upquote=true, language=XML, basicstyle=, frame=lines, tabsize=4, numbers=left, numberstyle=, tabsize=2, captionpos=b, escapeinside=%**)

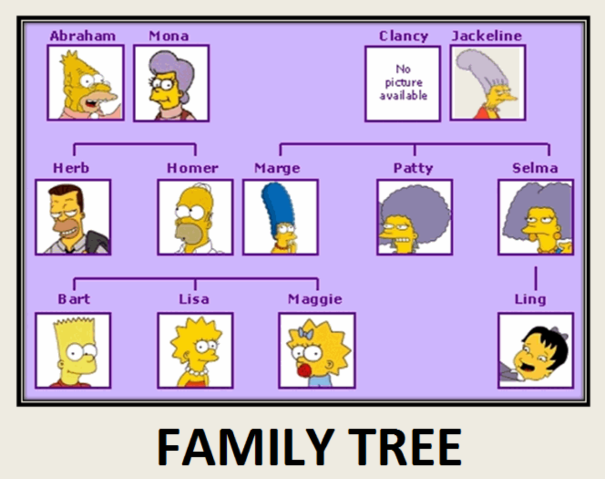
Logic Programming, 20-21

Lecturer: Vítor Santos Costa
DCC-FCUP
vsc@dcc.fc.up.pt (room: 1.45)

These slides are largely based on Prof. Inês Dutra's and Prof. Alípio Jorge

17 de março de 2022

The Simpsons family tree



- We want a program that answers questions about the family.

The Simpsons family tree

To describe a family tree we need some facts:

- Homer is the father of Bart, Lisa and Maggie.
- Marge is their mother.

Some entities are mentioned here (Homer, Bart, Lisa, etc.). These are important concepts which will be represented as **constants**. We also have relations such as "X is father of Y" and "X is mother of Y". father and mother are also called **predicates**. In Prolog the above **facts** are represented as follows.

- We want a program that answers questions about the family.

Stating the facts!

In Prolog, the above facts are represented as follows:

```
father(homer,bart).
father(homer,lisa).
father(homer,maggie).

mother(marge,bart).
mother(marge,lisa).
mother(marge,maggie).
```

Note

- constants
- predicates
- period

Asking questions

We can ask questions such as (pose queries)

- *Who is the father of Bart*?

```
?-father(X,bart).
```

Type enter

```
X = homer
```

Note

- The answer is simply "Homer"
- The query is posed with a **variable** `X`
- We are asking is "*Is there an X such that X is father of bart?*".
- The answer is "*yes, and the value of X is homer*".

Asking questions

We can ask questions such as (pose queries)

- *Who are the offsprings of Marge*?

```
?-mother(marge,X).
X = bart;
X = lisa;
X = maggie
```

Note

- `X` can take multiple values
- mother(marge,X) is **true** for multiple values of `X`
- but we used **mother** instead of **offspring**...

Defining concepts

We have facts concerning the predicates **mother** and **father**.

- We can ask about offsprings by quering about **mother** and **father**.
- Nothing wrong, but I would like to have the concept **offspring**.
- Poor solution: write the facts!
- Good solution: define the concept **offspring** in general.

```
offspring(X,Y):-mother(Y,X).
offspring(X,Y):-father(Y,X).
```

Note

- We wrote two **rules**.
- Each one is a **logical implication** (:- represents ←).
- How to read each rule?

Defining concepts

We have facts concerning the predicates **mother** and **father**.

- We can ask about offsprings by quering about **mother** and **father**.
- Nothing wrong, but I would like to have the concept **offspring**.
- Poor solution: write the facts!
- Good solution: define the concept **offspring** in general.

Posing queries about **offspring**.

```
?-offspring(X,marge).
...
?-offspring(bart,X).
...
?-offspring(X,Y).
...
```

- How are these queries answered?

Defining concepts

Examples.

- Define the concept **son**.
- Define the concept **daughter**.
- Define the concept **parent**, in three different ways.
- Define the concept **sibling** of both sides.

Comparing Prolog terms

Defining the predicate **sibling/2**

```
sibling(X,Y):-
    mother(M,X),mother(M,Y),
    father(F,X),father(F,Y).

?-sibling(bart,X).
X = bart;
X = lisa;
X = maggie
```

We do not want Bart to be his own sibling...

Comparing Prolog terms

We need to state that X and Y are different objects

```
sibling(X,Y):-
    mother(M,X),mother(M,Y),
    father(F,X),father(F,Y),
    not(X==Y).
```

- not(X) is true if X is false.
- X==Y is true if X and Y are the same objects.
- X=Y is true if X and Y are unifiable (and they unify).
- X=:=Y is true if X and Y are the same numerical (or expressions that result in the same numerical)

Unifying Prolog terms

Let us extend our “knowledge base” (the Prolog program) with more facts.

```
father(homer,bart).           father(homer,lisa).
father(homer,maggie).         father(abraham,homer).
father(abraham,herb).         father(clancy,marge).
father(clancy,patty).         father(clancy,selma).


mother(marge,bart).           mother(marge,lisa).
mother(marge,maggie).         mother(mona,homer).
mother(mona,herb).            mother(jackline,marge).
mother(jackline,patty).       mother(jackline,selma).
mother(selma,ling).
```

Unifying Prolog terms

And define the predicate **grandfather/2**.

```
grandfather(X,Y):-father(X,A), father(A,Y).
grandfather(X,Y):-father(X,A), mother(A,Y).
```

In both clauses A is unified implicitely...
We can use explicit unification (without advantage here).

```
grandfather(X,Y):-father(X,A), father(B,Y), A=B.
grandfather(X,Y):-father(X,A), mother(B,Y), A=B.
```

Compound terms

Besides constants and variables, Prolog also has compound terms.
Let's write a program for arithmetics using logic only.

- The number zero is represented as 0.
- one as $s(0)$, two as $s(s(0))$ and so on.

s is a **functor** of arity 1, i.e. it has one argument.
How to define the predicate **sum/3** ?

```
sum(0,X,X).
sum(s(X),Y,s(Z)):-sum(X,Y,Z).
```

Compound terms

How to define the predicate **sum/3** ?

```
sum(0,X,X).
sum(s(X),Y,s(Z)):-sum(X,Y,Z).
```

Calculating by infering.

```
?-sum(s(0),s(0),X),sum(X,X,Y),sum(X,Y,Z).
X = s(s(0)),
Y = s(s(s(s(0)))),
Z = s(s(s(s(s(s(0))))))
```

Vocabulary

- **term:** constant, variable or compound term.
  - a, adam, X, f(a), 23.
- terms are **ground** if they contain no variables.
  - adam is ground, X is not.
- a **substitution** is a set of pairs `t/X`, where `t` is a term and X is a variable.
- **father(adam,abel)** is an **instance** of **father(X,Y)** because there is a substitution θ that when applied to the more general term results in the more specific one.
  - but not of **father(X,X)**.

Vocabulary

- a **logic program** is a finite set of clauses.
- a **clause** or rule has the form
  $A \leftarrow B_1, B_2, ..., B_k. \quad k \geq 0$
  where A and B_i are **literals**.
- a clause has the **head** and **body**
- when $k = 0$ the clause is a **fact** and is simply A.
- a **query** has the form ?-$A_1, A_2, ..., A_n$.
  - A_i are also called **goals**.

Defining concepts

Examples.

- Define the concept **grandparent**.
- Define the concept **uncle**.
- Define the concept **ancestor**.
- Define the concept **sibling1** of exactly one side.