

# *Logic Programming*

## *Prolog as Language*

Temur Kutsia

Research Institute for Symbolic Computation  
Johannes Kepler University Linz, Austria  
`kutsia@risc.jku.at`

1/30

## Prolog as Language

- ▶ Syntax
- ▶ Operators
- ▶ Equality
- ▶ Arithmetic
- ▶ Satisfying Goals

2/30

# Syntax

## Terms:

- ▶ constant
- ▶ variable
- ▶ structure

3/30

## Constants

- ▶ Naming (specific objects, specific relationships)
  - ▶ likes mary john book wine owns jewels  
can\_steal
  - ▶ a
  - ▶ void
  - ▶ =
  - ▶ 'george-smith'
  - ▶ -->
  - ▶ george\_smith
  - ▶ ieh2304
- ▶ Integers (size is implementation dependent)

4/30

# Non-Constants

The following symbols are not constants:

- ▶ `2340ieh` – Begins with number.
- ▶ `george-smith` – Contains dash.
- ▶ `Void` – Begins with capital.
- ▶ `_alpha` – Begins with underscore.

5/30

# Variables

Begin with capital or with underscore:

- ▶ `Answer`
- ▶ `Input`
- ▶ `_3_blind_mice`

Anonymous variable: A single underscore

- ▶ `likes(john, _)`.
- ▶ Need not be assigned to the same variable `likes(_, _)`.

6/30

# Structures

- ▶ Collection of Objects, *Components*, grouped together in one object.
- ▶ Help Organize.
- ▶ Make code more readable.

7/30

# Structures

Example: Index Card for Library

- ▶ Author's Name
- ▶ Title
- ▶ Date
- ▶ Publisher
- ▶ Name could be split also first, last, etc.

8/30

# Examples

- ▶ `owns(john, book)` .
- ▶ **One Level:**  
`owns(john, wuthering_heights)` .  
`owns(mary, moby_dick)` .
- ▶ **Deeper:**  
`owns(john, book(wuthering_heights, bronte))` .  
`owns(john, book(wuthering_heights,  
author(emily, bronte)))` .

9/30

# Questions

- ▶ **Does John own a book by the Bronte sisters?**  
`owns(john, book(X, author(Y, bronte)))` .
- ▶ **For the yes/no question**  
`owns(john, book(_, author(_, bronte)))` .  
(note that each `_` could be different)

10/30

# Equality

An infix operator =

- ▶  $X = Y$   
A match is attempted between expression  $X$  and expression  $Y$
- ▶ PROLOG does what it can to match  $X$  and  $Y$

11/30

## Example: Instantiated

- ▶  $X$  is uninstantiated.
- ▶  $Y$  is an object.
- ▶  $X = Y$ :  $X$  and  $Y$  will be matched.
- ▶ Thus  $X$  will be instantiated by the object  $Y$ .

```
?- rides(man,bicycle) = X.
```

```
X = rides(man,bicycle) .
```

12/30

## Example: Symbols

```
?- policeman = policeman.
```

Yes

```
?- paper = pencil.
```

No

```
?- 1066 = 1066.
```

Yes

```
?- 1206 = 1583.
```

No

13/30

## Arguments Instantiated

- ▶ If the structures are equal then their arguments are matched.

```
?- rides(man,bicycle) = rides(man,X).
```

```
X = bicycle.
```

14/30

# Arguments Instantiated

?- a(b, C, d(e, F, g(h, i, J))) =  
a(B, c, d(E, f, g(H, i, j))).

B = b

C = c

E = e

F = f

H = h

J = j

15/30

# Equality

?- X = X.

true

?- Y = X.

Y = X

16/30



# Equality

?- X = Y, X = 1200.  
X = 1200, Y = 1200  
?-

17/30

# Arithmetic Comparisons

$X = Y$

$X \backslash= Y$

$X < Y$

$X > Y$

$X =< Y$

$X >= Y$

18/30

# Arithmetic

```
?- 123 > 14.
```

```
true
```

```
?- 14 > 123.
```

```
false
```

```
?- 123 > X.
```

```
ERROR: Arguments are not sufficiently  
instantiated
```

```
?-
```

19/30

## Example

- ▶ Prince was a prince during year, Year if Prince reigned between years Begin and End, and Year is between Begin and End.

```
prince(Prince, Year) :-  
    reigns(Prince, Begin, End),  
    Year >= Begin,  
    Year =< End.
```

```
reigns(rhodri, 844, 878).  
reigns(anarawd, 878, 916).  
reigns(hywel_dda, 916, 950).  
reigns(lago_ad_idwal, 950, 979).  
reigns(hywel_ab_ieuaf, 979, 985).  
reigns(cadwallon, 985, 986).  
reigns(maredudd, 986, 999).
```

20/30

# Runs

- ▶ Was Cadwallon a prince in 986?
- ▶ Is Rhodri a prince in 1995?

```
?- prince(cadwallon, 986).  
true
```

```
?- prince(rhodri, 1995).  
false
```

```
?-
```

21/30

# Who was a Prince When

- ▶ Who was the prince in 900?
- ▶ Who was the prince in 979?

```
?- prince(Prince, 900).  
Prince = anarawd ;  
false
```

```
?- prince(Prince, 979).  
Prince = lago_ad_idwal ;  
Prince = hywel_ab_ieuaf ;  
false
```

```
?-
```

22/30

# Invalid Question

- ▶ When was Cadwallon a prince?

```
?- prince(cadwallon, Year).
```

```
ERROR: Arguments are not sufficiently  
instantiated
```

23/30

## Calculating

- ▶ Calculating the Population Density of a Country:  
Population over the Area

```
density(Country, Density) :-  
    pop(Country, Pop),  
    area(Country, Area),  
    Density is Pop/Area.
```

```
pop(usa, 305).  
pop(india, 1132).  
pop(china, 1321).  
pop(brazil, 187).
```

```
area(usa, 3).  
area(india, 1).  
area(china, 4).  
area(brazil, 3).
```

24/30

## Questions

- ▶ What is the population density of USA?

```
?- density(usa, X).  
X = 101.667 ;  
false
```

25/30

## Questions

- ▶ What Country has which density?

```
?- density(X, Y).
```

```
X = usa  
Y = 101.667 ;
```

```
X = india  
Y = 1132 ;
```

```
X = china  
Y = 330.25 ;
```

```
X = brazil  
Y = 62.3333 ;
```

```
false  
?-
```

26/30

# Arithmetic Operations

$X + Y$

$X - Y$

$X * Y$

$X / Y$

$X \text{ mod } Y$

27/30

## How Prolog Answers Questions

**Program:**

```
female(mary).
```

```
parent(C, M, F) :-
```

```
    mother(C, M),
```

```
    father(C, F).
```

```
mother(john, ann).
```

```
mother(mary, ann).
```

```
father(mary, fred).
```

```
father(john, fred).
```

**Question:**

```
?-female(mary), parent(mary, M, F), parent(john, M, F).
```

**How does it work?**

28/30

# Matching

- ▶ An uninstantiated variable will match any object. That object will be what the variable stands for.
- ▶ An integer or atom will only match itself.
- ▶ A structure will match another structure with the same functor and the same number of arguments and all corresponding arguments must match

29 / 30

## How Is this Matched

```
?- sum(X+Y) = sum(2+3) .
```

```
X = 2,
```

```
Y = 3
```

30 / 30