

1. Facts and rules. Family tree

In this session you will learn how to write a Prolog program, and how to call Prolog predicates. We will start by a simple example – the genealogy tree. We will define a series of relationships – *father, mother, sibling, brother, sister, aunt, uncle, grandmother, grandfather* and *ancestor*. Let's begin with a (partial) depiction of such a tree:

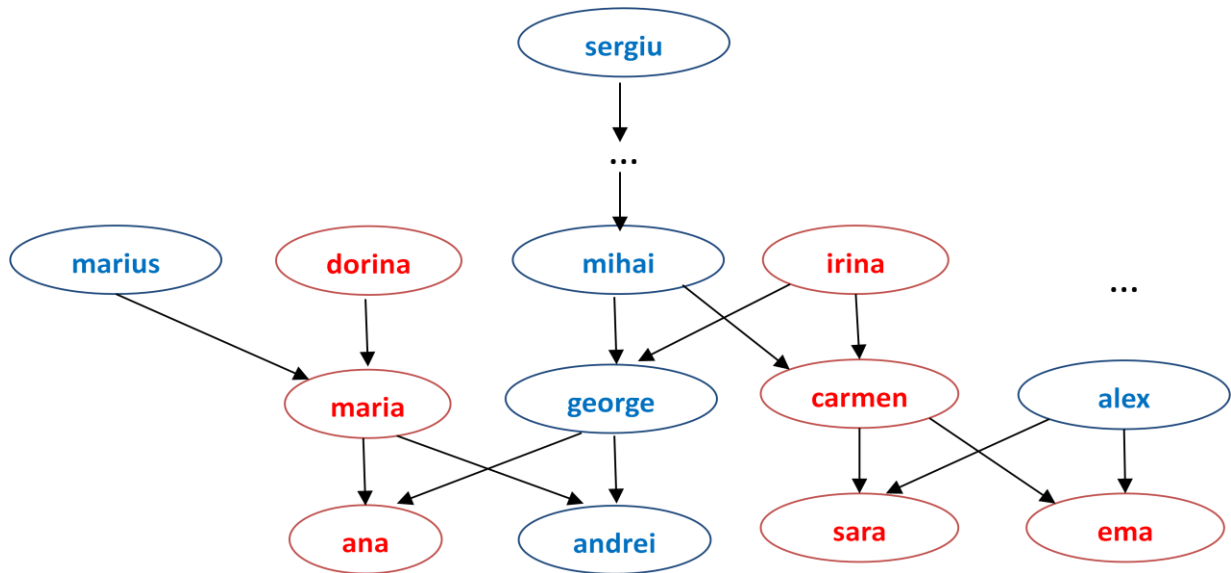


Figure 2.1 – Genealogy tree example

We will build our predicate base incrementally. So, let's start by declaring a number of facts – i.e. ground truths.

Hint: Predicate specifications are written in a source file (a simple text file). You may save it with the extension .pl (recognized by Sicstus Prolog and other Prolog engines, such as SWI Prolog – a free version of Prolog). A line comment in Sicstus starts with %. Inline comments are enclosed between / ... */.*

```
woman(ana). % Remember, predicate names are constant (start with lowercase letter)
woman(sara).
woman(ema).
woman(maria).
% etc...
man(andrei).
man(george).
man(alex).
%etc...
parent(maria, ana). % maria is ana's parent
parent(george,ana). % george also is ana's parent
parent(maria,andrei).
parent(george,andrei).
```

Therefore, we have defined three predicates: man/1, woman/1 and parent/2, each consisting of a series of facts. Predicate parent(X,Y) is to be interpreted as *X is the parent of Y*.

Now let's define a predicate for the *mother* relationship. Of course, we need two arguments – i.e. mother/2 – and we will employ the parent/2 and woman/1 predicates:

```
mother(X,Y):-woman(X), parent(X,Y).  
% X is Y's mother, it X is a woman and X is the parent of Y
```

Exercise 2.1: Define predicate father/2.

Let us now call the predicates. To let Sicstus know about your defined predicates, you have to use the consult/1 built-in predicate – which you can also access directly from the *File->Consult* menu. In the file dialog, select your source file. If all goes well (i.e. not syntax errors in your source file), you should see something like this:

```
| ?- :-  
consult('E:/Scohol/Catedra/PL/Problems/genealogy.pl').  
% consulting e:/scohol/catedra/pl/problems/genealogy.pl...  
% consulted e:/scohol/catedra/pl/problems/genealogy.pl in module user, 0 msec 1248  
bytes  
| ?-
```

Calling a predicate in Prolog is known as “*asking a question*”. Below you may find a listing of Prolog queries, with the answers provided by the engine, on the predicates defined so far (do not copy-paste this in the Sicstus console, it will not work):

```
| ?- man(george). % is george a man?  
yes  
| ?- man(X). % who is a man?  
X = andrei ? ; % use ; or n to repeat the question and ask for another answer  
X = george ? ;  
X = alex ? ;  
no  
| ?- parent(X,andrei). % who are andrei's parents?  
X = maria ? ;  
X = george ? ;  
no  
| ?- parent(maria,X). % who are maria's children?  
X = ana ? ;  
X = andrei ? ;  
no  
| ?- mother(ana,X). % who are ana's children?  
no  
| ?- mother(X, ana). % who is ana's mother?  
X = maria ? ; % repeat the question, i.e. does ana have another mother besides maria?  
no
```

Ok, now try the above queries on your own in Sicstus.

Exercise 2.2: Complete the predicates `man/1`, `woman/1` and `parent/2`, to have the entire genealogy tree in *Fig. 2.1* covered.

Exercise 2.3: Re-consult your source file in Sicstus, and execute the following queries:

- a. `?- father(alex, X).`
- b. `?- father(X,Y).`
- c. `?- mother(dorina, maria).`

Let us now extend the predicate base with several other predicates:

% sibling/2: X and Y are siblings if they have a common parent, and they are different

`sibling(X,Y):-parent(Z,X), parent(Z,Y), X\=Y.`

% sister/2: X is Y's sister if X is a woman and X and Y are siblings

`sister(X,Y):-sibling(X,Y), woman(X).`

% aunt/2: X is Y's aunt if she is the sister of Z, who is a parent for Y.

`aunt(X,Y):-sister(X,Z),parent(Z,Y).`

Exercise 2.4: Extend the predicate base with predicates `brother/2`, `uncle/2`, `grandmother/2` and `grandfather/2`.

Exercise 2.5: Re-consult your source file in Sicstus, and trace the execution of the following queries (by repeating the question):

- a. `?- aunt(carmen, X).`
- b. `?- grandmother(dorina,X).`
- c. `?- grandfather(X,ana).`

Hint: to activate the trace option, simply type `?-trace.` in Sicstus' query prompt. You will be able to follow the execution of your queries call by call. To deactivate it, use `?-notrace.`

Last, let us focus on writing a predicate `ancestor/2`: X is the ancestor of Y if it is linked to Y via a certain number of parent relations. In *Fig. 2.1*, *sergiu* is an ancestor of *mihai*, *sergiu*, *andrei*, *carmen* and *sara*.

Exercise 2.6: Write the `ancestor/2` predicate, and execute several queries on it to test its correctness.

1.1. Quiz Exercises

Phonebook?

Crime_suspect?