

Logic Programming

The Basics

Temur Kutsia

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

1 / 44

Contents

Basics of PROLOG

- Facts
- Questions
- Variables
- Conjunction
- Rules

2 / 44

PROLOG

Used to solve problems involving

- ▶ objects, and
- ▶ relationships between objects.

3 / 44

Relationships

Example

John owns the book

- ▶ The relationship: *ownership*
- ▶ The objects: *book*, *John*

Directional:

- ▶ John owns the book
- ▶ **Not:** The book owns John

4 / 44

Questions

Example

Does John own the book?

Asks a question about a relationship already established.

5 / 44

Rules

Describe Relationships Using other Relationships.

Example

Two people are sisters if they are both female and have the same parents.

Gives a definition of one relationship given other relationships.

- ▶ Both must be females.
- ▶ Both must have the same parents.
- ▶ If two people satisfy these conditions, then they are sisters (according to our simplified relationship)

6 / 44

Programming in PROLOG

- ▶ **Declaring Facts** about objects and their relationships.
- ▶ **Defining Rules** about objects and their relationships.
- ▶ **Asking Questions** about objects and their relationships.

7 / 44

PROLOG

- ▶ Program can be thought of as a storehouse of facts and rules.
- ▶ Conversational Language: The user can ask questions about the set of facts and rules in the PROLOG program.

8 / 44

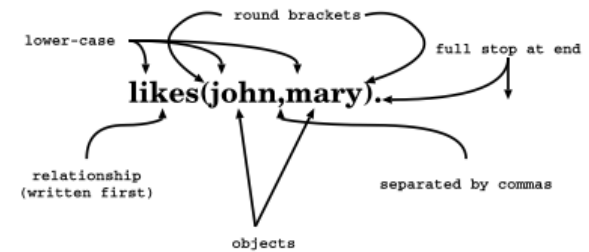
PROLOG

Sisters Example:

- ▶ A rule defining sisters and the facts about the people involved.
- ▶ The user would ask:
Are these two people sisters?
- ▶ The system would answer
yes (true) or **no** (false)

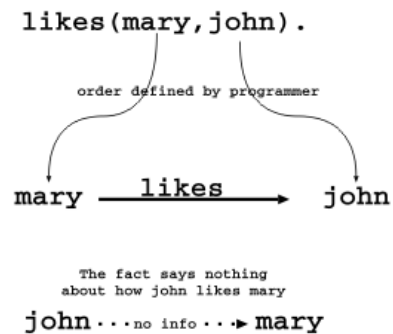
9/44

Parts of Fact



10/44

Order of Objects



11/44

Examples of Facts

Example

Gold is valuable.
`valuable(gold)`

Jane is a female.
`female(jane)`

John owns some gold.
`owns(john, gold)`

John is the father of Mary.
`father(john, mary)`

Are these expressions really facts? Is there anything missing?

12/44

Interpretation of Names

The name refers to an object.

- ▶ **Semantic Meaning:** Given by the programmer.
- ▶ **Syntactic Meaning:** a set of characters, as PROLOG sees it.

13 / 44

Interpretation of Names

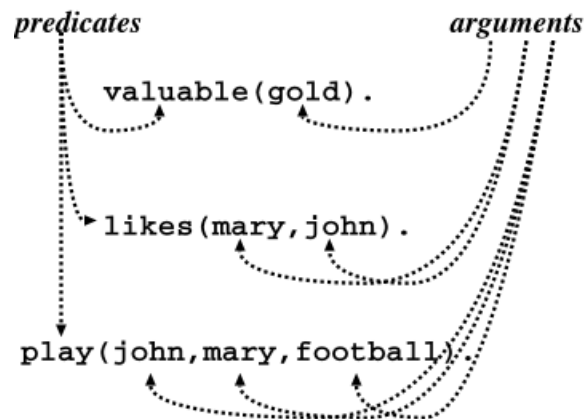
Name refers to an object.

- ▶ Name `gold` can refer to:
 - ▶ a particular lump of gold, or
 - ▶ the chemical element Gold having atomic number 79.
- ▶ `valuable(gold)` can mean:
 - ▶ that particular lump of gold, named `gold`, is valuable, or
 - ▶ the chemical element Gold, named `gold`, is valuable.

The programmer decides (in her usage) the meaning.

14 / 44

Fact Terminology



15 / 44

Database

Definition

In PROLOG, **database** is a collection of facts.

- ▶ PROLOG draws its knowledge from these facts.
- ▶ The programmer is responsible for their accuracy.

16 / 44

Questions

- ▶ The database contains the facts from which the questions are answered.
- ▶ A question can look exactly like a fact:
`owns(mary, book).`
- ▶ The difference is in which mode one is in.

17 / 44

Questions

In the interactive question mode (indicated by the question mark and dash `?-`):

- ▶ Question: `?- owns(mary, book).`
- ▶ Meaning:
 - ▶ If `mary` is interpreted as a person called Mary, and `book` is interpreted as some particular book, then
 - ▶ `?- owns(mary, book).` means: **Does Mary own the book?**

18 / 44

Database Search

Example

Facts in the database:

```
likes(joe, fish).  
likes(joe, mary).  
likes(mary, book).  
likes(john, book).
```

Questions:

```
?- likes(joe, money).  
no  
?- likes(joe, mary).  
yes  
?- king(john, france).  
no
```

19 / 44

Knowledge

The questions are always answered with respect to the database.

Example

Facts in the database:

```
human(socrates).  
human(aristotle).  
athenian(socrates).
```

Question:

Is Socrates Greek?

```
?- greek(socrates).
```

The answer with respect to this database is **No**.

20 / 44

Questions

Up until now questions just reflect exactly the database.

Does Mary like the book?

```
?- likes(mary, book).
```

More Interesting Question:

What objects does Mary like?

Variables.

21 / 44

Variables

Tiresome to ask about every object:

```
likes(john, this).
```

```
likes(john, that).
```

Better to ask:

What does John like?

or

What is an **X** such that John like **X**?

(i.e. use variables)

22 / 44

Question With Variables

What is an **X** such that John like **X**?

```
?- likes(john, X).
```

or

```
?- likes(john, SomethingThatJohnLikes).
```

X and **SomethingThatJohnLikes** are variables.

Variable begins with a capital letter.

23 / 44

PROLOG Answer

Database:

```
likes(john, flowers).
```

Question:

```
?- likes(john, X).
```

PROLOG answers:

```
X=flowers
```

24 / 44

Many Answers

Database:

```
likes(john, flowers).  
likes(john, mary).  
likes(paul, mary).
```

Question:

```
?- likes(john, X).
```

PROLOG answers:

```
X=flowers  
and the user acknowledges  
X=mary  
and the user acknowledges  
no
```

25 / 44

Placemark

- ▶ The first match is found: `X=flowers`.
- ▶ The user acknowledges.
- ▶ From that place on the next match is found (the search continues).
- ▶ From the place of the last instantiation no more match was found.
- ▶ Thus answer: `no`.

26 / 44

Conjunctions

More complicated relationships:

Does Mary like John and does John like Mary?

Both conditions must be fulfilled.

27 / 44

Conjunctions

Comma means conjunction:

```
?- likes(john, mary), likes(mary, john).
```

```
likes(mary, food).  
likes(mary, wine).  
likes(john, wine).  
likes(john, mary).
```

Answer: `no`

A match for `likes(john, mary)`
but none for `likes(mary, john)`

28 / 44

Conjunctions with Variables

Is there anything that both mary and john like?

Find out what Mary likes and then see if John likes it.

```
?- likes(mary, X), likes(john, X).
```

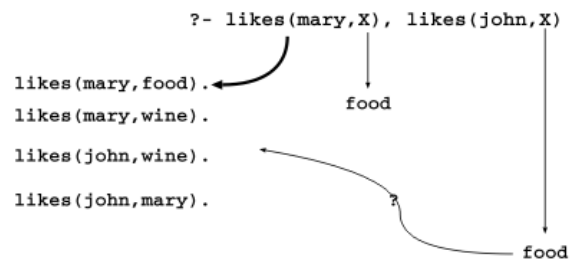
29 / 44

Backtracking

- ▶ Find match for the first goal.
- ▶ Then see if it matches the second.
- ▶ If not, find another match for the first.
- ▶ See if this matches the second.
- ▶ etc.

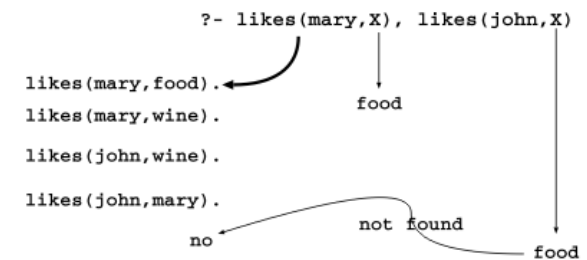
30 / 44

Match First



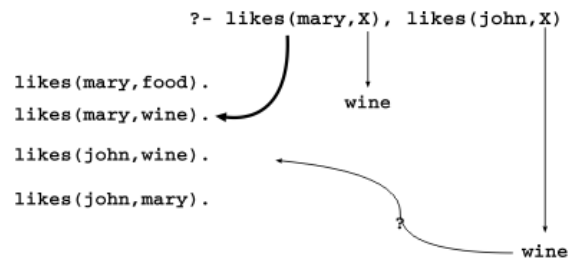
31 / 44

Match Second



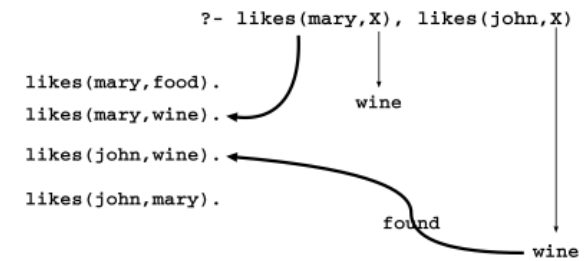
32 / 44

Backtrack



33 / 44

Success



34 / 44

Rules

How to express that John likes all people?

Listing all people?

- ▶ likes(john, alfred).
- ▶ likes(john, bertrand).
- ▶ likes(john, charles).
- ▶ likes(john, david).
- ▶ etc.

Not feasible. More compact way: Using [rules](#).

John likes any object provided it is a person.

35 / 44

Rule Examples

Rules state Dependence:

- ▶ I use an umbrella **if** there is rain.

Rules Define:

- ▶ X is a bird **if** X is an animal and X has feathers.

36 / 44

Formulating Rules

- ▶ John likes anyone who likes wine.
- ▶ John likes any something if it likes wine.
- ▶ John likes X if X likes wine.

37 / 44

Rule Syntax

`likes(john,X) :- likes(X,wine).`

↑ ↑ ↑
head rule delimiter body

38 / 44

Variable Scope

The occurrences of `x` within a rule:

instantiates here
↓
`likes(john, X) :- likes(X, wine),`
↑ ↑
returns here checked here
`likes(X, food).`

39 / 44

Royal Parents

Example

- ▶ The parents of X are Y and Z.
- ▶ Y is the mother.
- ▶ Z is the father.

Database:

```
male(albert).  
male(edward).  
female(alice).  
female(victoria).  
parents(edward, victoria, albert).  
parents(alice, victoria, albert).
```

40 / 44

Sisters

Example

X is a sister of Y if:

- ▶ X is female,
- ▶ X has parents M and F,
- ▶ Y has parents M and F.

Rule:

```
sister(X, Y) :-  
    female(X),  
    parents(X, M, F),  
    parents(Y, M, F).
```

41/44

Sisters Question

Rule:

```
sister(X, Y) :-  
    female(X),  
    parents(X, M, F),  
    parents(Y, M, F).
```

Question:

```
sister(alice, edward).
```

- ▶ The question (goal) matches the head of the rule, if one replaces X with alice and Y with edward.
- ▶ The instance of the body becomes a new goal:

```
female(alice),  
parents(alice, M, F),  
parents(edward, M, F).
```

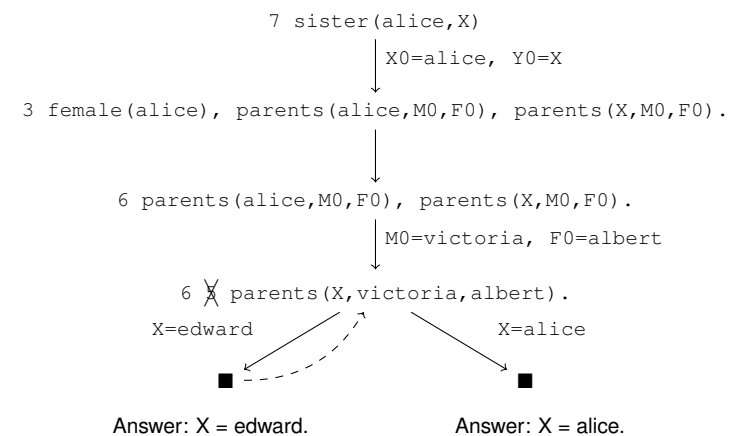
42/44

The Complete Program

```
(1) male(albert).  
(2) male(edward).  
(3) female(alice).  
(4) female(victoria).  
(5) parents(edward, victoria, albert).  
(6) parents(alice, victoria, albert).  
(7) sister(X, Y) :-  
    female(X),  
    parents(X, M, F),  
    parents(Y, M, F).
```

43/44

Complete Derivation Tree



44/44