

# **Изкуствен интелект / интелигентни системи**

*гл. асистент д-р Венета Табакова-Комсалова  
ФМИ, ПУ „П. Хилендарски“, Пловдив*

## Рекурсия

Пролог поддържа дескрептивен стил на програмиране, при който проблемите се задават като спецификация на обекти и релации между тях. При един такъв подход особено значение придобива способността на програмистите да конструират достатъчно общовалидни дефиниции. Едно мощно средство за създаване на общовалидни описания на обектите и релациите е рекурсията.

Силата на рекурсията е в създаване на кратки програми.

Недостатък на рекурсията е, че тя не е естествен начин за мислене и разсъждения при хората, както и факта, че изисква повече изчислителни ресурси.

# Същност на рекурсивните дефиниции

Най-общо, една рекурсивна дефиниция включва в лявата страна понятието, което искаме да дефинираме (дадено в дясната страна). Смисълът на един рекурсивен проблем е в това, дали можем циклично да го опростяваме и да прилагаме същия начин за търсене на решение както за оригиналния.

Освен това е необходимо да специфицираме „дъното“ на рекурсията, т.е. проблемът е решим с „единична“ (позната) операция.

Една рекурсивна дефиниция в Пролог, обикновено се задава с две клаузи - рекурсивна клауза и клауза за единичната операция (стоп клауза).

Рекурсивните дефиниции могат да бъдат:

Ляворекурсивни

Дяснорекурсивни.

## Пример за рекурсивен проблем: „Ханойски кули“

Ханойските кули (от името на град Ханой) е математическа игра, измислена от френския математик Едуард Люка през 1883 година.

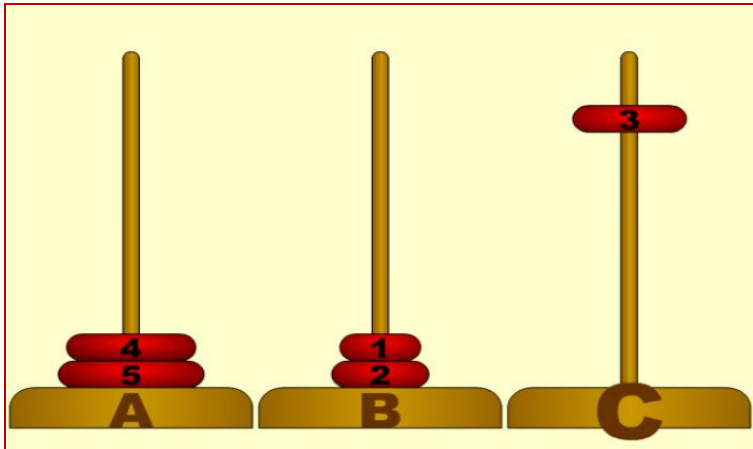
Играта се играе с няколко диска, различни по размер един от друг, и три стълба. В началото дисковете са подредени по големина на левия стълб, като най-големият е най-отдолу, а най-малкият е най-отгоре. Целта е кулата да бъде преместена от левия на десния стълб като се използва един помощен стълб по средата като се спазват следните правила:

- може да се мести само по един диск на ход и
- не може по-голям диск да бъде поставен върху по-малък.

Всеки ход е съставен от взимането на горния диск от даден стълб и в поставянето му най-отгоре на друг стълб.

Любопитно. Съществува вариант на играта със 64 диска, наречен Кулата на Брахма. Легендата разказва, че когато монасите от Брахма приключат играта ще настъпи краят на света.

## Пример за рекурсивен проблем: „Ханойски кули“



% „Дъно“ на рекурсията

```
move(1, X, Y, _) :-  
  write('Премести пул от'),  
  write(X),  
  write(' върху '),  
  write(Y),  
  nl.
```

% Рекурсивна процедура

```
move(N, X, Y, Z) :-  
  N > 1,  
  M is N-1,  
  move(M, X, Z, Y),  
  move(1, X, Y, _),  
  move(M, Z, Y, X).
```

Нека зададем следния въпрос към БД:  
**?- move(3, a, c, b).**

Премести пул от а върху с

Премести пул от а върху b

Премести пул от с върху b

Премести пул от а върху с

Премести пул от b върху а

Премести пул от b върху с

Премести пул от а върху с

**true**

Съществуват различни разновидности на рекурсията.

Различаваме:

**Рекурсивни програми** - в програмирането с **рекурсия** се обозначава случай, когато една подпрограма вика предходна своя функция. Рекурсията условно се разделя в две категории: директна (пряка) и индиректна (косвена). Рекурсията е пряка, когато в тялото на подпрограмата има референция към нея. Косвена е тази рекурсия, при която една подпрограма вика друга, а тя вика предходната. Съществуват и случаи на косвена рекурсия, при които подпрограмата извиква себе си, след поредица от обръщения към други подпрограми.

**Рекурсивни структури** – типичен пример за рекурсивни структури са списъците

**Рекурсивни проблеми** – проблемът, който искаме да решаваме, естествено, се представя рекурсивно.

**Рекурсивен предикат** – в дясната част на някое негово правило се среща същият предикат.

$p(X):-a(X,Y), p(Y).$

Използване:

Задачата се формулира рекурсивно (отношението се описва чрез себе си);

Обработка се рекурсивна структура от данни.

Рекурсията – единствен начин за реализиране на итерация (**цикли**).

Рекурсивната дефиниция включва:

**Стоп-клауза** (една или повече) за завършване на рекурсията

**Рекурсивна клауза** (една или повече)

Пример:

Дефиниране на потомък (наследник) чрез предикатите син и дъщеря.

потомък(X,иван):-син(X,иван). % стоп клауза

потомък(X,иван):-дъщеря(X,иван). % стоп клауза

потомък(X,иван):-син(Y,иван), потомък(X,Y). % рекурсивна клауза

потомък(X,иван):- потомък(Y,иван), потомък(X,Y). % рекурсивна  
клауза



ПОТОМСТВО(X,Y):- родител(X,Y).

ПОТОМСТВО(X,Y):- родител (X,Z),ПОТОМСТВО(Z,Y).

ПОТОМСТВО(X,Y):- родител (X,Z), ПОТОМСТВО(Z,Y).

ПОТОМСТВО(X,Y):- родител(X,Y).

ПОТОМСТВО(X,Y):- родител(X,Y).

ПОТОМСТВО(X,Y):- ПОТОМСТВО(X,Z),родител(Z,Y).

ПОТОМСТВО(X,Y):- ПОТОМСТВО(X,Z), родител(Z,Y).

ПОТОМСТВО(X,Y):- родител (X,Y).

**Задача 1.** В програма на Пролог са дефинирани предикатите **person**, **female**, **likes**, **loves** и **has** (човек, приятел, харесва, обича и има), (виж по-доли в коментарите на програмата кой предикат какво отношение представя).

1. Дефинирайте предикати, описващи следните твърдения/отношения

А) Щастлив е този, който има компютър и някой го обича.

Б) Мария би харесала мъж, който я обича и има кола.

В) Предикат, който определя дали двама души харесват тенис.

Г) Предикат, който определя кой човек не харесва тенис.

Д) Предикат, който определя кой човек не харесва нищо.

Е) Предикат, който определя кога (дали) нещо се харесва поне от двама души.

Ж) Мъж и жена могат да образуват двойка, ако се обичат и има поне едно нещо, което и двамата харесват.

З) Мъж и жена могат да образуват двойка, ако се обичат и харесват еднакви неща. (всичко, което се харесва от единия се харесва и от другия).

2. Напишете вътрешни цели, които извеждат всички отговори на Д) и Ж).

# ФАКТИ

person(ivan).  
person(petar).  
person(maria).  
person(anna).  
person(elena).  
person(dimitar).  
person(tanya).  
person(veselin).

% male(X)  
male(ivan).  
male(petar).  
male(dimitar).

% female(X)  
female(X):-person(X), \+ male(X).

% likes(X,Y)  
likes(ivan,football).  
likes(ivan,computer\_games).  
likes(ivan,books).  
likes(maria,tennis).  
likes(maria,flowers).  
likes(anna,football).  
likes(anna,books).  
likes(anna,computer\_games).  
likes(petar,tennis).  
likes(tanya,tennis).  
likes(maria,football).  
likes(maria,books).  
  
likes(dimitar,X):-likes(petar,X).

# Факти

% loves(X,Y)

loves(ivan,maria).

loves(ivan,anna).

loves(ivan,elena).

loves(petar,maria).

loves(elena,ivan).

loves(maria,petar).

loves(tanya,X):-male(X).

loves(dimitar,tanya).

loves(maria,ivan).

has(ivan,computer).

has(petar,car).

has(petar,computer).

А) Щастлив е този, който има компютър и някой го обича.

Б) Мария би харесала мъж, който я обича и има кола.

В) Предикат, който определя дали двама души харесват тенис.

Г) Предикат, който определя кой човек не харесва тенис.

Д) Предикат, който определя кой човек не харесва нищо.

Е) Предикат, който определя кога (дали) нещо се харесва поне от двама души.

Ж) Мъж и жена могат да образуват двойка, ако се обичат и има поне едно нещо, което и двамата харесват.

З) Мъж и жена могат да образуват двойка, ако се обичат и харесват еднакви неща. (всичко, което се харесва от единия се харесва и от другия).

А) Щастлив е този, който има компютър и някой го обича.

`happy(X):- has(X,computer), loves(_,X).`

Б) Мария би харесала мъж, който я обича и има кола.

`could_be_like(maria,X):- male(X), loves(X, maria),  
has(X,car).`

В) Предикат, който определя дали двама души харесват тенис.

`like2tennis(X,Y):-likes(X,tennis), likes(Y, tennis), X\=Y.`

Г) Предикат, който определя кой човек не харесва тенис.

`don't_like_tennis(Z):-person(Z), not like(Z,tennis).`

Д) Предикат, който определя кой човек не харесва нищо.

`likes_nothing(X):-person(X), \+likes(X,_).`

Е) Предикат, който определя кога (дали) нещо се харесва поне от двама души.

`likes2(X):-likes(A,X), likes(B,X), A\=B.`

Ж) Мъж и жена могат да образуват двойка, ако се обичат и има поне едно нещо, което и двамата харесват.

`like2exactly(X):-likes(A,X), likes(B,X), A\=B,  
 \+(likes(Z,X),Z\=A, Z\=B).`

З) Мъж и жена могат да образуват двойка, ако се обичат и харесват еднакви неща. (всичко, което се харесва от единия се харесва и от другия).

`couple(X,Y):-male(X), female(Y), loves(X,Y),  
 \+(likes(X,S), not likes(Y,S); likes(Y,S), not likes(X,S)).`


2. Напишете вътрешни цели, които извеждат всички отговори на Д) и Ж).

?- write(`Кой не харесва нищо:`), nl, likes\_nothing(X), write(X), nl, fail.


?- write(`Възможни двойки:`), nl, pair(X,Y), pair(X,Y), write(X), write(`--`), write(Y), nl, fail.

# Пресмятане на аритметични изрази


?- X is  
5+sqrt(2) .

 X is 5+sqrt(2).  
X = 6.414213562373095  
?- X is 5+sqrt(2).|

?- Y=5, X is Y+1.


 Y=5, X is Y+1.  
X = 6,  
Y = 5

?- 5+6 == 3+8.

 5+6 == 3+8.

true

?- 5+6 \= 4+6.

 5+6 \= 4+6.

true



# Задача

1. Дефинирайте предикат за пресмятане по зададено  $X$  на стойността на функцията:

$$g(x) = \begin{cases} \ln(x+3), & 0,5 < x \leq 5 \\ -3, & -0,5 \leq x \leq 0,5 \\ \sqrt{|12-x|} + x^2, & x < -0,5 \text{ или } x > 5 \end{cases}$$

## 2. Пресмятане на факториел

А) Рекурсивен вариант – съответства на написване на рекурсивна програма на процедурен език. Това е естественият начин за дефиниране на отношения.

% 0!=1 (стоп-клауза) `fact(0,1).`

%n!=n\*(n-1)! (рекурсивна-клауза) `fact(N,F):- N>0, N1 is N-1, fact(N1,F1), F is F1*N.`

Б) Интерактивен вариант – съответства на написване на цикъл процедурен език. Предикатът е пак рекурсивен, но рекурсивното извикване е последно в последната клауза (дясна рекурсия), което позволява по-голяма ефективност.

`fact(N,F):- f(0,N,1,F).`

`f(N,N,F,F).`

`f(X,N,Y,F):-X<N, X1 is X+1, Y1 is Y*X1, f(X1,N,Y1,F).`

`x=0; f=1;`  
`While (x<n)`  
`{x++; f=f*x}`

Внимание! Ако един предикат се извика с аргумент свързана променлива, не може тази променлива да приеме друга стойност в процеса на търсене на решение.

Не може да присвоявате стойност така `X is X+1`, защото от двете страни на `is` трябва да има еднакви стойности, а никое число не е равно на себе си плюс 1. Т.е. щом променливата е свързана с една стойност, тя не може да се свърже чрез присвояване `is` с друга стойност.

3. Пресмятане на степен:  $x^n$   $n > 0$

4. Отпечатване на екрана на числата от 1 до зададено число.

5. Пресмятане на сумата  $(x+2)^2 + (x+4)^2 + (x+6)^2 + \dots + (x+a)^2$   
където  $a$  е най-голямото четно число,  
ненадхвърлящо  $n$

6. Пресмятане на НОД на две числа

7. Пресмятане на  $n$ -ти елемент на редицата на  
Фибоначи

$$f_0 = 1; f_1 = 1; f_n = f_{n-1} + f_{n-2}, n > 1.$$

### Задачи за упражнение:

1. Дефинирайте предикати за намиране на стойността на функциите:

$$f(x, y) = \begin{cases} \sqrt{|y^2 - 3x^2|}, & y > x \\ x \sin x, & x = y, \\ e^x + \ln(x - y), & x > y \end{cases} \quad h(y) = \begin{cases} \frac{2y^2}{y+4}, & y < -4 \\ 0, & y \in [-4, 4] \\ \frac{\ln(y-4)e^y}{y^3 + \sqrt{y}}, & y > 4 \end{cases} \quad \text{и} \quad k(n) = \frac{n!}{(n+2)^2}.$$

2. Дефинирайте предикат за пресмятане на сумата:  $1 + \frac{1}{2^2} + \frac{1}{3^2} + \dots + \frac{1}{n^2}$  по дадено  $n$ .

3. Дефинирайте предикат за пресмятане на сумата:  $1 + \frac{1}{2}x^2 + \frac{1.3}{2.4}x^4 + \frac{1.3.5}{2.4.6}x^6 + \dots$  за  $|x| < 1$  със зададена точност – много малко  $\varepsilon$  (абсолютна грешка).

4. Дефинирайте предикат за пресмятане на произведението  $n.(n-k).(n-2k)\dots(n-k^2)$  при дадени  $n$  и  $k$ .

5. Дефинирайте предикат за пресмятане на произведението  $\frac{1}{2^2-1} \cdot \frac{1}{3^2-2} \dots \frac{1}{n^2-n+1}$  по дадено  $n$ .

6. Дефинирайте предикат, който намира  $n$ -тия член на рекурентната редица  $T_0(x)=1$ ,  $T_1(x)=x$ ,  $T_k(x)=2.x.T_{k-1}(x)-T_{k-2}(x)$ ,  $k=2, 3, \dots$  при зададени  $x$  и  $n$ .

7. Дадена е редицата  $a_n = \frac{n!}{5^n}$ ,  $n=1, 2, \dots$ . Дефинирайте предикат, който намира първото  $a_n$ , което е по-голямо от дадено  $x$  (т.е.  $a_n > x$  и  $n$  - минимално).