Рекурсия и итерация.

Лектор:
доцент каф. АОИ
Салмина Нина Юрьевна

Простейшие примеры рекурсии

Определение предка (семейные отношения)

```
% Терминальная ветвь
Predok (X,Y) :- parent (X,Y).
% Рекурсивная ветвь
Predok (X,Y) :- parent (X,Z), predok (Z,Y).
```

Простейшие примеры рекурсии

Определение связности вершин в ориентированном графе (при отсутствии циклов!)

```
Edge (a,d).     . . .
Edge (f,b).
Connected(A1,A2) :- edge (A1,A2).
Connected(A1,A2) :- edge (A1,X),
                    connected (X,A2).
```

Описание ребер

Примеры числовых рекурсий

- Вычисление факториала F(N,Y)
- Возведение числа X в степень N
  - N > 0 и N < 0

a) $X^N = X * X^{N-1}$

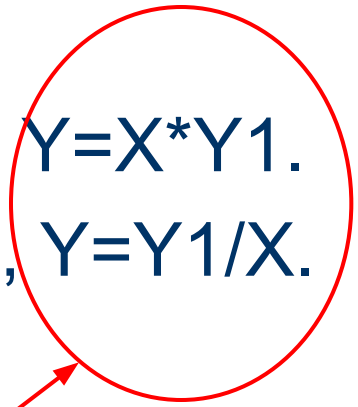
b) $X^N = Y * Y$, если N – четное, $Y=X^{N/2}$;

$X^N = X * X^{N-1}$, если N – нечетное.

(отсечение / условия)

Отложенные вычисления.

```
f(_,0,1).
f(X,N,Y) :- N>0, N1=N-1, f(X,N1,Y1), Y=X*Y1.
f(X,N,Y) :- N<0, N1=N+1, f(X,N1,Y1), Y=Y1/X.
```



Вычисления откладываются до достижения цели (выполнения граничного условия)

Итерации

Рекурсия

Отложенные вычисления.

Объем памяти линейно зависит от числа выполняемых рекурсивных обращений

Итерация

Отсутствуют отложенные вычисления.

Нет необходимости в использовании дополнительной памяти стека (не зависит от числа обращений)

Отсечение

Механизм, позволяющий отбросить ненужные решения.

Встроенный предикат «!»

Пример использования отсечения (быстрое возведение числа в степень)

```
F1 (_,0,1) :- !.
F1 (X,N,Y) :- 0=N mod 2, !, N1=N/2,
              F1 (X, N1, Y1), Y=Y1*Y1.
F1 (X,N,Y) :- N1=N-1,
              F1 (X,N1,Y1), Y=Y1*X.
```

Общее правило применения отсечения

A :- … .

C: A :- B1,…,Bk,!,Bk+1,…Bn.

A :- … .

Если дошли до отсечения, то:

1) Фиксируется предложение C.
2) Все другие предложения процедуры A игнорируются.
3) Если некоторое Bi i>k не выполняется, то возврат – не далее отсечения.
4) Если предложение C не выполнимо (не дошли до отсечения), переход к следующему предложению процедуры A.

Определение максимума. Отсечение. Дополнительные условия.

```
Max (X, Y, X) :- X>=Y.
Max (X, Y, Y) :- X<Y.
```

```
Max (X, Y, X) :- X>=Y, !.
Max (X, Y, Y) .:- X<Y.
```

.

Определение максимума. Отсечение. Дополнительные условия.

Max (X, Y, X) :- X>=Y.

Max (X, Y, Y) :- X<Y.

Max (X, Y, X) :- X>=Y, !.

Max (X, Y, Y) .:- X<Y.

? – max (5, 3, 3).

Сложные термы

- Структуры
- Перечислимые термы
- Списки

Структуры

S(X1, X2, …, XN)

S – имя структуры (функтор).

Xi – компоненты (аргументы) структуры: константы; переменные; структуры.

N – число аргументов (арность).

дата(1, май, 1983)

дата(День, Месяц, Год)

точка(4,2)

треугольник(точка(4,2),точка(6,4),точка(7,1))

Описание структуры:

T = S(X1, X2, …, XN)

T – имя типа (может совпадать с именем структуры)

Сложные термы (описание)

Перечислимые термы

T = k1; k2; … kN

T – имя типа

ki – одно из возможных значений (атом, функтор)

Списки

T = <тип элементов списка>*

Примеры:

direction = north; south; west; east

sp = integer*

list = direction*

list2 = sp*

Пример: обезьяна и банан

Возле двери комнаты стоит обезьяна. В середине этой комнаты к потолку подвешен банан. Обезьяна хочет съесть банан, но не может дотянуться до него, стоя на полу. Около окна этой же комнаты на полу лежит ящик, которым обезьяна может воспользоваться.

Обезьяна может предпринимать следующие действия: ходить по полу, залезать на ящик, двигать ящик (если она уже находится возле него) и схватить банан, если она стоит на ящике прямо под бананом.

Может ли обезьяна добраться до банана?

Определение состояний «обезьяньего мира»

Исходное состояние:

- Обезьяна у двери
- Обезьяна на полу
- Ящик у окна
- Обезьяна не имеет банана

Состояние (<расположение обезьяны в комнате>,
<обезьяна на полу/ящике>,
<расположение ящика в комнате>,
<обладание бананом>).

Domains

```
where_v = down; up
where_g = door; window; middle
banana = have; no
state = state (where_g monkey,
               where_v,
                   where_g box,
                   banana)
```

Возможные ходы обезьяны

Четыре типа ходов:

- Схватить банан
- Залезть на ящик
- Подвинуть ящик
- Перейти в другое место

Ход (<Состояние1>, <Состояние2>).

```
Может_завладеть (Состояние(_,_,_,имеет)).
Может_завладеть (Состояние1) :-
        ход (Состояние1, Состояние2),
        может_завладеть (Состояние2).
```

Пример: обезьяна и банан Программа

```
Step (state (middle, up, middle, no),
        state (middle, up, middle, have)).  % take banana
Step (state (X, down, X, S),
        state (X, up, X, S)).          % up on the box
Step (state (X, down, X, S),
        state (Y, down, Y, S)).            % move with box
Step (state (X, down, Z, S),
        state (Y, down, Z, S)).            % move

may_have (state (_, _, _, have)).
may_have (P1) :- step (P1,P2), may_have (P2).

Goal
may_have (state (door, down, window, no)).
```

Обезьяна и банан. Вывод промежуточных состояний

```
may_have (P1) :- step(P1,P2), write(P2), nl,
                 may_have(P2).
```

```
State ( _, down, window, no)
State (window, up, window, no)
State ( _, down, _, no)
State ( _, up, _, no)
State (middle, up, middle, have)
yes
```

«Семейные отношения», структуры данных

```
domains
pol = man; woman
date = day(integer day, integer month, real year)
ass = string*

predicates
Person (string fam, symbol name, pol, date birthday)
Child (string fam, ass)
Family (string fam, string name_husband, string name_wife,
                                    ass childrens)
age_husband (string fam, date data_today, integer age)
Age (date date_today, date birthday, integer age)
```

«Семейные отношения», программа

```
clauses
Family (ivanov, ivan, lena, [olga,peter]).
Family (sidorov, david, olga, [vera, igor]).
child(X,Y) :- family(X,_,_,Y).
Person (ivanov, "ivan", man, day(5,1,1960)).
Person (ivanov, lena, woman, day(21,4,1962)).
Person (ivanov, olga, woman, day(13,12,1990)).
Age (day(D,M,G), day(A,B,C), Y)  :-  B>M, Y=G-C-1.
Age (day(D,M,G), day(A,B,C), Y)  :-  B<M, Y=G-C.
Age (day(D,M,G), day(A,B,C), Y)  :-  B=M, D>A, Y=G-C-1.
Age (day(D,M,G), day(A,B,C), Y)  :- B=M, D<=A, Y=G-C.
age_husband (X,C,Y) :- family(X,Z,_,_), person(X,Z,_,A), age(C,A,Y).
goal
age_husband (ivanov, day(1,9,2020), Y).
```

«Семейные отношения», без предупреждений и повторов

```
clauses
Family (ivanov, ivan, lena, [olga,peter]).
Family (sidorov, david, olga, [vera, igor]).
child(X,Y) :- family(X,_,_,Y).
Person (ivanov, "ivan", man, day(5,1,1960)).
Person (ivanov, lena, woman, day(21,4,1962)).
Person (ivanov, olga, woman, day(13,12,1990)).
Age (day(_,_,G), day(_,_,C),_) :- G<C, fail, !.
Age (day(_,M,G), day(_,B,C), Y)  :-  B>M, !, Y=G-C-1.
Age (day(_,M,G), day(_,B,C), Y)  :-  B<M, !, Y=G-C.
Age (day(D,_,G), day(A,_,C), Y)  :-  D>A, !, Y=G-C-1.
Age (day(_,_,G), day(_,_,C), Y)  :- Y=G-C.
age_husband (X,C,Y) :- family(X,Z,_,_), person(X,Z,_,A), age(C,A,Y).
goal
age_husband (ivanov, day(1,9,2017), Y).
```

Задача классификации объектов

В базе данных содержатся результаты теннисных партий, сыгранных членами некоторого клуба:

Победил (Победитель, Проигравший).

Необходимо определить отношение

Класс (Игрок, Категория)

где победитель – игрок, победивший во всех сыгранных им играх;

боец – игрок, в некоторых играх победивший, в некоторых – проигравший;

спортсмен – игрок, проигравший во всех сыгранных им играх.

Задача классификации объектов. Формулировки правил по категориям.

X – боец, если

существует Y такой, что X победил Y, И

существует Z такой, что Z победил X.

X – победитель, если

существует Y такой, что X победил Y, И

НЕ существует Z такой, что Z победил X.

X – спортсмен, если

существует Y такой, что Y победил X, И

НЕ существует Z такой, что X победил Z.

Задача классификации объектов. Программа на Прологе (НЕ)

```
Won (tom, ivan).
Won (peter, tom).
Won (peter, ivan).
Class(X, fighter) :- won (X,_), won (_,X).
Class(X, winner) :- won (X,_), not(won(_,X)).
Class(X, athlete) :- won (_,X), not(won(X,_)).
```

Задача классификации объектов. Формулировки правил по категориям. Замена НЕ на ИНАЧЕ

Если X победил кого-либо и X был кем-то побежден,

то X – боец,

иначе, если X победил кого-либо,

то X – победитель,

иначе, если X был кем-то побежден,

то X – спортсмен.

Задача классификации объектов. Программа на Прологе (ИНАЧЕ)

```
Won (tom, ivan).
Won (peter, tom).
Won (peter, ivan).
Class (X, fighter) :- won (X,_), won (_,X), !.
Class (X, winner) :- won (X,_), !.
Class (X, athlete) :- won (_,X).
```

Отсечение.
Изменение порядка предложений – декларативный смысл.

```
p :- a, b.
p :- c.
```

Декларативный смысл:

p <==> (a & b) U c

```
p :- a, !, b.
p :- c.
```

Декларативный смысл:

p <==> (a & b) U (~a & c)

```
p :- c.
p :- a, !, b.
```

Декларативный смысл:

p <==> c U (a & b)