



Язык представления и
обработки знаний Пролог.

Контроль механизма отката в
Прологе.



- В Прологе существуют два специальных предиката, которые позволяют **контролировать механизм бэктрекинга**: предикат ***fail***, заставляющий запускать механизм бэктрекинга, и ***cut*** (обозначается с помощью символа “!”), предназначенный для отмены бэктрекинга.

ИСПОЛЬЗОВАНИЕ ПРЕДИКАТА fail



- Пролог запускает бэктрекинг, когда очередное сопоставление (унификация) **завершается неудачей.**
- Предикат fail порождает значение «ложь» (т.е. неудача) и заставляет начать поиск решений заново.



```
domains
name = symbol

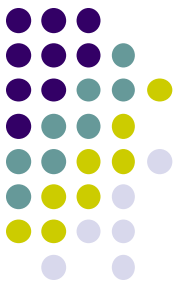
predicates
nondeterm father(name, name)
nondeterm everybody

clauses
father(leonard, katherine).
father(carl, jason).
father(carl, marilyn).
everybody:-father(X,Y), write(X, " is ", Y, "'s father\n").

goal
everybody.
```



```
leonard is katherine's father
yes
```



```
domains
name = symbol

predicates
nondeterm father(name, name)
nondeterm everybody

clauses
father(leonard, katherine).
father(carl, jason).
father(carl, marilyn).
everybody:-father(X,Y), write(X, " is ", Y, "'s father\n"), fail.

goal
everybody.
```



```
leonard is katherine's father
carl is jason's father
carl is marilyn's father
no|
```

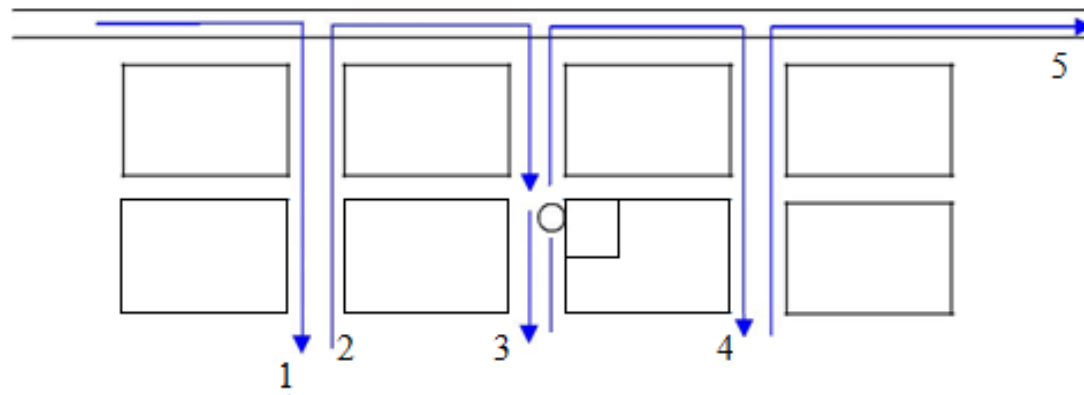
Применение отсечений (отмена бэктрекинга)



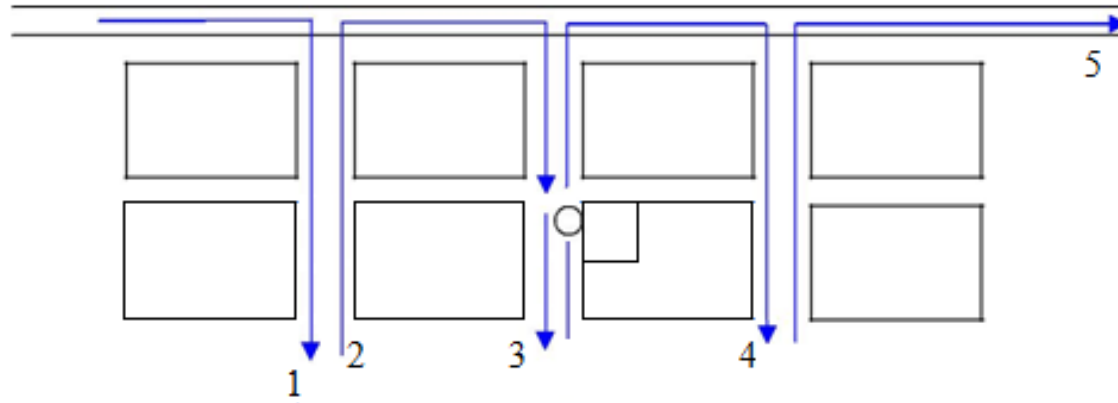
- Пусть необходимо найти дорогу на дачу. При этом инструкция, как туда добраться, выглядит следующим образом:
 1. Въехать в деревню Васино (может быть, Ванино - написано неразборчиво).
 2. Повернуть направо.
 3. Доехать до колодца.
 4. Искомый дом из красного кирпича – первый от колодца по правой стороне.



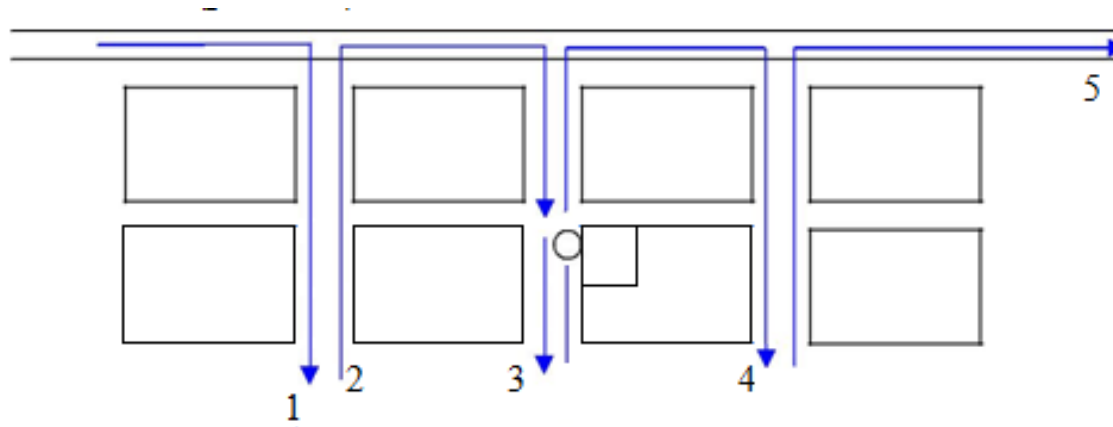
```
dacha1(X) :-enter_village(X), find_a_house.  
find_a_house :- turn_right, meet_mine,  
               see_a_red_brick_house.  
enter_village(vasino). enter_village(vanino).
```



- Въезжаем в деревню Васино и поворачиваем направо (стрелка 1). Проезжаем до конца деревни и не находим колодца. Выполняем откат, то есть возвращаемся на шоссе (стрелка 2) и едем до следующего поворота направо, доезжаем до колодца и видим, что ближайший к нему дом из белого кирпича. Едем до конца улицы и видим, что колодцев больше нет (стрелка 3).



- Откатываемся на шоссе и едем до следующего поворота (стрелка 4), проезжаем всю улицу и откатываемся, поскольку колодцев здесь нет (стрелка 5). Из этого делаем вывод, что название деревни мы прочитали неверно, и дачу следует искать в следующей деревне. В деревне Ванино повторяем все сначала.



- Отсечение позволяет сократить количество пробегаемых ветвей дерева решений. Если хозяин дачи даст важную дополнительную информацию о том, что в каждой деревне есть только один колодец, мы не будем выполнять действия, обозначенные стрелками 4 и 5, а сразу поймем, что попали не в ту деревню.



- **Отсечение** – это предикат, который обозначается восклицательным знаком и входит в правило.
- Отсечение уничтожает указатели отката, установленные в ходе унификации данного правила, т. е. делает предыдущие предикаты данного правила однозначными.



- Если мы хотим указать в программе поиска дачи, что колодец в деревне единственный, мы должны **поставить отсечение** после найденного колодца:
dacha2(X) :-enter_village(X),find_a_house.
find_a_house :- turn_right, meet_mine, !,
see_a_red_brick_house.
enter_village(vasino). enter_village(vanino).



- Как только Пролог достигнет отсечения, весь предикат *find_a_house* **станет однозначным**, следовательно, дальнейшая неудача в предикате *see_a_red_brick_house* приведет к тому, что предикат *find_a_house* также завершится неудачей.



- Так как предикат *enter_village* **является неоднозначным** (как показано в программе), то мы имеем шанс найти дачу в другой деревне, поскольку отсечение в правиле *find_a_house* не затрагивает свойства предиката *dacha* и установленные в нем указатели отката.



dacha3(X) :-enter_village(X), turn_right,
meet_mine, !,see_a_red_brick_house.

enter_village(vasino). enter_village(vanino).

- В этом случае отсечение приведет к тому, что все предыдущие предикаты в правиле *dacha* **станут однозначными**, в том числе и *enter_village*. Иными словами, нам будет запрещено искать дачу в следующей деревне.



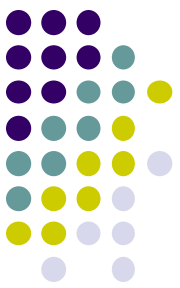
- Если отсечение не влияет на логику работы программы, а только сокращает возможные спуски по ложным ветвям дерева решений, то такое отсечение называется **зеленым**.
- Если отсечение влияет на логику программы (без него программа работает иначе), такое отсечение называется **красным**.



- В приведенном выше правиле *dacha2* отсечение **зеленое**, поскольку не влияет на логику программы.
- В правиле *dacha3* отсечение **меняет логику программы**, т.к. поиск ограничивается первым найденным колодцем на всем пути следования к цели, а не в каждой деревне.



- Использование отсечений позволяет обходить (не просматривать) при возврате некоторые цели.
- В результате чего, во-первых, программа работает быстрее, так как не рассматриваются цели, ничего нового не дающие для решения, во-вторых, программа занимает меньше места в памяти, так как нет необходимости запоминать точки возврата, чтобы обойти ненужные цели.




```
domains
name=symbol

predicates
nondeterm father(name,name) .
nondeterm grand(name,name) .

clauses
father(a,b) .
father(b,c) .
father(c,d) .
grand(X,Y) :- father(X,Y) .
grand(X,Y) :-father(X,Z),grand(Z,Y) .

goal
father(X,Y)|.
```

 [Inactive C:\VIP52\BIN\WI

X=a, Y=b
X=b, Y=c
X=c, Y=d
3 Solutions|

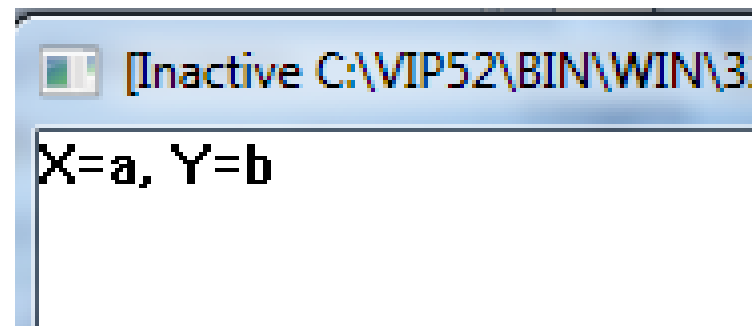


```
domains
name=symbol

predicates
nondeterm father(name,name) .
nondeterm grand(name,name) .

clauses
father(a,b) .
father(b,c) .
father(c,d) .
grand(X,Y) :- father(X,Y) .
grand(X,Y) :-father(X,Z),grand(Z,Y) .

goal
father(X,Y) , !.
```



Использование предиката `not`



- Унарный предикат `not` используется для отрицания. Предикат `not(Цель)` будет истинным в случае, когда `Цель` является ложной.

```
domains
  name = symbol
  gpa = real

predicates
  nondeterm honor_student(name)
  nondeterm student(name, gpa)
  nondeterm probation(name)

clauses
  honor_student(Name):-student(Name, GPA),GPA>=3.5, not(probation(Name)).
  student("Betty Blue", 3.5).
  student("David Smith", 2.0).
  student("John Johnson", 3.7).
  probation("Betty Blue").
  probation("David Smith").

goal
  honor_student(Name).
```