Dynamic Prolog

Dr. Mattox Beckman

University of Illinois at Urbana-Champaign
Department of Computer Science

Objectives

You should be able to ...

You can often tell what the language designers thought about their language by the libraries that are included with it. Many of Prolog's involve the analysis of structures. In this lecture, we will go over some of the builtin predicates of Prolog.

- Use `findall` and `checklist` to perform queries over a range.
- Use `call` and `assert` to modify Prolog's database.
- Use deconstruction operations to examine data in Prolog.

Two Useful List Predicates

- `findall(X,T,Y)` finds all values of `X` that make `T` true, and puts them into `Y`.
- `checklist(P,Y)` is true if predicate `P` is true for all values in list `Y`.

```
?- findall(X,possible(X),Xs).
X = G306
Xs = [anna, beth, cindy, david, ernest, frank, gloria, harry]
?- checklist(student,[anna,beth]).
Yes
?- checklist(student,[anna,harry]).
No
```

How could you write functions like these?

Examine Thyself

One power that Prolog programs have is the ability to examine and modify themselves.

- Used for AI – real learning requires the ability to "examine yourself."
- Prolog structures and Prolog programs have the same form.
  - Assembly language: bit patterns
  - Scheme and Lisp: lists
  - Prolog: structures
- A language with this property is called *homoiconic*.

```
1 likes(john,mary).
2 ?- isbst(bst(5,null,null)).
```

Functors: `likes`, `isbst`, and `bst`.

Types of a Term

We have predicates that will determine the type of a term.

```
?- atom(3).
No
?- atom(hi).
Yes
?- atomic(3).
Yes
?- integer(3).
Yes
?- integer(f).
No
```

```
?- X = 20, integer(X).
X = 20
Yes
?- var(X).
Yes
?- X = 20, var(X).
No
```

Name

The `name` predicate turns a term into a string (and back).

```
?- name(foo,X).
X = [102, 111, 111]
?- name(X,"foo").
X = foo
chop(X,Y) :- name(X,[_|S]), name(Y,S).
?- chop(asymmetric,X).
X = symmetric
```

This will be very useful for natural language processing.

Look What You've Done!

- The `listing` predicate will print out the definitions we have so far.

```
?- listing(mortal).

mortal(A) :-
        human(A).

Yes
```

Accessing Parts of Functors

- `functor(T,F,N)` – `F` will contain the name of the functor, `N` will contain the number of arguments.
- `arg(N,T,A)` – `A` will be argument number `N` of `T`.

```
1 -? functor(isbst(5,null,null),F,N).
2 F = isbst
3 N = 3
4 -? arg(1,isbst(5,null,null),A).
5 A = 5
```

The =.. Operator

Another way to deconstruct terms is with "=..".

```
?- bst(5,null,null) =.. L.
L = [bst, 5, null, null] ;
?- L =.. [likes,john,X].
L = likes(john, _G276)
X = _G276
?- (mortal(X) :- human(X)) =.. L.
X = _G324
L = [ (:-), mortal(_G324), human(_G324)]
```

Note that :- is a functor!

Database Modification

- `assert` allows you to modify things while Prolog is running.
- This only works for "dynamic" procedures, though.
- `retract` allows you to undo an assertion.

```
?- assert(prime(2)).
?- assert(prime(3)).
?- assert(prime(5)).
?- assert(prime(7)).
?- prime(3).
Yes
?- retract(prime(3)).
?- prime(3).
No
```

Making Things Dynamic

```
?- dynamic likes/2.
?- likes(john,mary).
No
?- assert(likes(X,Y) :- likes(Y,X)).
?- assert(likes(john,mary)).
?- likes(mary,X).
ERROR: Out of local stack
?- retract(likes(john,mary)).
Yes
?- asserta(likes(john,mary)).
Yes
?- likes(mary,X).
X = john
```

Executing Code

- The `call` predicate will execute its argument.
- Note that implications are asserted, not called.

```
1 ask_about(X,Y) :- Q =.. [Y,X], call(Q).
2 ?- ask_about(socrates,mortal).
3 Yes
4 ?- call(funny(X) :- human(X)).
5 ERROR: Undefined procedure: (:-)/2
6 ?- assert(funny(X) :- human(X)).
7 X = _G324
8 Yes
9 ?- funny(X).
10 X = socrates
11 X = muller
```

Example: answer

Now you can use Prolog to keep track of students' questions.

```
1 answer(X) :- question(X,Q), !, write(Q),
2              retract(question(X,Q)), call(Q).
3 ?- assert(question(jonny,mortal(muller))).
4 ?- assert(question(jonny,mortal(socrates))).
5 ?- answer(jonny).
6 mortal(muller)
7 Yes
8 ?- answer(jonny).
9 mortal(socrates)
10 Yes
11 ?- answer(jonny).
12 No
```