Prolog

PROgramming in **LOG**ic

Solves problems that involve
objects
and
their *relationships*

This Lesson

The Basics

Facts

Questions

Variables

Conjuctions

Rules

Relationships

Example

John owns the book

The relationship

ownership

The objects

book

John

Directional

John owns the book

but

The book does not own John

Questions

Does John own the book?

Asks a question about
a relationship
already established

Rules

Rules Describe Relationships
Using other Relationships

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 rules, then they are sisters
(according to our simplified relationship)

Programming in Prolog

Declaring Facts

about objects and their relationships

Defining Rules

about objects and their relationships

Asking Questions

about objects and their relationships

Prolog

A program in Prolog 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

Prolog

PROLOG Sisters Example

A rule defining sisters

and

the facts about the people involved

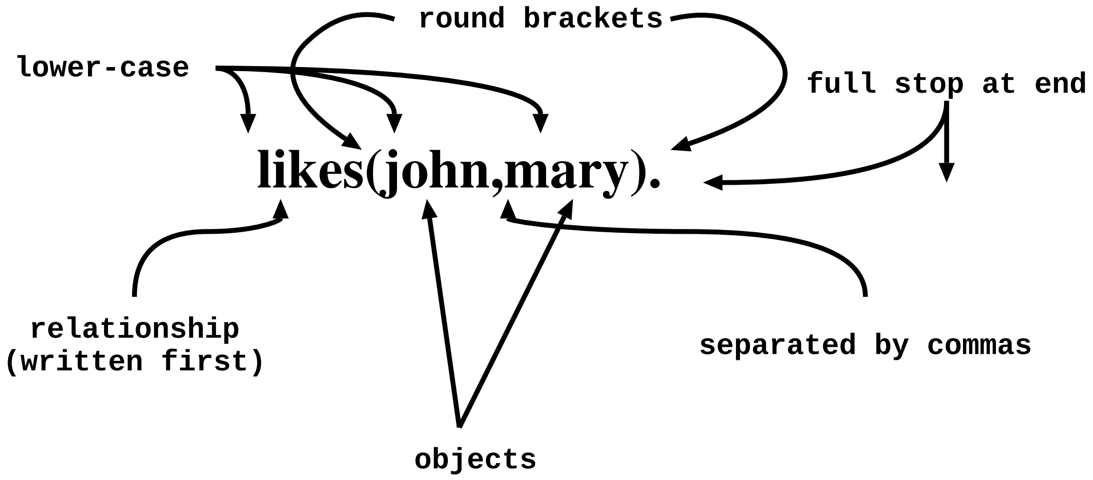
The user would ask

Are these two people sisters?

and the system would answer

yes (true) or **no** (false)

Parts of Fact



Order of Objects

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

order defined by programmer

$\texttt{mary} \xrightarrow{\texttt{likes}} \texttt{john}$

The fact says nothing
about how john likes mary

$\texttt{john} \xrightarrow{\texttt{no info}} \texttt{mary}$

Examples of Facts

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)`

Interpretation of Names

The name refers to an object

Semantic Meaning

The programmer gives it the meaning

Syntactic Meaning

PROLOG sees it as a set of characters

Interpretation of Names

Gold

`valuable(gold)`

A particular lump of gold (sitting on the table)
is valuable

or

The mineral gold (in general)
is valuable

The programmer decides (in his usage) the meaning

Fact Terminology

predicates ***arguments***

```
valuable(gold).

likes(mary,john).

play(john,mary,football).
```

Database

The collection of facts
in a PROLOG system
is the database

Note:
PROLOG draws its knowledge from these facts
The programmer is responsible for their accuracy

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

Questions

In the interactive question mode

`owns(mary,book)`

Does Mary (`mary`)

own (`owns`)

a particular book (`book`)?

PROLOG asks

Is there a fact

in the database

which says that mary owns the book?

Database Search

Facts

```
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
```

Knowledge

The questions are always answered

with respect to the database

Facts

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

Questions

Is socrates Greek?

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

The answer with respect to the simple database is **No**.

Reflects Database

Up until now

Questions just **reflects exactly** the database

Does Mary like the book

`likes(mary,book).`

More Interesting Question

What objects does Mary like?

Variables

Variables

Tiresome to ask about every object

`likes(john,this)`

`likes(john,that)`

.

.

.

Better to ask: *What does John like*

or

Does John like **X**

(i.e. use variables)

Question With Variables

Does John like **X**

`likes(john,X)`

or

`likes(john,SomethingThatJohnLikes)`

`X`

and

`SomethingThatJohnLikes`

are variables:

Variables

Begin with Capital

PROLOG Answer

Database

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

Question

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

PROLOG answers

```
X=flowers
```

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

Place-Marker

The first match is found

`X=flowers`

and the user acknowledges

and 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 answers `no`

Conjunctions

More Complicated Relationships

Does Mary like John

and

Does John like Mary

Both Conditions must be fulfilled

Conjunctions

Comma means Conjuction

```
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)`

Conjuctions 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).
```

Backtracking

Find match for first goal
then see if matches second
if not
Find another match for second
and see if this matches
etc.

Match 1st

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

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

likes(john,wine).

likes(john,mary).                    ?

                                                 food
```

Match 2nd

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

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

likes(john,wine).

likes(john,mary).
                                 not found
                  no                              food
```

Backtrack

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

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

wine

?

wine

Success

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

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

wine

found

wine