Сошников Дмитрий Валерьевич
к.ф.-м.н., доцент
http://soshnikov.com

Лекция 2: Суть логического программирования

Логическое программирование

https://soshnikov.com/courses/logpro/

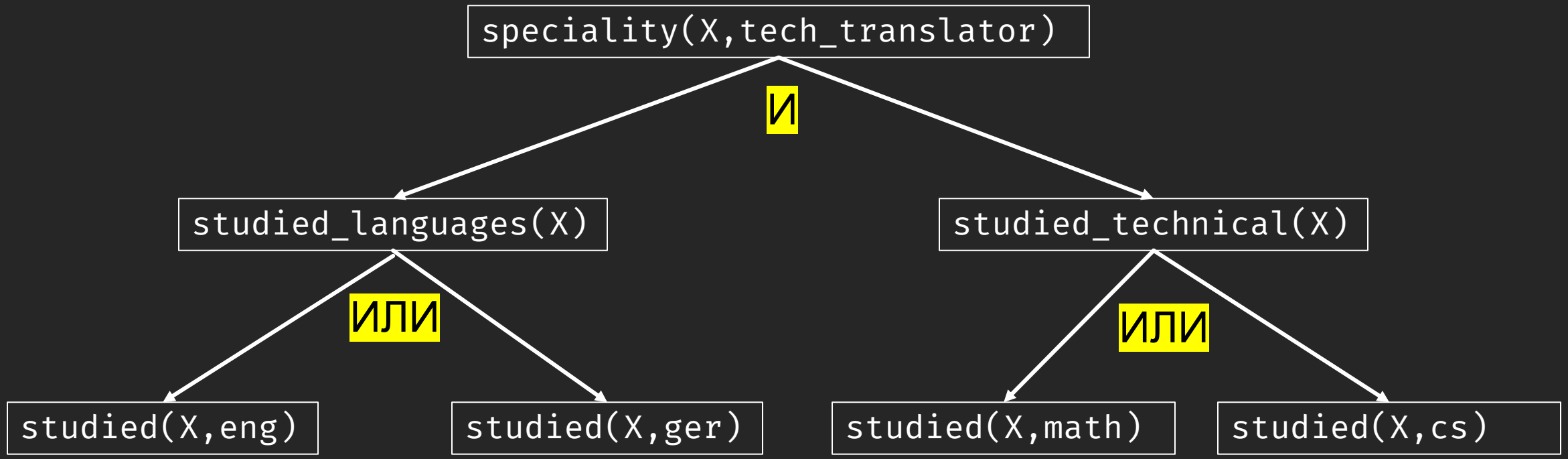
Рассмотрим пример

```
speciality(X,tech_translator) :-
      studied_languages(X),studied_technical(X).
speciality(X,programmer) :-
      studied(X,mathematics),studied(X, compscience).
speciality(X,lit_translator) :-
studied_languages(X),studied(X,literature).

studied_technical(X) :- studied(X,mathematics).
studied_technical(X) :- studied(X,compscience).
studied_languages(X) :- studied(X,english).
studied_languages(X) :- studied(X,german).

studied(petya,mathematics).     studied(vasya,german).
studied(petya,compscience).     studied(vasya,literature).
studied(petya,english).
```

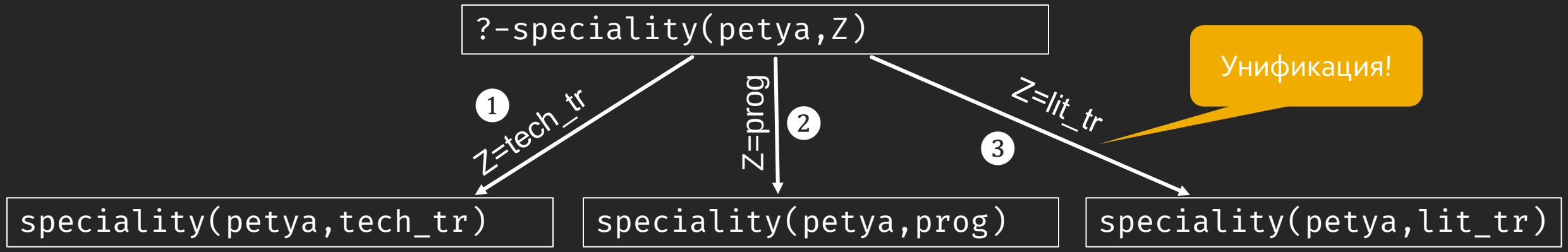
Правила

Факты

Дерево И-ИЛИ


speciality(X,tech_translator)
И
studied_languages(X)
studied_technical(X)
ИЛИ
ИЛИ
studied(X,eng)
studied(X,ger)
studied(X,math)
studied(X,cs)

Дерево решений




```
speciality(X,tech_translator) :-
        studied_languages(X),
        studied_technical(X).
speciality(X,programmer) :-
       studied(X,mathematics),
        studied(X, compscience).
speciality(X,lit_translator) :-
        studied_languages(X),
        studied(X,literature).
```

1

2

3

Поиск в дереве решений

Доказываем подцели слева направо!

?-speciality(petya,Z)

Z=tech_tr

```
speciality(X,tech_translator) :-
        studied_languages(X),
        studied_technical(X).
```

st_lang(petya), st_tech(petya)

```
st_lang(X) :- st(X,eng).
```

st(petya,eng), st_tech(petya)

```
studied(petya,english).
```

st_tech(petya)

```
st_tech(X):-studied(X,math).
```

```
st_tech(X):-studied(X,cs).
```

st(petya,math)

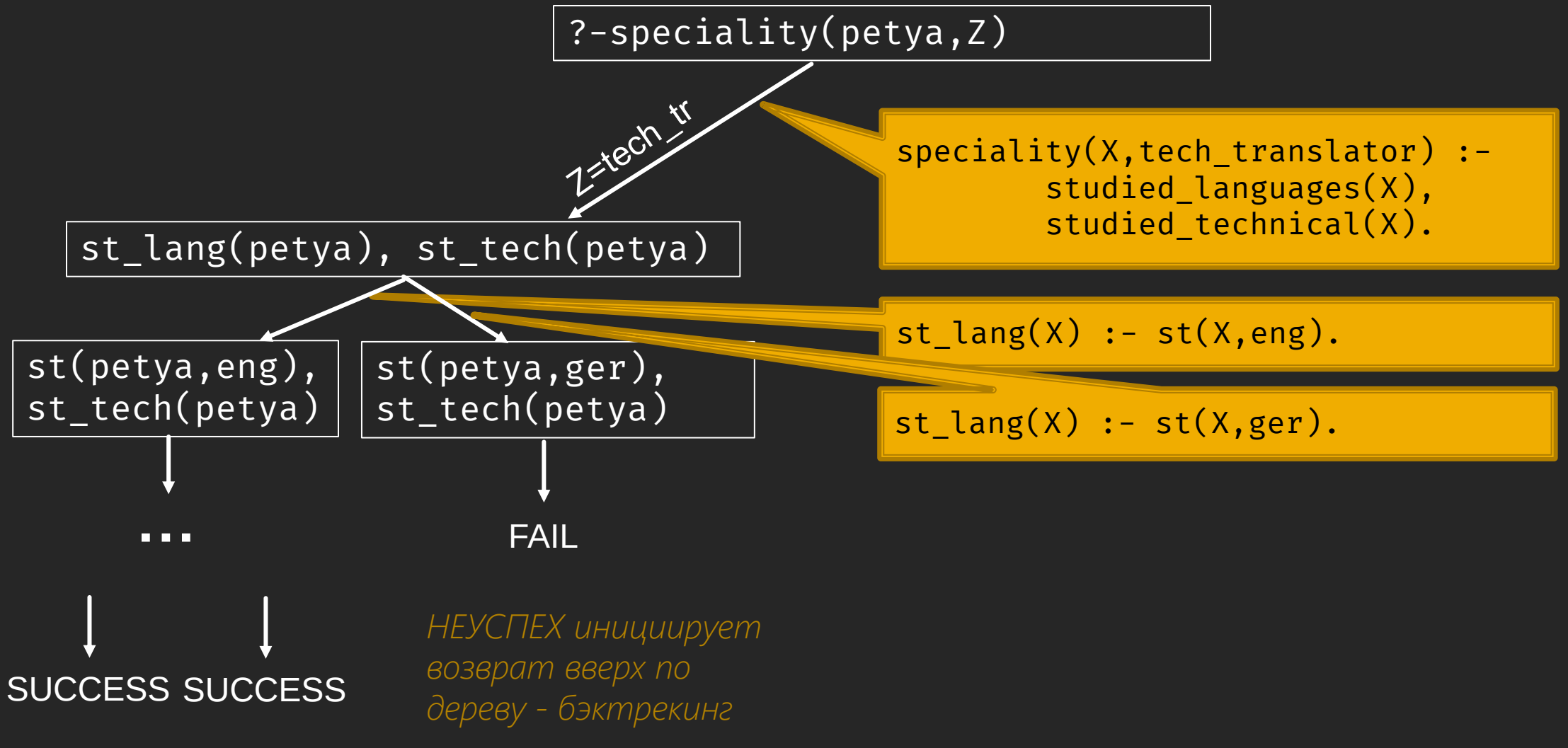
st(petya,cs)

```
studied(petya,math).
```

SUCCESS

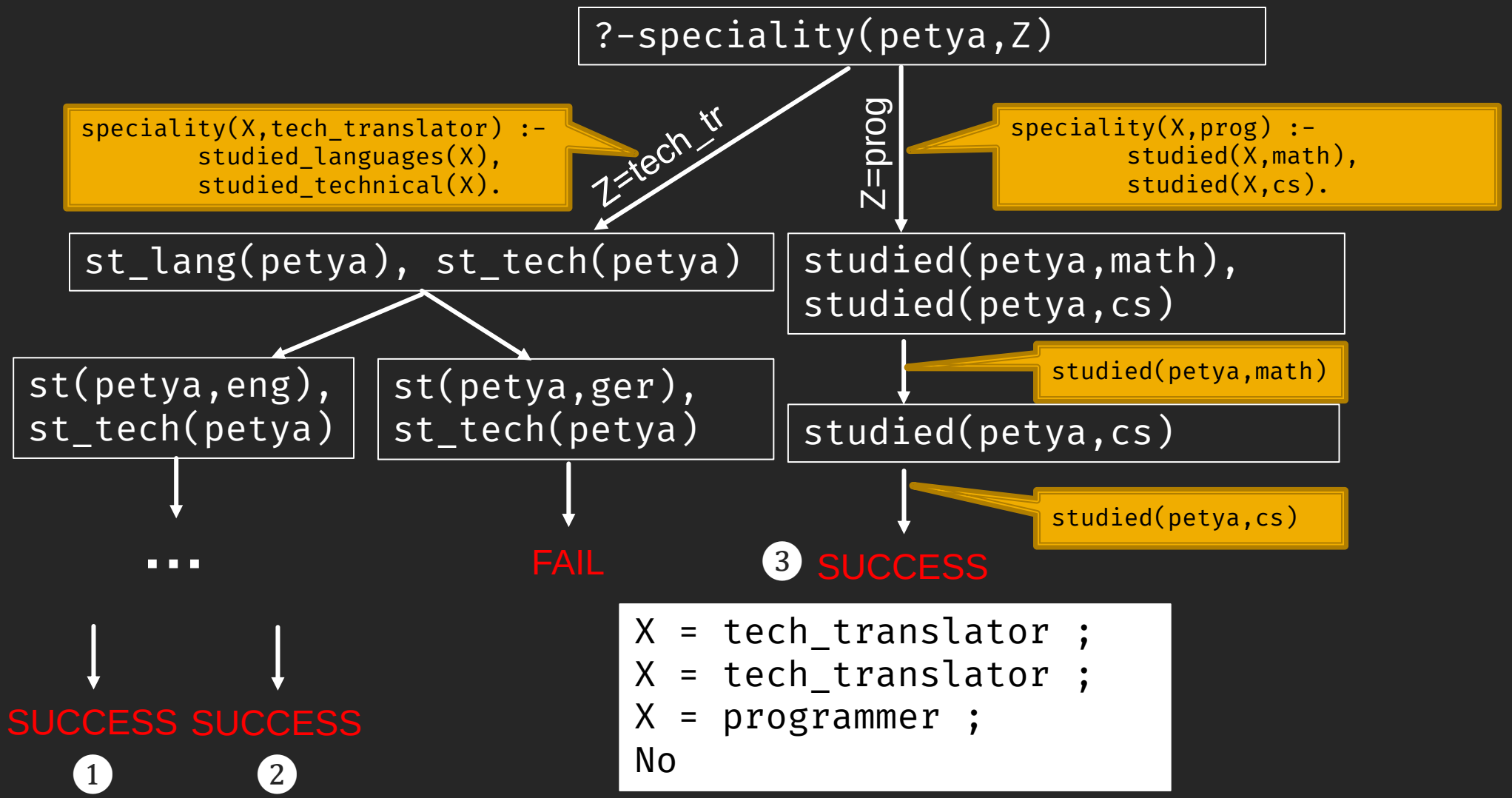
SUCCESS

Поиск в дереве решений



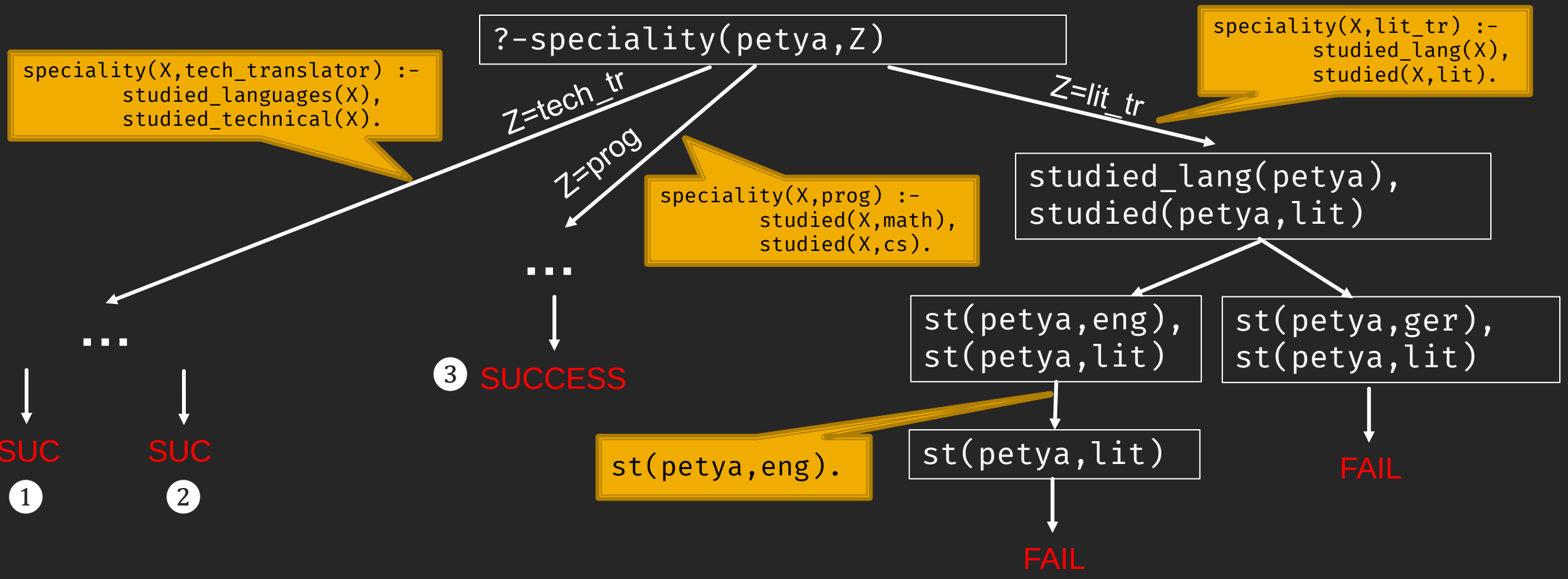
НЕУСПЕХ инициирует возврат вверх по дереву - бэктрекинг

Возврат (бэктренинг)




```
X = tech_translator ;
X = tech_translator ;
X = programmer ;
No
```

Возврат (бэктренинг)



Механизм работы логического интерпретатора

- Запрос (целевое утверждение) сопоставляется (унифицируется) с головами имеющихся в программе правил и фактов.
- Начиная с первого найденного правила, целевое утверждение подменяется правой частью правила (с учетом замены переменных)
- Если встречается неуспех (правило не находится), то происходит откат (backtracking)

Задание 1

- Построить дерево вывода для следующих запросов:

- `?- speciality(X,tech_translator).`
- `?- speciality(petya, lit_translator).`
- `?- speciality(petya, programmer).`
- `?- speciality(X,Y).`

Какое отношение это имеет к логике?

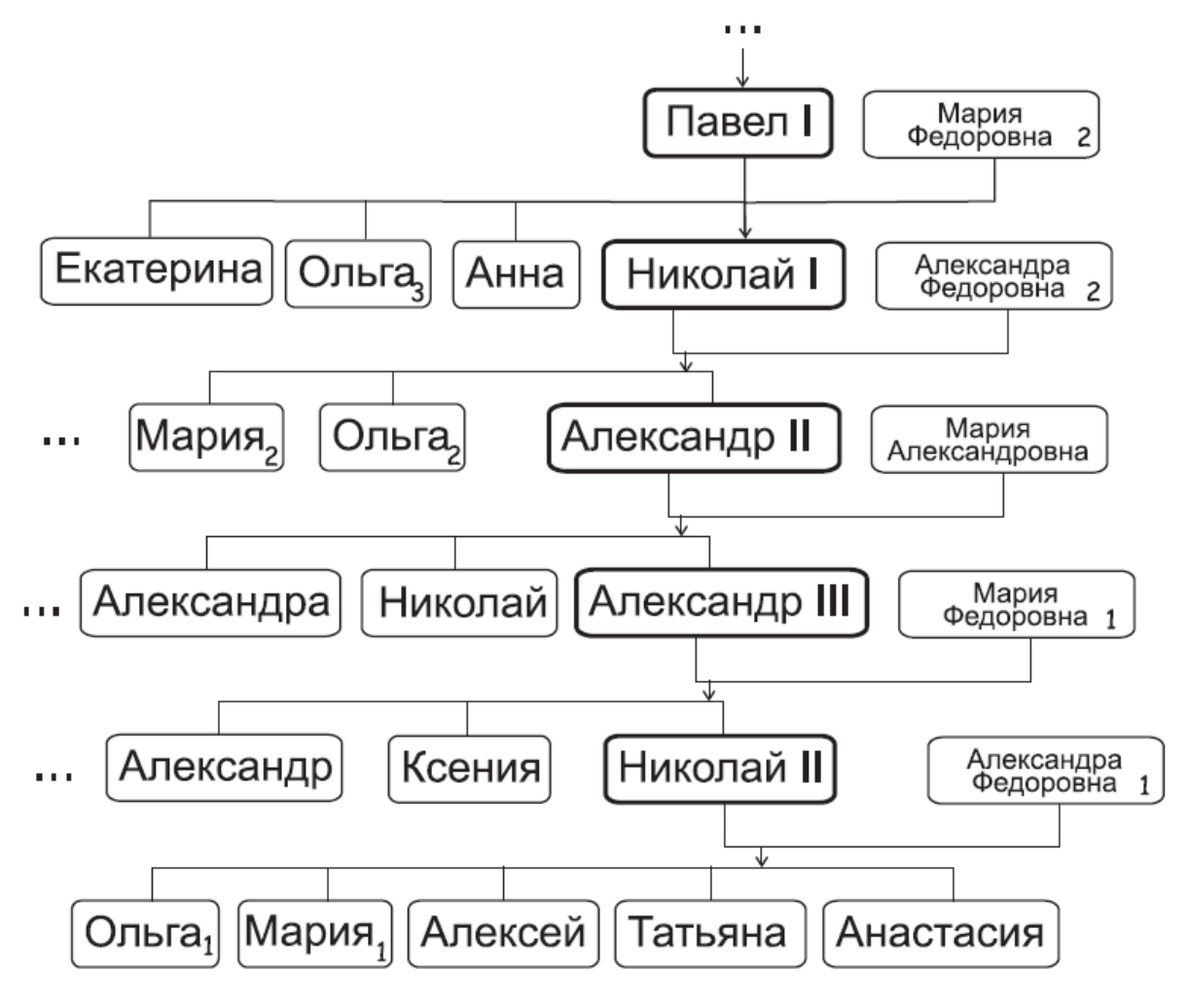
```
speciality(X,tech_translator) :-
      studied_languages(X),studied_technical(X).
```

$$(\forall X) \quad \texttt{specialty}(X, \texttt{tech\_translator}) \subset \\ \subset \texttt{studied\_languages}(X) \wedge \texttt{studied\_technical}(X)$$

Дедуктивные базы данных

- В приведенном выше примере можно условно выделить базу фактов (кто какой предмет изучал) и базу правил
- Дедуктивные базы данных – это базы данных, снабженные средствами логического программирования для вывода дополнительных фактов
- Примеры: Mercury, Datalog

Классический пример



Как можно описать предметную область?

```
father(nicholas_ii,olga_1).
father(nicholas_ii,tatyana).
mother(alexandra_fedorovna_1,olga_1).
mother(alexandra_fedorovna_1,tatyana).
```

```
parent(nicholas_ii,olga_1).
parent(alexandra_fedorovna_1,olga_1).
parent(nicholas_ii,tatyana).
parent(alexandra_fedorovna_1,tatyana).
male(nicholas_ii).
female(alexandra_fedorovna_1).
father(X,Y) :- parent(X,Y), male(X).
mother(X,Y) :- parent(X,Y), female(X).
```

```
parents(nicholas_ii,alexandra_fedorovna_1,olga_1).
parents(nicholas_ii,alexandra_fedorovna_1,tatyana).
```

Какие достоинства и недостатки у каждого из таких способов описания?

Реляционные базы данных

Отец

Родитель	Ребенок
Николай 2	Ольга 1
Николай 2	Мария 1
Александр 3	Александр
Александр 3	Ксения

Мать

Родитель	Ребенок
Алекс.Фед. 1	Ольга 1
Алекс.Фед. 1	Мария 1
Мария Фед. 1	Александр
Мария Фед. 1	Ксения

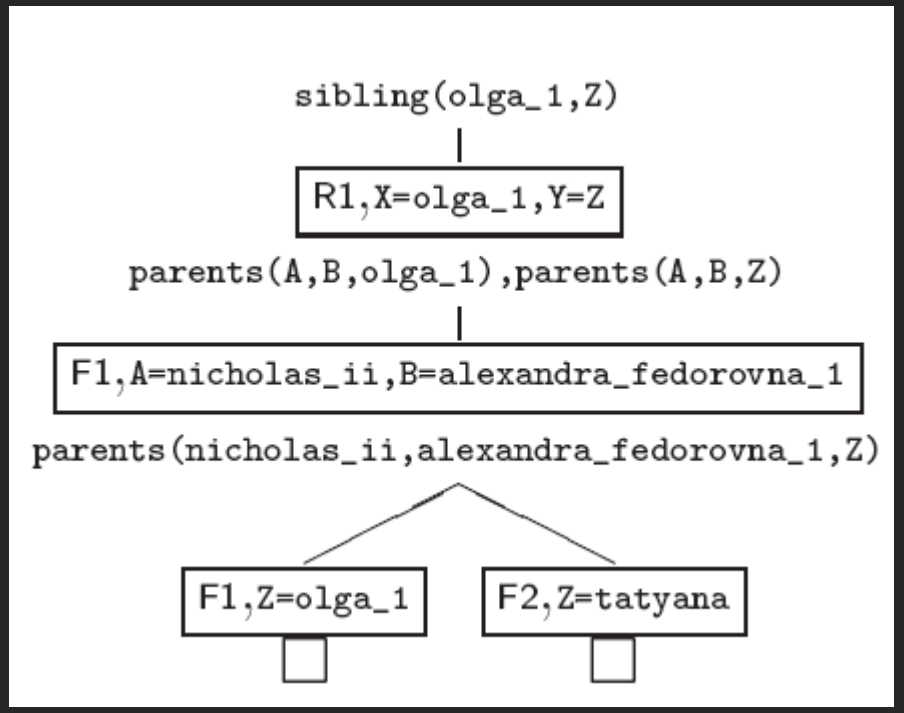
Родители

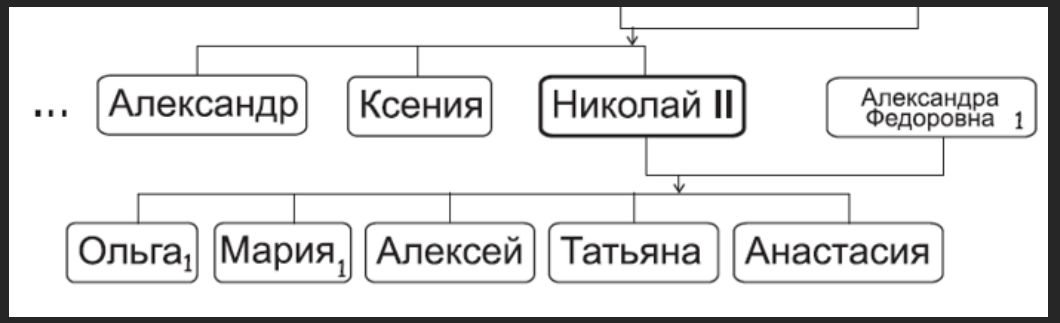
Отец	Мать	Ребенок
Николай 2	Алекс.Фед. 1	Ольга 1
Николай 2	Алекс.Фед. 1	Мария 1
Александр 3	Мария Фед. 1	Александр
Александр 3	Мария Фед. 1	Ксения

Описание отношений родства

```
sibling(X,Y) :- parents(A,B,X), parents(A,B,Y).
```

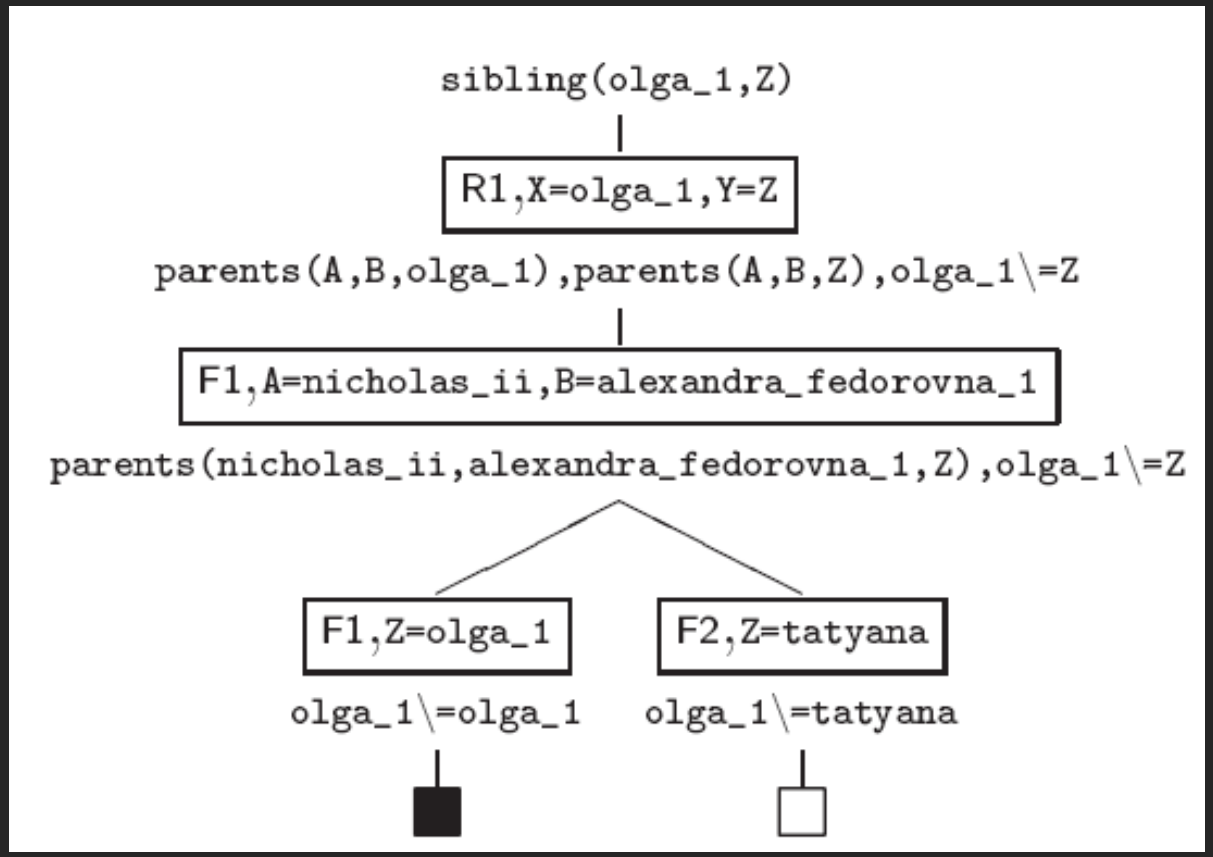
$$(\forall X)\ (\forall Y)\ (((\exists A)\ (\exists B)\ P(A,B,X) \wedge P(A,B,Y)) \supset S(X,Y))$$





Усовершенствованный пример

```
sibling(X,Y) :- parents(A,B,X), parents(A,B,Y), X\=Y.
```



Предки и потомки

Опишем предикат нахождения всех предков и/или потомков

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

- Рекурсия
- Один предикат может использоваться как для нахождения предков, так и для нахождения потомков

?-ancestor(alex3,X)

parent(alex3,X)

X=alex
X=xenia
X=nich2

parent(alex3,Z),
ancestor(Z,X)

ancestor(alex,X)

parent(alex,X)

FAIL

parent(alex3,Z),
ancestor(Z,X)

FAIL

ancestor(nich2,X)

parent(nich2,X)

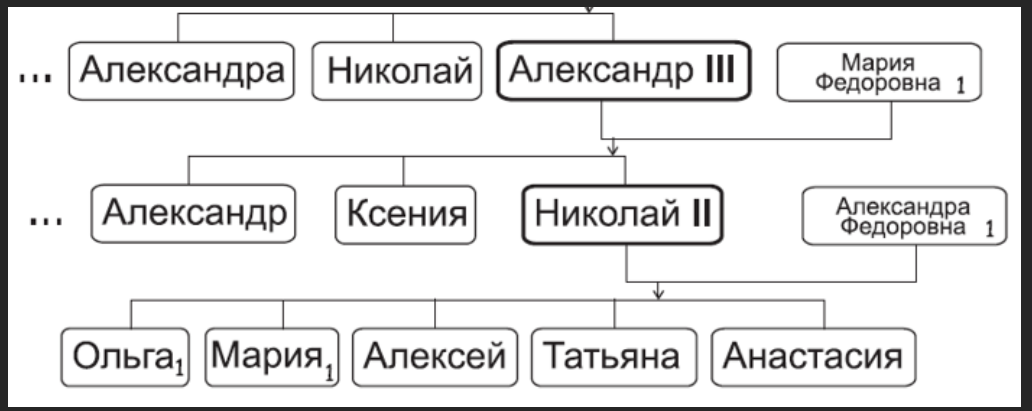
X=olga1
X=maria1
X=alexey
X=tatyana
X=anastasia

parent(nich2,Z1),
ancestor(Z1,X)

ancestor(olga1,X)

FAIL

FAIL




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

Задание 2

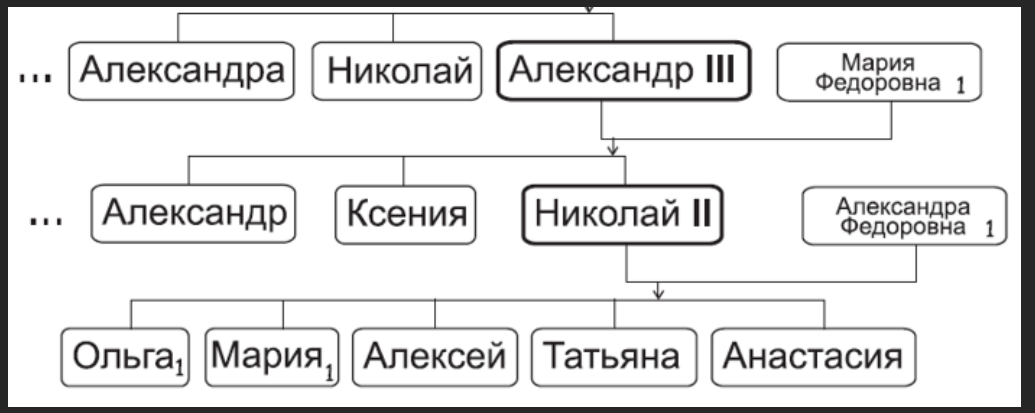
- Построить дерево вывода для запроса:
- `?- ancestor(X,olga1).`

Эквивалентная программа

- Логически-эквивалентная формула?

$$
\begin{aligned}
&(\forall X)(\forall Y) \\
&\quad (parent(X,Y) \vee \\
&\quad (\exists Z)\ parent(X,Z) \wedge ancestor(Z,Y)) \\
&\qquad\qquad\qquad\qquad \supset\ ancestor(X,Y))
\end{aligned}
$$

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




```
?-ancestor(alex3,X)
```

```
ancestor(Z,X),
parent(alex3,Z)
```

```
parent(alex3,X)
```

X=alex
X=xenia
X=nich2

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

```
ancestor(Z1,X),
parent(Z,Z1),
parent(alex3,Z)
```

```
parent(Z,X),
parent(alex3,Z)
```

Z=nich2
X=olga1

Z=nich2
X=maria1

```
ancestor(Z2,X),
parent(Z1,Z2),
parent(Z,Z1),
parent(alex3,Z)
```

FAIL

```
parent(nich2,olga1),
parent(alex3,nich2)
```

```
parent(nich2,maria1),
parent(alex3,nich2)
```

SUCCESS
X=olga1

SUCCESS
X=maria1

⋮

При другом порядке правил получается бесконечное дерево, которое тем не менее содержит все решения

Декларативная и процедурная семантика логических программ

- Декларативная семантика программы задается соответствующей формулой предикатов
  - т.е. соответствует «общечеловеческому» декларативному смыслу описания
- Процедурная семантика определяется механизмом вывода логического интерпретатора
- При реализации реальных программ приходится обращать внимание на декларативную и процедурную семантику

Рекомендации по написанию рекурсивных программ

- Располагать вначале нерекурсивное правило, затем рекурсивное
- Располагать рекурсивный вызов ближе к концу правила, чтобы к моменту вызова как можно больше переменных уже конкретизировалось.

Что дальше?

- Вспомним логику предикатов
- Посмотрим, как можно на ее основе создать систему программирования
  - (хотя сама логика – неразрешима!)
- Изучим различные стратегии логического вывода
- Синтаксис Пролога более формально
- Списки, деревья
- Логическое программирование с отрицанием
- Типизация, язык Mercury, доказательства программ и вывод типов
- Решение логических задач
- Решение задач методом поиска в пространстве состояний
- Приложения
  - компьютерная алгебра
  - анализ естественных и искусственных языка
  - Экспертные системы

Очень сложное задание (курсовой)

- Усовершенствуйте пример с родословным деревом, чтобы можно было для любых двух человек определять их степень родства.

- (Hint: размышляйте в терминах нормальных форм баз данных)

Вопросы?

- http://t.me/log_pro
- https://soshnikov.com/courses/logpro/