Overview

Imperative Languages

- sequence of instructions, executed after each other

- **Procedural Languages**
  - variables, assignments, control structures

- **Object-Oriented Languages**
  - objects and classes
  - ADT and inheritance

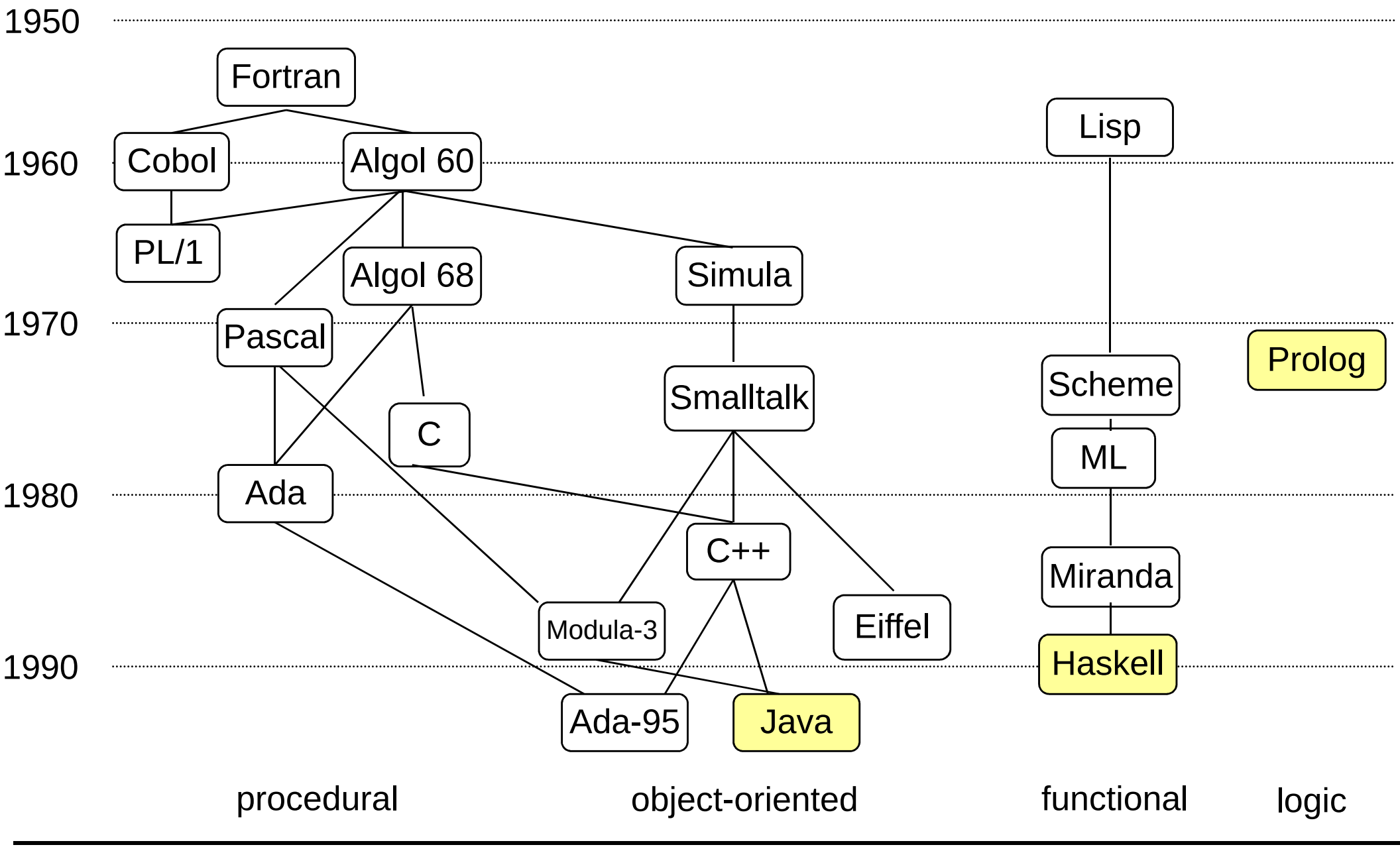
Declarative Languages

- specify *what* should be computed
- compiler determines *how* the computation works

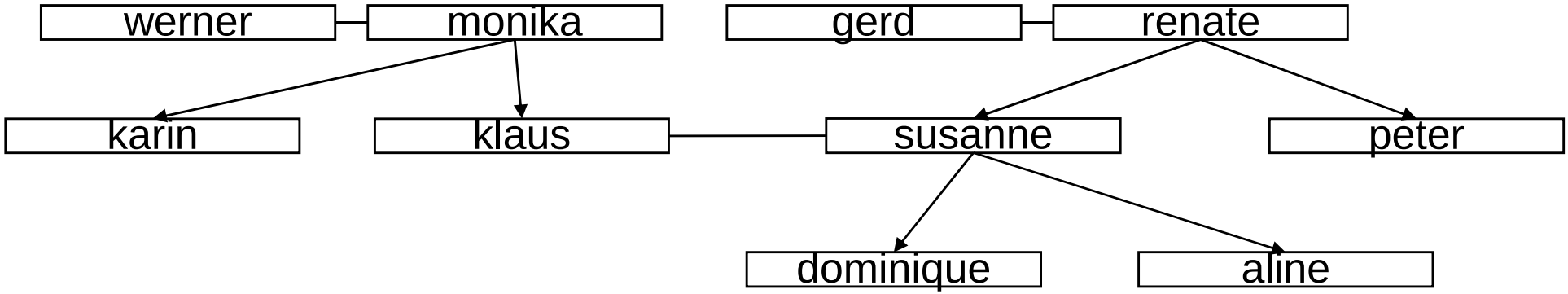
- **Functional Languages**
  - no side-effects
  - recursion

- **Logic Languages**
  - rules to define relations

Important Programming Languages



Facts and Queries



Program:

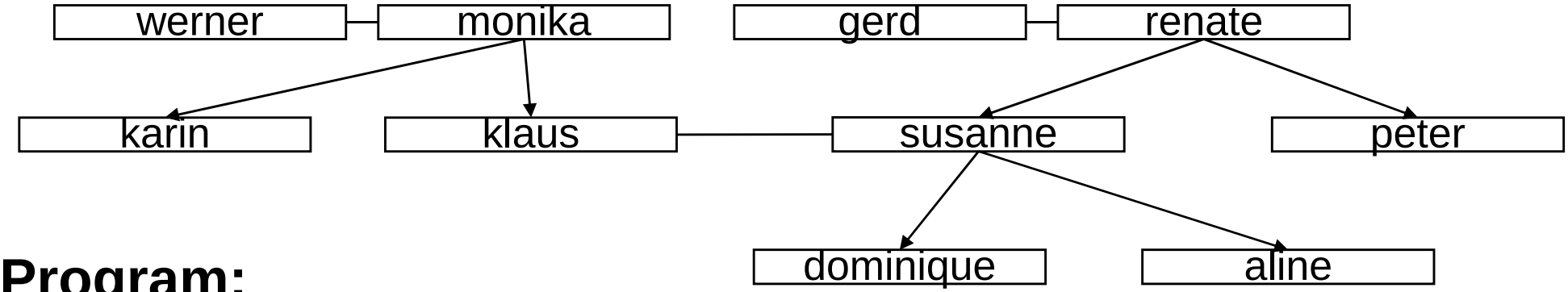
```
female(monika).
female(karin).
female(renate).
female(susanne).
female(aline).

married(werner, monika).
married(gerd, renate).
married(klaus, susanne).

human(X).

male(werner).
male(klaus).
male(gerd).
male(peter).
male(dominique).

motherOf(monika, karin).
motherOf(monika, klaus).
motherOf(renate, susanne).
motherOf(renate, peter).
motherOf(susanne, aline).
motherOf(susanne, dominique).
```

```
?- male(gerd).
true
```

```
?- married(gerd, monika).
false
```

```
?- human(gerd).
true
```

Variables in Queries



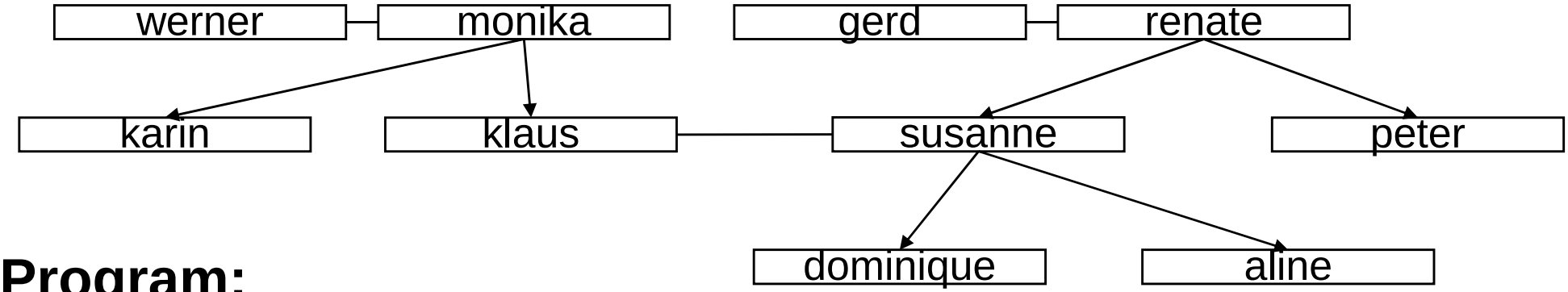
Program:

```
female(monika).                    male(werner).
...                                ...
female(aline).                     male(dominique).
married(werner, monika).           motherOf(monika, karin).
...                                ...
married(klaus, susanne).           motherOf(susanne, dominique).
```

```
?- motherOf(X, susanne).
X = renate
```

```
?- motherOf(renate, Y).
Y = susanne ;
Y = peter
```

Combined Queries

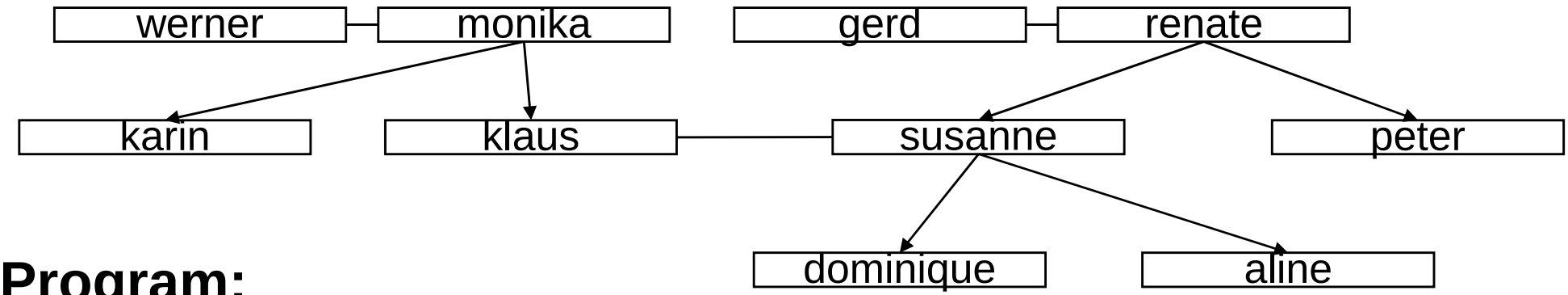


Program:

```
female(monika).                    male(werner).
...                                ...
female(aline).                     male(dominique).
married(werner, monika).           motherOf(monika, karin).
...                                ...
married(klaus, susanne).           motherOf(susanne, dominique).
```

```
?- married(gerd,W), motherOf(W,susanne).
W = renate

?- motherOf(Grandma,Mom), motherOf(Mom,aline).
Grandma = renate
Mom = susanne
```

Rules



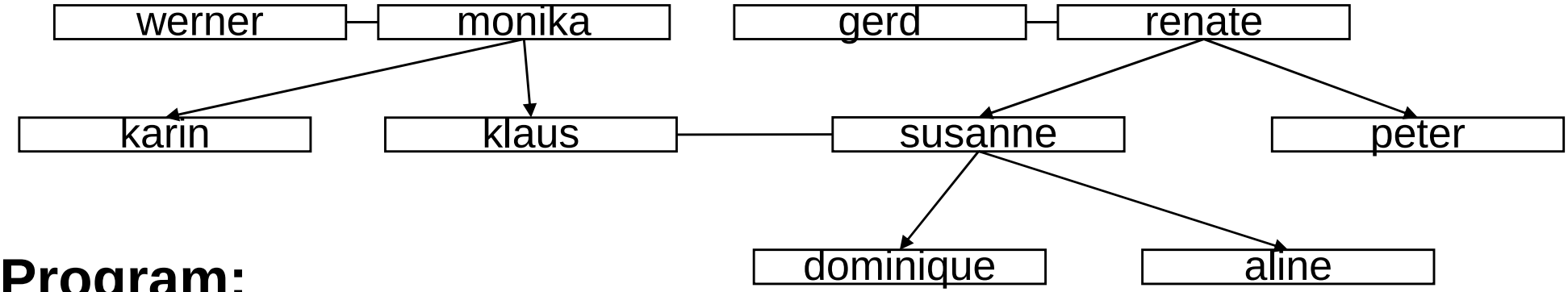
Program:

```
female(monika).                    male(werner).
...                                ...
female(aline).                     male(dominique).
married(werner, monika).           motherOf(monika, karin).
...                                ...
married(klaus, susanne).           motherOf(susanne, dominique).

fatherOf(F,C)    :- married(F,W), motherOf(W,C).
```

```
?- fatherOf(gerd, susanne).
true
```

```
?- fatherOf(gerd, Y).
Y = susanne ;
Y = peter
```

Several Rules for one Predicate

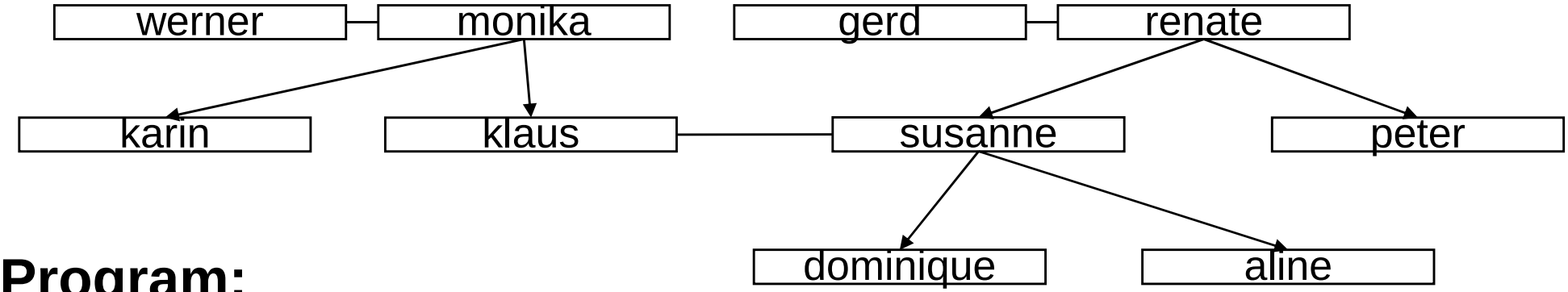


Program:

```
female(monika).                    male(werner).
...                                ...
female(aline).                     male(dominique).
married(werner, monika).           motherOf(monika, karin).
...                                ...
married(klaus, susanne).           motherOf(susanne, dominique).
fatherOf(F,C)    :- married(F,W), motherOf(W,C).

parent(X, Y)     :- motherOf(X,Y).
parent(X, Y)     :- fatherOf(X,Y).
```

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

X = renate  ;

X = gerd
```

Recursive Rules



Program:

```
female(monika).                    male (werner).
...                                ...
married(klaus, susanne).           motherOf(susanne, dominique).
fatherOf(F,C)    :- married(F,W), motherOf(W,C).

parent(X, Y)    :- motherOf(X,Y).
parent(X, Y)    :- fatherOf(X,Y).

ancestor(V,X) :- parent(V,X).
ancestor(V,X) :- parent(V,Y), ancestor(Y,X).
```

```
?- ancestor(X, aline).
```

X = susanne ; X = klaus ;

X = monika ; X = renate ; X = werner ; X = gerd