

# ЛЕКЦИЯ 2

## МАШИНА ВЫВОДА ПРОЛОГА

---

# Механизмы машины вывода Пролога

- Унификация
- Откат

# Унификация

Для  $p(x), p(y)$  унификаторы:  $\{x = y\}, \{x = a, y = a\}$

Для  $\{x = y\}, \{y = a\}$  композиция:  $\{x = a, y = a\}$

Для  $p(x), p(y)$  наибольший общий унификатор  $\theta = \{x = y\}$   
 $\theta_1 = \{x = a, y = a\}, \theta_2 = \{y = a\}, \theta\theta_2 = \theta_1$

Для формул  $p(x, f(g(y, h(z)), b))$  и  
 $p(x, f(g(y, a), c))$  множество рассогласований:  $\{h(z), a\}$

# Алгоритм поиска наибольшего общего унификатора

$$\theta_0 = \emptyset, A_0 = A, B_0 = B, k = 0$$

пока( $A_k \neq B_k$ )

1)  $D_k$  = множество рассогласований формул  $A_k, B_k$

2) если  $D_k$  содержит переменную  $x$  и терм  $t$ ,

в который не входит переменная  $x$ ,

то  $\sigma_k = \{x = t\}$  (подстановка)

иначе

формулы  $A$  и  $B$  не унифицируемы.

3)  $A_{k+1} = A_k \sigma_k, B_{k+1} = B_k \sigma_k, \theta_{k+1} = \theta_k \sigma_k, k = k + 1.$

# Пример работы алгоритма

$$A = p(x, x, f(g(a))), B = p(y, b, f(z))$$

(переменные  $x, y$  и  $z$  и константы  $a$  и  $b$ ).

$$D_0 = \{x, y\}.$$

$$\sigma_0 = \{x = y\}$$

$$A_0 = p(y, y, f(g(a))), B_0 = p(y, b, f(z)).$$

$$D_1 = \{y, b\}$$

$$\sigma_1 = \{y = b\}$$

$$A_1 = p(b, b, f(g(a))), B_1 = p(b, b, f(z)).$$

$$D_2 = \{g(a), z\}$$

$$\sigma_2 = \{z = g(a)\}$$

$$A_2 = B_2 = p(b, b, f(g(a)))$$

$$\theta = \{x = y\}\{y = b\}\{z = g(a)\} = \{x = b, y = b, z = g(a)\}.$$

# Процедурная семантика логической программы

$Q = ? - C_1, \dots, C_{i-1}, C_i, C_{i+1}, \dots, C_m$  — запрос

$B_0: - B_1, \dots, B_n$  — вариант  $D$  правила  $A_0: - A_1, \dots, A_n$ ,

нет совпадающих переменных, и

$\theta$  — наибольший общий унификатор формул  $C_i$  и  $B_0$ .

SLD-резольвента запроса  $Q$  и правила  $D$  с подстановкой  $\theta$  - запрос

$$Q = ? - (C_1, \dots, C_{i-1}, B_1, \dots, B_n, C_{i+1}, \dots, C_m)\theta.$$

Частичное SLD-резольтивное вычисление -  $\{D_i, \theta_i, Q_i\}$

# Пример

млекопитающее("слон").

млекопитающее("зебра").

животное("страус").

животное("уж").

животное( $X$ ):- млекопитающее( $X$ ).

Цель  $Q_0 = ?$  – животное(слон).

Правило  $D_0 =$  животное( $A$ ):- млекопитающее( $A$ ).

Подстановка  $\theta_0 = \{A = \text{слон}\}$

SLD-резольвента  $Q_1 = ?$  – млекопитающее(слон).

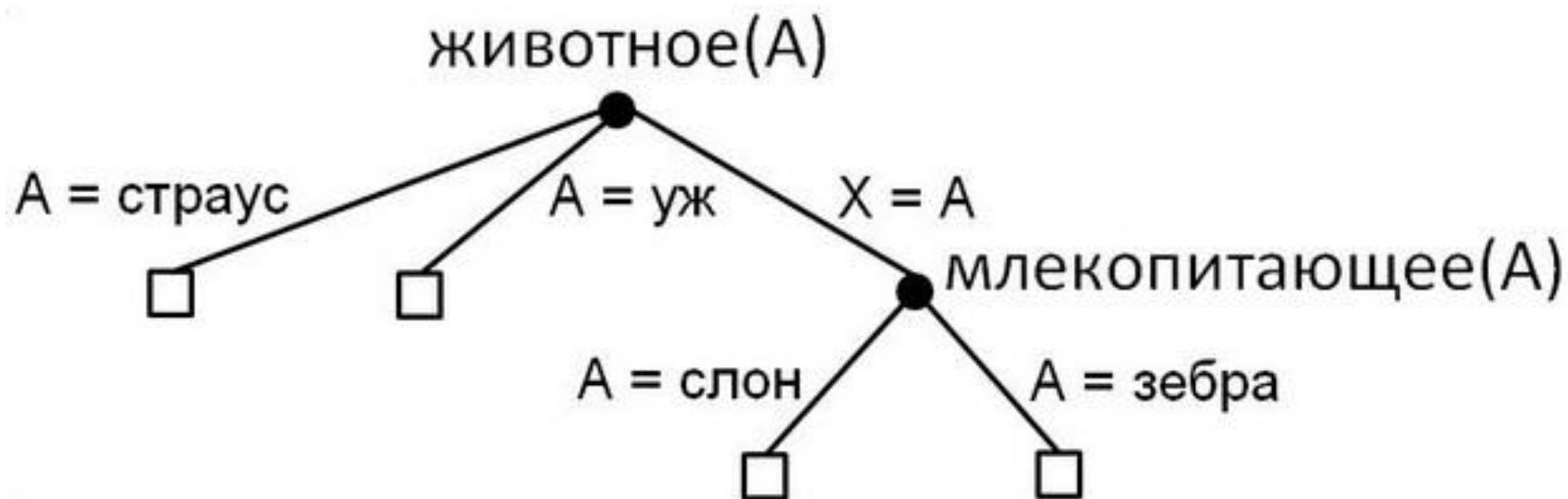
Правило  $D_1 = ?$  – млекопитающее(слон)

Подстановка  $\theta_1 = \{\emptyset\}$

SLD-резольвента  $Q_2 = \emptyset$ .

Таким образом, цель животное(слон) выводится из программы.

# Пространство вычислений





# Устройство вычислений в Прологе

```
родитель("Иван", "Мария").  
родитель("Анна", "Мария").  
родитель("Мария", "Павел").  
родитель("Мария", "Петр").
```

```
женщина("Мария").  
женщина("Анна").
```

```
мать(X, Y):- женщина(X), родитель(X, Y).
```

Цель  $Q$  = `мать(X,Y)`.

# Устройство вычислений в Прологе

```
родитель("Иван", "Мария").  
родитель("Анна", "Мария").  
родитель("Мария", "Павел").  
родитель("Мария", "Петр").
```

```
женщина("Мария").  
женщина("Анна").
```

```
мать(X, Y):- женщина(X), родитель(X, Y).
```

Цель  $Q$  = мать( $X, Y$ ).

$X$  = Мария,  $Y$  = Павел

$X$  = Мария,  $Y$  = Петр

$X$  = Анна,  $Y$  = Мария

# Устройство вычислений в Прологе

```
родитель("Иван", "Мария").  
родитель("Анна", "Мария").  
родитель("Мария", "Павел").  
родитель("Мария", "Петр").
```

```
женщина("Мария").  
женщина("Анна").
```

```
мать(X, Y):- женщина(X), родитель(X, Y).
```

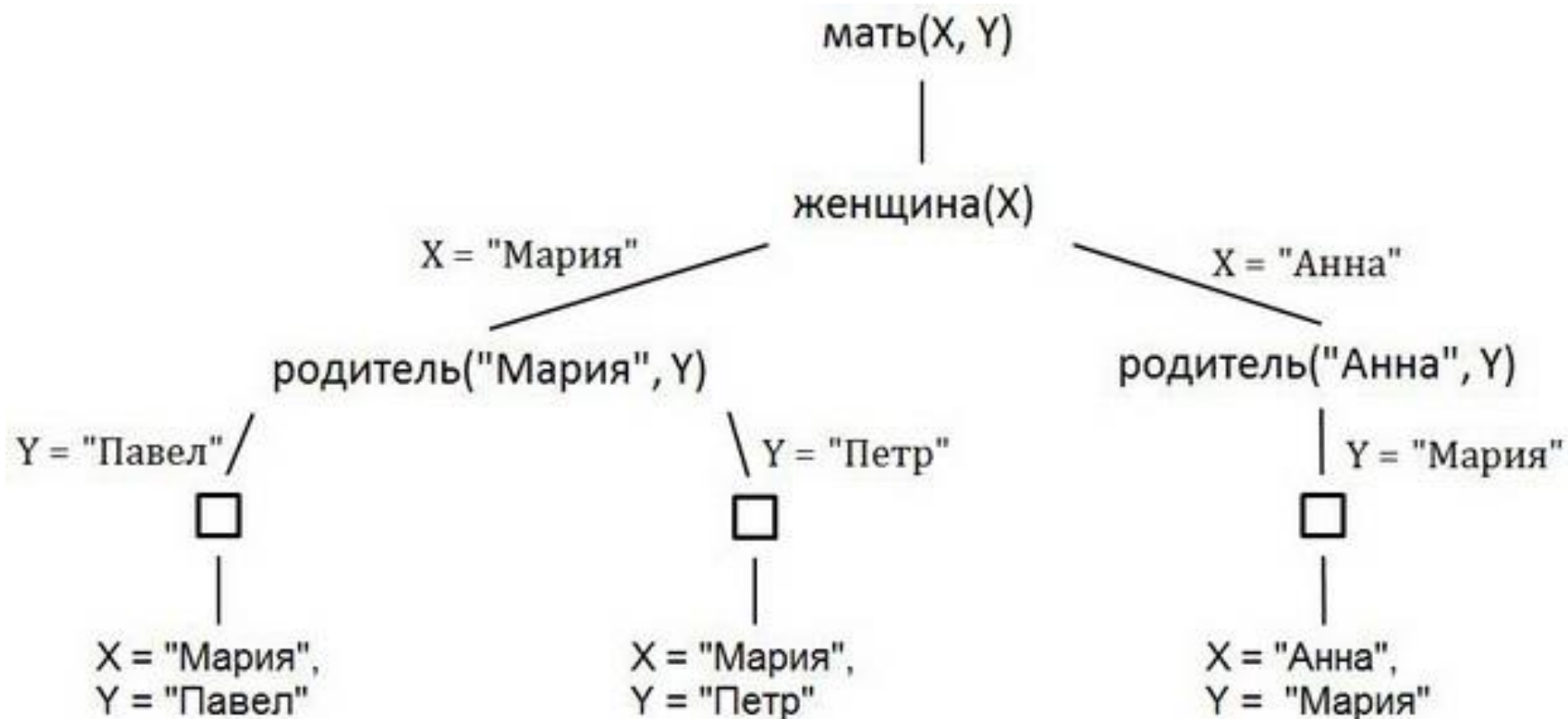
Цель  $Q = \text{мать}(X, Y) \Rightarrow \text{женщина}(X), \text{родитель}(X, Y).$

$X = \text{Мария}, Y = \text{Павел}$

$X = \text{Мария}, Y = \text{Петр}$

$X = \text{Анна}, Y = \text{Мария}$

# Дерево поиска



# Устройство вычислений в Прологе

мужчина("Иван").

мужчина("Павел").

мужчина("Петр").

женщина("Мария").

женщина("Анна").

Цель  $Q$  = женщина( $X$ ); мужчина( $X$ ).

$X$  = Мария

$X$  = Анна

$X$  = Иван

$X$  = Павел

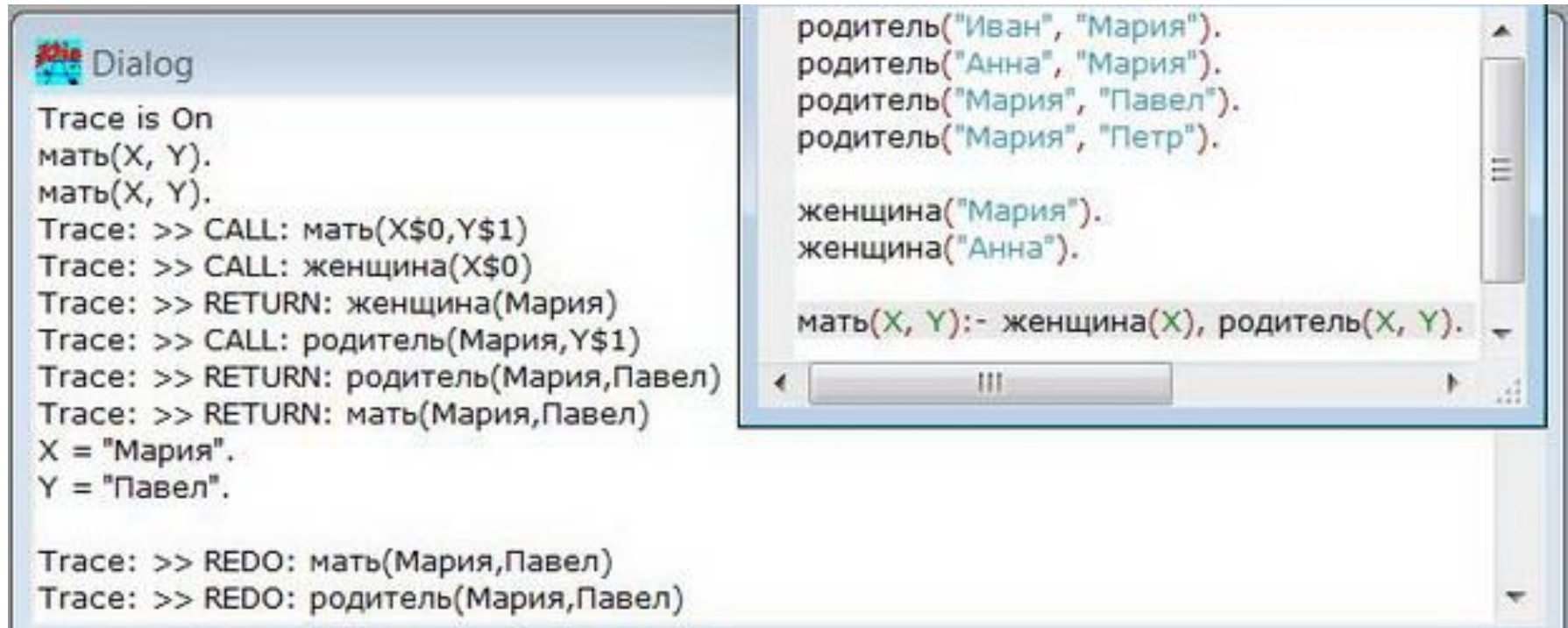
$X$  = Петр

# Трассировка в PIE

## Engine > Trace Calls

- CALL — вызов цели;
- RETURN — завершение (возврат результата);
- REDO — откат (возврат по успеху для поиска других решений);
- FAIL — возврат по неудаче.

# Трассировка в PIE



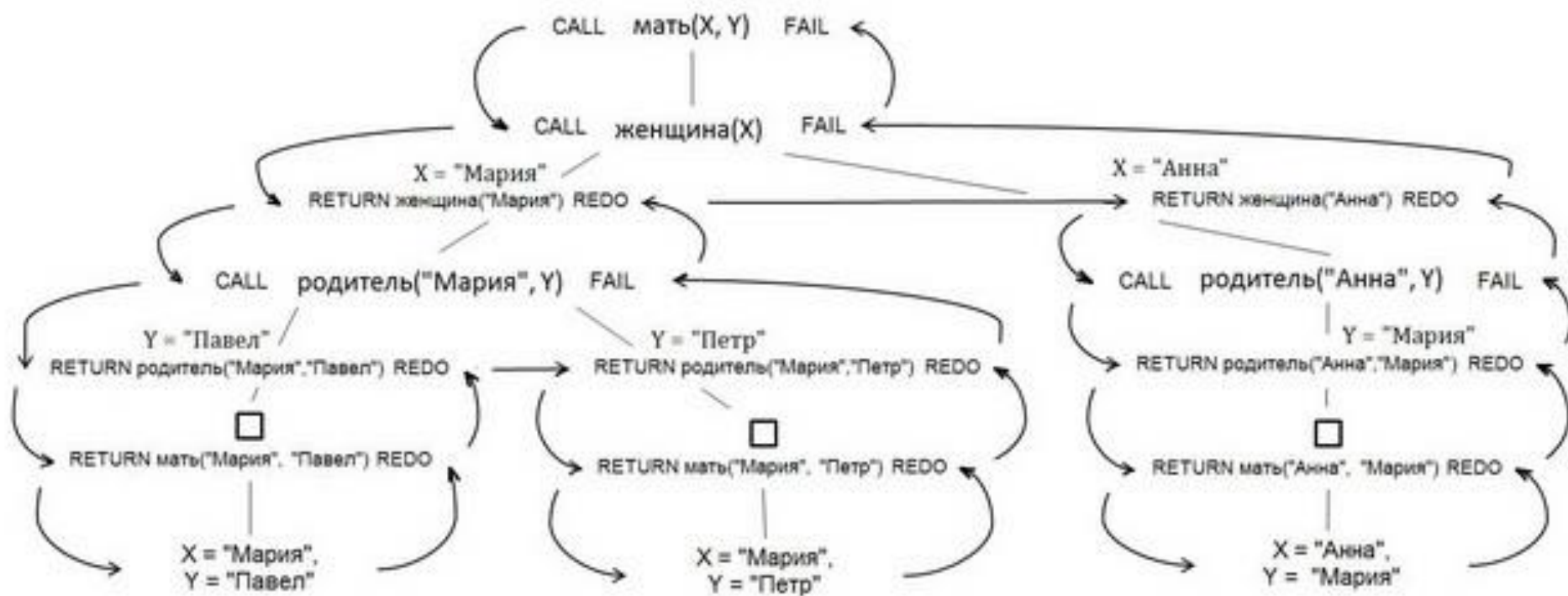
**Dialog**

Trace is On  
мать(X, Y).  
мать(X, Y).  
Trace: >> CALL: мать(X\$0,Y\$1)  
Trace: >> CALL: женщина(X\$0)  
Trace: >> RETURN: женщина(Мария)  
Trace: >> CALL: родитель(Мария,Y\$1)  
Trace: >> RETURN: родитель(Мария,Павел)  
Trace: >> RETURN: мать(Мария,Павел)  
X = "Мария".  
Y = "Павел".

Trace: >> REDO: мать(Мария,Павел)  
Trace: >> REDO: родитель(Мария,Павел)

```
родитель("Иван", "Мария").  
родитель("Анна", "Мария").  
родитель("Мария", "Павел").  
родитель("Мария", "Петр").  
  
женщина("Мария").  
женщина("Анна").  
  
мать(X, Y):- женщина(X), родитель(X, Y).
```

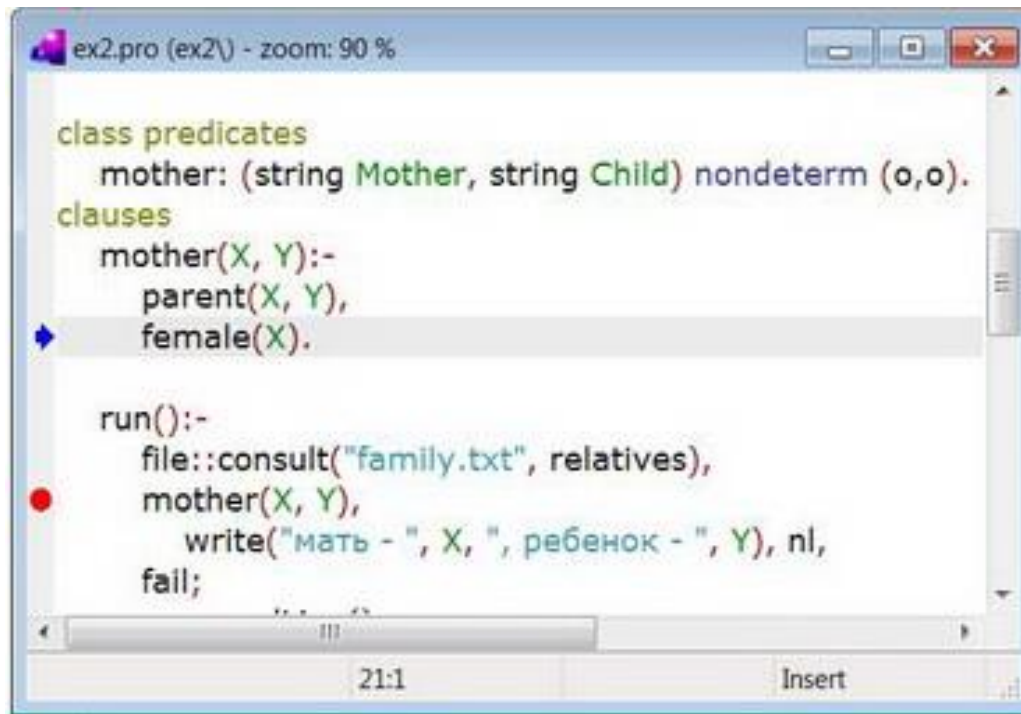
# Трассировка в РІЕ



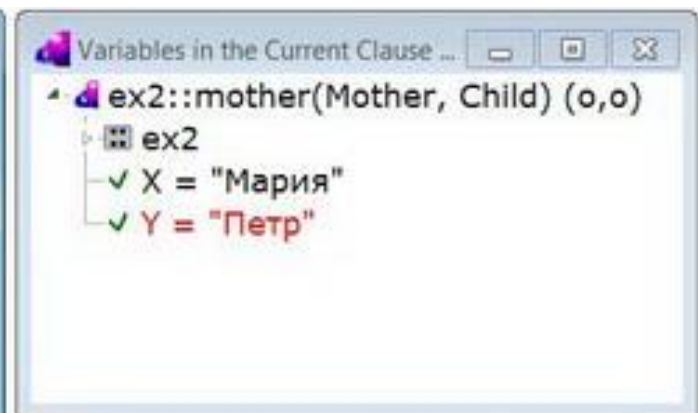


# Отладчик

Debug > Run



```
ex2.pro (ex2\) - zoom: 90 %  
  
class predicates  
  mother: (string Mother, string Child) nondeterm (o,o).  
clauses  
  mother(X, Y):-  
    parent(X, Y),  
    female(X).  
  
run():-  
  file::consult("family.txt", relatives),  
  mother(X, Y),  
  write("мать - ", X, ", ребенок - ", Y), nl,  
  fail;
```



```
Variables in the Current Clause ...  
  
ex2::mother(Mother, Child) (o,o)  
  ex2  
    ✓ X = "Мария"  
    ✓ Y = "Петр"
```



```
Run Stack - zoom: 90 %  
  
ex2::mother(Mother, Child) (o,o)  
ex2::run()  
goal()  
VIP::Kernel::RunTrappedProgram
```

# Сложные термы

*Список* – это конечная последовательность элементов

$\text{cons}(1, \text{cons}(2, \text{cons}(3, \text{nil})))$ ,

где функтор *nil* - пустой список.

$[1, 2, 3]$

Пустой список  $[]$ .

$[H | T]$

$[_ , _]$  унифицируется с любым списком, состоящим ровно из двух элементов,

$[_ | _]$  с любым непустым списком.

# Пример «Библиотека»

```
publication = book(author, string Название, edition Издание);  
    magazine(string Название, integer Номер, integer Год).  
author = author(string Фамилия, string Имя, string Отчество).  
edition = edition(string Место, string Издательство, integer  
Год).
```

```
class facts
```

```
    library: (publication).
```

```
clauses
```

```
    library(magazine("Компьютерра", 2, 2009)).
```

```
    library(magazine("Наука и жизнь", 11, 2012)).
```

```
    library(book(author("Чехов", "Антон", "Павлович"),  
        "Избранное", edition("Москва", "АСТ, Астрель", 2003))).
```

```
    library(book(author("Великова", "Людмила", "Викторовна"),  
        "Русский язык", edition("Москва", "МЦНМО", 2003))).
```

# Пример «Библиотека»

```
run():-
```

```
    % Что есть в библиотеке?
```

```
    library(X),
```

```
        write(X), nl,
```

```
    fail;
```

```
    % Названия книг, изданных в 2003 году
```

```
    library(book(_, Title, edition(_, _, 2003))),
```

```
        write(Title), nl,
```

```
    fail;
```

```
    _ = readLine().
```

# Пример «Иностранные языки»

```
class facts
```

```
    knows: (string, string*).
```

```
clauses
```

```
    knows("Даша", ["английский", "испанский", "французский"]).
```

```
    knows("Маша", ["немецкий", "английский"]).
```

```
    knows("Глаша", ["английский", "немецкий"]).
```

```
    knows("Паша", ["английский"]).
```

```
run():-
```

```
    knows(X, Y),
```

```
        write(X, " - ", Y), nl,
```

```
    fail;
```

```
    _ = readLine().
```

## Условные выражения. Знак равенства

$f(x, y, \dots, z, u, \dots) \leq f(x, y, \dots, z, v, \dots)$ , если  $u \leq v$ .

$\text{tuple}(\underline{5}, 8) < \text{tuple}(\underline{6}, 4)$  и  $[\underline{4}, 5] > [\underline{1}, 2, 3]$

$\text{term}_1 = \text{term}_2$

$2 = X, \text{tuple}(Y, \text{tuple}(4, X)) = \text{tuple}(3, Z)$

$X = 2, Y = 3, Z = \text{tuple}(4, 2)$

# Отрицание

супруг ("Иван", "Анна").

мужчина ("Иван").

мужчина ("Петр").

мужчина ("Степан").

? - мужчина(X), not(супруг(X, \_)).

# Пример «Студенты»

domains

date = date(integer День, integer Месяц, integer Год).

class facts

dateOfBirth: (string Имя, date ДатаРождения).

clauses

dateOfBirth("Елизавета", date(2, 5, 1999)).

dateOfBirth("Тимофей", date(10, 10, 2000)).

dateOfBirth("Даниил", date(25, 2, 2000)).

% ...

class predicates

age: (string Name, integer Age) nondeterm (o,o).

youngestPerson: (string Name, integer Age) nondeterm (o,o).



# Пример «Студенты»

clauses

```
age(Name, Age):-  
    Time = time::new(),  
    Time:getDate(CurrentYear, _M, _D),  
    dateOfBirth(Name, date(_, _, YearOfBirth)),  
    Age = CurrentYear - YearOfBirth.
```

```
youngestPerson(Name, Age):-  
    age(Name, Age),  
    not((age(_, X), X < Age)).
```

```
run():-  
    youngestPerson(Name, Age),  
        write(Name, " - ", Age), nl,  
    fail;  
    _ = readLine().
```

# Пример «Родственные отношения 2»

```
class facts - relatives
```

```
    parent: (string Родитель, string Ребенок).
```

```
    spouse: (string Муж, string Жена).
```

```
    male: (string).
```

```
    female: (string).
```

```
class predicates
```

```
    sister: (string Сестра, string Чья) nondeterm (o,o).
```

```
    bloodSister: (string Сестра, string Чья) nondeterm (o,o).
```

```
    halfSister: (string Сестра, string Чья) nondeterm (o,o).
```

```
    haveCommonFather: (string, string) nondeterm anyflow.
```

```
    haveCommonMother: (string, string) nondeterm anyflow.
```

# Пример «Родственные отношения 2»

clauses

```
sister(X, Y):-  
    bloodSister(X, Y);  
    halfSister(X, Y).
```

```
bloodSister(X, Y):-  
    female(X),  
    haveCommonFather(X, Y),  
    haveCommonMother(X, Y).
```

```
halfSister(X, Y):-  
    female(X),  
    (haveCommonFather(X, Y),  
    not(haveCommonMother(X, Y)));  
    haveCommonMother(X, Y),  
    not(haveCommonFather(X, Y))).
```

```
haveCommonFather(X, Y):-  
    male(Z),  
    parent(Z, X),  
    parent(Z, Y),  
    X <> Y.
```

```
haveCommonMother(X, Y):-  
    female(Z),  
    parent(Z, X),  
    parent(Z, Y),  
    X <> Y.
```

# Пример «Родственные отношения 2»

```
run():-
```

```
    file::consult("family.txt", relatives),  
    sister(X, Y),  
        write(X, " - сестра для - ", Y), nl,  
    fail;  
    _ = readLine().
```