Learn Prolog through Lectures, Text book and Practice

- Lectures:
  - You will be introduced the basics
  - Then we’ll teach you Prolog by examples
- Reading: Bratko Part 1
  - This is where you learn the language the formal way
- Lab sessions
- Exercises and assignments
  - Practice, Practice, Practice!

Course Schedule

- Basics of Prolog Programming (3 weeks)
  - Syntax by example, List manipulation
- Search and control in Prolog (2 weeks)
  - Implementing basic search methods
- AI Applications of Prolog (5 weeks)
  - Introduction to AI applications
  - Examples of Prolog implementations

Programming Languages for AI

- AI requires *symbolic computation*
  - Which is awkward in procedural languages such as Java, C, C++, Pascal, etc.
- Logic and functional languages better suited
- Logic languages
  - best known being Prolog, based on First Order Predicate Calculus
- Functional languages
  - AI people used LISP, based on λ-Calculus

What is PROLOG

- PROgramming with LOGic
  - Based on “resolution” in “first order logic”
- Aim: concentrate on your logic and write it down
  - The logic that you’ve written down is your program!
- Need a different way of thinking
  - Recursion is norm
    (Think of it in terms of mathematical induction)
- When mastered:
  - Fast prototyping, Easy to debug and modify
  - Used to be slow, but
    - boosted by constraints technology
**AI Applications to cover**
- Search
- Simple Agents – bargaining
- Natural Language – simple parsing
- Machine Learning – Classes learning
- Constraint Satisfaction – Forward Checking
- AI Planning – Simplified “STRIPS” planner

**Prolog**

**Prolog Basics**

**What you should know about Prolog**
- 95% of your errors will be typing errors, as:
  - Prolog is case sensitive
  - There is no need to declare variables
  - All variables are local (scope ended with “.”)
  - There are no assignments (x = 4)
    - Only matching/unification
  - There are no loops
    - Only recursion, which is the norm

**Basic Syntax of Prolog Programs**
- All programs are either facts or rules.
- Each clause ends with a full stop
- Start with capital == Variable
- Facts take the form “Functor(Arg1, ..., Argn).”
  - mother( mary, adrian ).
- Rules take the form “Head :- Body.”
  - father(X, Y) :- husband(X, Someone), mother(Someone, Y).

**Simple Prolog Program**

```
mother(mary, adrian).
mother(mary, jane).
mother(jane, richard).
husband(john, mary).
husband(tony, jane).
father(X, Y) :-
  husband(X, Woman),
  mother(Woman, Y).
```
- It’s up to you to define the interpretations before you program
- mother(X, Y) means “X is the mother of Y” here
- The rule defines one condition under which X is the father of Y

**Queries Answering in Prolog**

```
?- father(john, jane).
yes
?- father(john, Who).
Who = adrian
yes
?- female(mary).
No
```
- Prolog returns the first answer that it can find
  - Instantiating variables if needed
- Prolog answers by “walking a tree”
  - Essential to know
  - See textbook for details
- Prolog only returns answers according to the rules only
  - It answers “no” when it cannot prove something w.r.t. the facts and rules

Edward Tsang (all rights reserved)
Example: Family Tree

parent( pam, bob ).
parent( tom, bob ).
parent( tom, liz ).
parent( bob, ann ).
parent( bob, pat ).
parent( pat, jim ).

grandparent( X, Z ) :-
parent( X, Y ),
parent( Y, Z ).

Building up rules

parent(P, C) :- mother(P, C).
parent(P, C) :- father(P, C).
ancestor(Anc, Desc) :-
parent(Anc, Desc).
ancestor(Anc, Desc) :-
parent(Anc, X),
ancestor(X, Desc).

Assignments in Prolog

• Assignment is implemented via matching / unification

add1(X, X_plus_1) :-
X_plus_1 is X + 1.
?- add1( 8, What ).
What = 9;
/* ask for alternative answer */
no
/* no alternative answers */

Conditions in Prolog

• IF X > Y THEN Z = X ELSE Z = Y

max0( X, Y, Z ) :-
X > Y -> Z = X; Z = Y.
?- max0( 7, 9, X ).
X = 9
X = 7

Loops and Recursion

/* Factorial( N, Fact ) */

factorial( 0, 1 ).
factorial( N, F ) :-
N is N - 1,
factorial( N1, F1 ),
F is N * F1.

Factorial

fact1( 0, 1 ).
fact1( N, F ) :-
N > 0,
N1 is N - 1,
fact1( N1, F1 ),
F is N * F1.

/* ok with: */
?- factorial( -1, X ).
?- fact1( 0, 3 ).
/* but how about this: */
?- fact1( X, 6 ).

To handle that, fact1 must be modified
**No Global Variables**

- No global variables in Prolog
- To implement a constant, define a fact:

  ```prolog
  pi(2.1416).
  ```

  ```prolog
  circumference(Radius, Circ) :-
  pi(Pi),
  Circ is 2 * Radius * Pi.
  ```

  ```prolog
  ?- circumference(2, C).
  C = 8.5664
  ```

**Exercises: Common Errors**

/* would the following program work? why? */

```prolog
sum(X, Y, Sum) :-
  Sum = X + Y.
? - sum(2, 5, Sum).
```

```prolog
increment(A) :-
  A = A + 1.
? - increment(6).
```