment
```

```
woman(mia).
woman(jody).
woman(yolanda).
playsAirGuitar(jody).
party.
```

Discussion of various queries and results in the lecture. What is the difference between queries ?- woman(mia). and ?-woman(X).

Knowledge base 2

```
%kb2

happy(yolanda).

playsAirGuitar(mia) :- listensToMusic(mia).

playsAirGuitar(yolanda) :- listensToMusic(yolanda).

listensToMusic(mia).

listensToMusic(yolanda):- happy(yolanda).
```

Try to answer: Who plays the air guitar?

Knowledge base 3

```
happy(vincent).
listensToMusic(butch).

playsAirGuitar(vincent):-
        listensToMusic(vincent),happy(vincent).
playsAirGuitar(butch):-
        happy(butch).
playsAirGuitar(butch):-
        listensToMusic(butch).
```

Knowledge base 4

```
woman(mia).
woman(jody).
woman(yolanda).

loves(vincent,mia).
loves(marcellus,mia).

loves(pumpkin,honey_bunny).
loves(honey_bunny,pumpkin).

% jealous(X,Y):- loves(X,Z),loves(Y,Z).
```

Some History of Prolog

1972 First Prolog interpreter by Colmerauer and Roussel

1973 SLD Resolution defined by Kowalski

1977 Implementation of DEC10 compiler by Warren

1980 Definite Clause Grammars implementation by Pereira and Warren

1980s/90s Prolog grows in popularity especially in Europe and Japan

2005 Prolog used to program natural language interface in International Space Station by NASA

2011 Prolog used in Watson to win *Jeopardy! Man vs. Machine Challenge*

IBM’s Watson Program

Jeopardy

- Jeopardy, Quiz game on US TV, given an answer, try to find th cessing (software mainly written in Prolog) and links to internet.
- *Watson* beat best human players in contest in 2012!

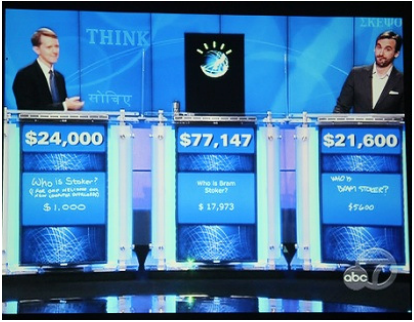
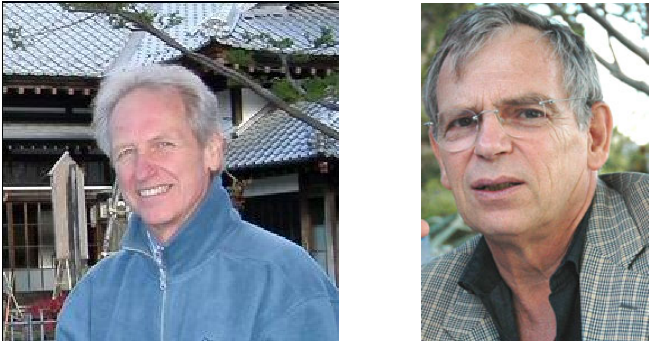


Figure 2: Watson playing Jeopardy against experts

Two Pioneers



(a) Robert Kowalski

(b) Alain Colmerauer

Figure 3: Founders of Prolog/LP

Sicstus Prolog

We'll use Sicstus Prolog for execution of programs

- Available cross platform - see BB for details of obtaining compiler
- Run in the Eclipse environment with SPIDER plug-in
- On line manual for Sicstus 4.2.3 - worth downloading pdf, but DON'T print out (1400 pages)
- Many libraries provided
- Code can run as interpreted or compiled
- Can link with Java using PrologBeans or Jasper

Some Prolog Syntax

Atoms

1. A sequence of characters of upper-case letters, lower-case letters, digits, or underscore, *starting with a lowercase letter*
   Examples: `bill`,`lecture_one`, `classOne`
2. An arbitrary sequence of characters enclosed in single quotes
   Examples: `'Bill'`,`'Chapter One'`,`'@#'`
3. A sequence of special characters
   Examples: `:   , ; .   :- =>`

Numbers

1. Integers: 12, -34, 22342
2. Floats: 34573.3234

Variables

- A sequence of characters of upper-case letters, lower-case letters, digits, or underscore, starting with either an uppercase letter or an underscore
  Examples: `X`, `Y`, `Temp`, `Person`, `_num`

Some Prolog Syntax - Terms

Terms - basic data structure of Prolog

Terms are built up (defined inductively) from

1. Basic terms - atoms, numbers and variables
2. A functor applied directly to a sequence of terms
   So if `f` is a functor and `t1`, `t2`, ..., `tn` are terms then
   `f(t1,t2,..,tn)` is a term
   Examples: `man(bill)`, `f(a,X,g(b,c))`,
   `father(father(bill))`

- A functor must be an atom
- The *arity* of `f` is the number of arguments it takes
- An n-ary functor is denoted f/n (usually used when a predicate)
  Example: in above `man/1`, `f/3` and `father/1`

Terms

- Terms are extremely flexible data structures and are essentially flat representations of trees
- Everything in Prolog is essentially a term (including program clauses) but some functors use infix notation (e.g. $:-$)
- In Prolog you can define two predicates with the same functor but with different arity
- Prolog would treat this as two different predicates
  Example: `append/2` and `append/3` are different