BASIC PROLOG

Comp3031 Lab 09
Fall 2019

Chenyang LEI& Lipeng WANG

Introduction to Prolog

- Prolog is short for PROgramming in LOGic.
- A “Relational Programming” language.
- Conceived in early 1970s, first developed in 1972; still remains popular today.
- Widely used by AI researchers
  - Theorem proving
  - Expert systems
  - Games
  - Automated answering systems
  - Control systems

Get Started

- Type “swipl” at the prompt:

```
csl2wk01:sqiuac:109> swipl
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 6.6.6)
Copyright (c) 1990-2013 University of Amsterdam, VU 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).

?-
```

- Load file in Prolog
  - [filename]. %load “filename.pl”
  - [‘filename.xx’]. % load “filename.xx”
- Type in relations directly:
  - [user]. %Enter predicates as in a file
  - Press “Ctrl + D” to quite this mode

Example – family.pl

- Knowledge base in a Prolog program
  - Download it on canvas:

```
parent('John', 'James').% fact
parent('John', 'Mary'). % fact
parent('James', 'Judy'). % fact

age('John',60). % fact
age('James',35). % fact
age('Mary',30). % fact
age('Judy',9). % fact

male('John'). % fact
male('James'). % fact
female('Mary'). % fact
female('Judy'). % fact

sibling(X, Y) :- parent(_1,X), parent(_1,Y).      % rule
ancestor(X, Y) :- parent(X, Y).                   % rule
ancestor(X, Y) :- parent(_1, Y), ancestor(X, _1). % rule
```

Atom, Variable & Number

- Atom
  - ?- atom('John'). % true.
  - ?- atom(hongkong). % true.
  - ?- atom(_). % false.
- Variable
  - ?- var(_). % true.
  - ?- var(X). % true.
  - ?- var(Less). % true.
- Number
  - ?- number(1.2e24). % true.
  - ?- integer(23). % true.
  - ?- float(12.3). % true.

Facts and Rules

- Prolog programs describe relations by clauses
- Facts
  - Syntax:
    - <fact> := <functor>(<terms>). | <functor>.
    - <terms> := <term> |<terms>,<term>
    - <term> := <atom> | <variable> | <number> | <functor>(<terms>) | <functor>
  - <functor>(<terms>) are structures or compound terms.
  - Example
    - parent('John', 'James').
    - male('John').
    - parent(_1, X).

Facts and Rules

- Prolog programs describe relations by clauses
- Rules
  - Head :- Body.
  - Read as: “Head is true if Body is true”.
  - Body consists of calls to predicates.
  - The comma “,” is logical conjunction, meaning *and*.
  - The semicolon “;” can be used in the body for *or*.
  - The “\+” can be used in the body for *not*.

Predicates

- Structures are predicates.
- Built-in predicates in Prolog
  - Input/Output predicates: read, write, etc.
  - Control predicates:
    - X ; Y % X or Y
    - (X -> Y) %If X, then try Y, otherwise fail
    - true
  - Arithmetic predicates: +, *, is, etc.
  - Listing and debugging predicates: listing(p)

Predicates

- Predicates are relations (structures).
- Example:
  - Control predicates “X ; Y” can be taken as “; (X, Y)”.
- family.pl defines six predicates
  - parent/2
  - age/2
  - male/1
  - female/1
  - sibling/2
  - ancestor/2
- A prolog program is a collection of clauses (facts and rules); consists of predicates each of which ends with ".”.

Comparison Predicates

- Two types of comparisons in Prolog
- Term comparisons to compare the terms literally
  - ==, \==
- Arithmetic comparisons to compare the arithmetic values of the terms
  - =:=, =\=, <, =<, >, >=

Comparison Predicates Examples

- ?- monday == 'Monday'.
- false.
- ?- monday == 'monday'.
- true.
- ?- 2+1 == 3.
- false.
- ?- 2+1 =:= 3.
- true.
- ?- 2+1 \== 3.
- true.
- ?- 2+1 =\= 3.
- false.

Query

- Load the file “family.pl”

```
?- [family].
% family compiled 0.00 sec, 15 clauses
true.
```

- *Is John a male?*
  - ?- male('John').
  - true.
- *Who is James’ parent?*
  - ?- parent(X, 'James').
  - X = 'John'.
- *Who is John’s child?*
  - ?- parent('John', Y).
  - Y = 'James' ;
  - Y = 'Mary'.

- Prolog can produce all of the possible answers
  - If the user types a semicolon ';' , Prolog will look for a next answer
  - If the user just hits Enter, then Prolog stops looking for answers

Recursive Definition and Query

- A predicate is recursively defined if it refers to itself in the rule definition.
- Example
  - ancestor(X, Y) :- parent(_1, Y), ancestor(X, _1).
  - Query: *John is who's ancestor?*
    - ?- ancestor('John', X).
    - X = 'James' ;
    - X = 'Mary' ;
    - X = 'Judy' ;

List

- List is a recursive data structure in Prolog.
  - ?- [X|Y] = [monday, [2, 3, 4], X, []].
  - X = monday,
  - Y = [[2, 3, 4], monday, []].
- Relations on list
  - cons(X, Y, [X|Y]).
  - Query:
    - ?- cons([1], [2, 3], L).
    - L = [[1], 2, 3].

List

- Relations on list
  - len([], 0).
    len([_|T], N) :- len(T, X), N is X+1.
  - Recursive definition:
    - The empty list has length zero.
    - A non-empty list has length 1 + len (T), where len (T) is the length of its tail.
  - Query:
    - ?- len([[string, []], comp3031, []], N).
    - N = 3.

Exercise

- Modify the file “family.pl” in order to achieve the following goals
  - Modify the sibling rule so that 'James' will not be shown as an answer for **sibling(X, 'James')**
  - Write a query to list the people who are older than 30 in this family
  - Define a new relation **brother(X,Y)** where X and Y are siblings and X is male