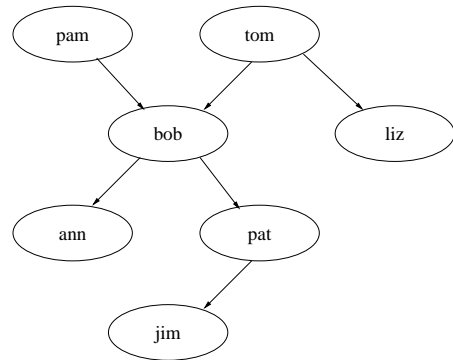


Family Example

```
parent(tom, bob).  
parent(pam, bob).  
parent(tom, liz).  
parent(bob, ann).  
parent(bob, pat).  
parent(pat, jim).
```

Family Example — Structure



1

2

Rules

Program: 6 clauses

Clause: their collection defines a *predicate* (parent relation)

Relation: a relation R on sets A, B is a subset $R \subseteq A \times B$.

Question:

```
?- parent(bob, pat).
```

yes.

etc.

Rules and Variables

Question: `?- parent(X, liz).`

It asks: “Does liz have a parent? If yes, for which X is this the case?”

Answers $\exists X : \text{parent}(X, \text{liz})$ and instantiates X.

Alternative Question: `?- parent(bob, X).`

What does this ask?

Another Question: what does

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

ask?

$X, Y : \text{parent}(X, Y)$ and, if yes instantiates X and Y.

“Do there exist parent-children pairs”?

3

4

Closed-World Assumption

System knows only about facts (and deducible relations). All else is considered *false*.

Question: Does jim have grandparents (in Prolog this question makes sense) and, if yes, who are they.

Reformulate: Does there exist a parent who is a parent to someone who is a parent to jim?

Formally:

$$\exists X \exists Y : \text{parent}(X, Y) \wedge \text{parent}(Y, \text{jim})$$

In Prolog: `?- parent(Y, jim), parent(X, Y)`

fulfills the predicate 1, then fulfilling the next.

Note: reverse order *does not change anything!*

Question: who are tom's grandchildren?

Question: who are the siblings of ann?

5

6

Syntax

Relations: n -tuple facts

Queries: about relations

Clauses: end with full stop

Objects: of *first kind*: atoms, like tom and bob

Objects: of *second kind*: variables

Questions: are goals

Satisfiable Goal: succeeds

Unsatisfiable Goal: fails

Goal sequences: interpreted as conjunction of goals

Style

Consider: `female(pam).`

`male(tom).`

`male(bob).`

Unary Relation: property, present or not

Alternative Description: `gender(pam, female).`

`gender(tom, male).`

`gender(bob, male).`

Function: binary relation $f(., .)$ with

$$\forall x \exists y : f(x, y)$$

More general: n -ary relation, where to each $n - 1$ -tuple there exists exactly one n -th entry:

$$f(x_1, \dots, x_{n-1}) = x_n \Leftrightarrow f(x_1, \dots, x_{n-1}, x_n)$$

7

8

Define: predicate offspring:

$$\forall x \forall y : \text{parent}(x, y) \Rightarrow \text{offspring}(y, x)$$

offspring(Y,X) :- parent(X,Y)

:- represents \Leftarrow

Rule: defined by “:-”

Note:

1. a *fact* is always true
2. left side of a rule is true if goals on right side are satisfied

Left side: head

Right side: body

9

Matching

offspring(liz , tom) X and Y matched

↑ ↑
X Y

(“unified”) and goal parent(Y,X) tested with this instantiation.

Trivially true since it is a fact.

Note: $\forall x \forall y$: always implicit for the left side

Example

```
Example: mother(X, Y) :-
    parent(X, Y),
    gender(X, female).

grandparent(X, Z) :-
    parent(X, Y), /* for some Y */
    parent(Y, Z).

sister(X, Y) :-
    parent(Z, X), /* for some Z */
    parent(Z, Y), /* satisfied for the same Z */
    gender(X, female).
```

Check:

```
?- sister(X, pat).
ann;
pat
```

11

Negation

Question: Is pat sister to herself?

Ideas: use concept for different: use \neq , giving

```
sister(X, Y) :-
    parent(Z, X), /* for some Z */
    parent(Z, Y), /* satisfied for same Z
    X  $\neq$  Y,
    gender(X, female).
```

Note: This implies a *negation*. Truth is a deep-lying concept of predicate logic and the incompleteness has to be covered somehow by Prolog as a computer language. Therefore, negation becomes a nontrivial concept in Prolog, treated later.

Note: in principle, Prolog clauses and goals need not to be ordered to describe the logically identical object. There are exceptions, like negation and other predicates implicitly using negation. E.g. \neq requires both arguments to be instantiated before use to work correctly, In above example, X and Y. It could not begin the body of the clause and work as expected. Reasons will be given later.

12

Bottom Line

Clauses: 3 types

1. facts: always true
2. rules: true on condition
3. question: establishing which things are true
4. clause: head & body
5. body: list of goals separated by commas (conjunctions)
6. facts: head with empty body
7. questions: only body
8. rules: head & nonempty body

Variables: can be instantiated

- by other object during satisfaction
- in the head read \forall
- in the body read \exists