1

Functors

• Functors are used to define data structures in Prolog.

• Functors are terms with 0 or more arguments. Each functor can be thought as a tree structure, with the function name as the root, and the arguments as the leaves.

• Functors are the only data structures provided in Prolog!

• Functors are not automatically evaluated in theorem proving, or evaluation! For example:

```
|?-  print(3 + 5).
3 + 5
|?-  print(X is 3 + 5).
X is 3 + 5
|?-  X is 3+5, print(X).
8
```


©1993 John Mylopoulos

2

Numbers

Let's try to define the greatest common divisor function

```
gcd(X, X, X).
gcd(X, Y, D) :-  lessThan(X, Y),
   minus(Y, X, Y1), gcd(X, Y1, D).
gcd(X, Y, D) :-  greaterThan(X, Y),
   minus(X, Y, Y1), gcd(Y1, Y, D).
```

But, it is inefficient to use inference to decide that a number is less than another, or to find the difference of two numbers.

```
gcd(X, X, X).
gcd(X, Y, D) :- lessThan(X, Y),
   Y - X = Y1, gcd(X, Y1, D).
gcd(X, Y, D) :-  notLessThan(X, Y),
   X - Y = Y1, gcd(Y1, Y, D).
```

won't work either

`?- gcd(1, 2, Z)` will generate

```
?- gcd(1, 2 - 1, Z1)
?- gcd(1 - (2 - 1), 2 - 1, Z2)
```

or fail depending on how `lessThan, notLessThan` is defined

What is needed is functions that evaluate one or both their arguments.

©1993 John Mylopoulos

3

Numbers

```
0,  -999  0.23E5
```

The built-in function **`is`** evaluates its second argument, an arithmentic expression, and binds the result to its first argument, a variable:

```
X is 2 * 3      binds 6 to X
7 is 2 *3       fails
```

Any variables in the arithmetic expression must be bound to numbers or arithmetic functors that are bound.

Arithmetic Functors:

`X + Y`	addition
`X - Y`	subtraction
`X * Y`	multiplication
X / Y	division
`X // Y`	integer division
`X mod Y`	modulus
`-X`	negation
`X /\ Y`	bitwise and
`X << Y`	shift left X by Y bits

Arithmetic functors do not evaluate by themselves!

©1993 John Mylopoulos

4

Numbers (cont'd)

Arithmetic Predicates:

`X =:= Y`	values are equal
`X < Y`	
`X > Y`	
`X =< Y`	
`X >= Y`	

The left and right hand side of arithmetic predicates must be bound to numbers, or arithmetic functors that are completely bound.

Arithmetic Predicates evaluate the left hand and right hand arithmetic expressions before testing whether the arithmetic relationship holds.

Meta-Logical Predicates:

`number(X)`	`X` is currently instantiated to a number
`integer(X)`	`X` is currently instantiated to an integer

©1993 John Mylopoulos

5

Numbers

Now we can get `gcd` right!

```
gcd(X,X,X).
gcd(X,Y,D) :-
   X < Y, Y1 is Y - X, gcd(X,Y1,D).
gcd(X, Y, D) :-
   Y < X, Y1 is X - Y, gcd(Y1,Y,D).
```

- `?- gcd(6,2,V)` succeeds with `{V/2}`
- `?- gcd(2,V,W)` fails

There is a price to pay
for moving away from pure Prolog,
and computation viewed purely as inference!!

©1993 John Mylopoulos

6

= vs =:= vs is

`?- =(17,+(8,9))`	fails
`?- =(N,+(8,9))`	N = +(8, 9)
`?- =(+(N,9),+(8,9))`	N = 8
`?- =(+(16,1),+(8,9))`	fails
`?- =:=(17,+(8,9))`	succeeds
`?- =:=(N,+(8,9))`	syntax error
`?- =:=(+(N,9),+(8,9))`	syntax error
`?- =:=(+(16,1),+(8,9))`	succeeds
`?- is(17,+(8,9))`	succeeds
`?- is(N,+(8,9))`	N = 17
`?- is(+(8,9),N)`	syntax error
`?- is(+(N,9),+(8,9))`	fails
`?- is(+(16,1),+(8,9))`	fails

©1993 John Mylopoulos

7

Character Strings

In C-Prolog, character strings are represented as lists of numbers equal to the ASCII representation of each character. To simplify character string input a shorthand for a character string is to place the characters withing double quotes:

```
   | ?- X = "abcd".
   X = [97,98,99,100]
   yes

|?- get0(X).    reads one character from
|: a  <Return>   input stream
X = 97
yes

|?- name(abcd, [97,98,99,100]).
yes

|?- name(abcd,X).  converts atom to list
X = [97,98,99,100]  of ASCII codes
yes

|?- name(X,[97,98,99,100]).
             converts list of ASCII codes
X = abcd     to an atom
yes
```


©1993 John Mylopoulos

8

Input and Output

The basic input and output functions are `read`, `write`, `nl` and `put`:

```
   | ?- read(X).
   |  john .
   X = john
   yes

   | ?- write(john).
   john
   yes

|?-  write(john),  write(mary),  nl,
write(sue).
johnmary
sue
yes

|?- put(97),put(98),nl,put(99),put(100).
ab
cd
yes
```


©1993 John Mylopoulos