Podstawy Prologu

dr M.Kołowska-Gawiejnowicz

Wydział Matematyki i Informatyki
Uniwersytet im. Adama Mickiewicza w Poznaniu

e-mail: mkolowsk@amu.edu.pl

Struktury danych - listy

- **Lista** – ciąg elementów o dowolnej długości.
- Elementy listy mogą być dowolnymi termami: stałymi, zmiennymi, strukturami (w tym listami).
- Lista jest albo listą pustą, nie zawierającą żadnych elementów, albo jest strukturą z dwiema składowymi: **głową** i **ogonem**.

Listy

- Lista jest strukturą rekurencyjną (do jej konstrukcji użyto funktora . (kropka)
- Listę pustą zapisuje się: []
- Głowa i ogon listy są argumentami funktora . (kropka)

Przykłady:
.(a,[]) lista jednoelementowa
.(a,.(b,[])) lista o elementach a, b
.(a,.(b,.(c,[]))) lista o elementach a, b, c
.(a,.(b,.(c,.(d,[])))) lista o elementach a, b, c, d

Przykłady list

Wygodniejszy zapis listy: elementy oddziela się przecinkami i umieszcza między nawiasami [oraz]

Zamiast np.: .(1,.(2,.(3,[])))
pisze się: [1,2,3]

Przykłady list:

- [wydział, informatyki]
- [X,lubi,Y]
- [autor(adam,mickiewicz),'Pan Tadeusz']
- [[2,3],[5,6,7],[2,8]]
- [a,1,[2,3],b]
- [prolog, haskell, scheme, c++]

Lista z głową X i ogonem Y [X|Y]

lista	głowa	ogon
[ ]	niezdefiniowane	niezdefiniowane
[a]	a	[ ]
[a,b]	a	[b]
[a|[b]]	a	[b]
[a,b,c]	a	[b,c]
[a|[b,c]]	a	[b,c]
[a,b|[c]]	a	[b,c]
[a,b,c|[ ]]	a	[b,c]
[[1,2],[3,4],5,6]	[1,2]	[[3,4],5,6]
[[1,2],[a,b]]	[1,2]	[[a,b]]
[[1,2]|[a,b]]	[1,2]	[a,b]

Unifikacja list - przykłady

1 ?- .(a,.(b,.(c,[]))) = [a,b,c].

true.

2 ?- .(a,.(B,.(C,[]))) = [a,b,c].

B = b,
C = c.

3 ?- [a,V,1,[c,s],p(X)] = [A,B,C,D,E].

V = B,
A = a,
C = 1,
D = [c, s],
E = p(X).

Unifikacja list - przykłady

5 ?- [1,2,3,4] = [1|[2,3,4]].
true.

6 ?- [1,2,3,4] = [1,2|[3,4]].
true.

7 ?- [1,2,3,4] = [1,2,3|[4]].
true.

8 ?- [1,2,3,4] = [1,2,3,4|[]].
true.

9 ?- [1,2,3,4] = [1,2,3,4,[]].
false.

10 ?- [1,2,3,4] = [1|2,3,4].
ERROR: Syntax error: Unexpected comma or bar in rest of list
ERROR: [1,2,3,4] = [1|
ERROR: ** here **
ERROR: 2,3,4] .

Unifikacja list - przykłady

```
10 ?- [H|T]=[1,2,3,4].
H = 1,
T = [2, 3, 4].

11 ?- [A|B]=[[1,2],3,4].
A = [1, 2],
B = [3, 4].

12 ?- [H|T]=[A,b].
H = A,
T = [b].

13 ?- [H|T]=[b].
H = b,
T = [].

14 ?- [A,B|C]=[a|[1,2,3]].
A = a,
B = 1,
C = [2, 3].
```

Przetwarzanie list

- Listy są strukturami rekurencyjnymi, do ich przetwarzania służą procedury rekurencyjne.
- Procedura – zbiór klauzul zbudowany w oparciu o ten sam predykat.
- Procedura rekurencyjna składa się z klauzul:
  1. Faktu opisującego sytuację, która powoduje zakończenie rekurencji, np. napotkanie listy pustej.
  2. Reguły, która przedstawia sposób przetwarzania listy. W jej ciele znajduje się ten sam predykat, co w nagłówku, tylko z innymi argumentami.

Przykład procedury rekurencyjnej

Wypisanie na ekranie elementów listy

```
pisz([ ]).
pisz([X|Y]):-write(X), nl, pisz(Y).
```

Fakt mówi, że w przypadku napotkania listy pustej (końca listy) nie należy nic robić.
Reguła mówi: podziel listę na głowę i ogon, wydrukuj głowę listy, następnie ją pomiń i zastosuj tę samą metodę do powstałego ogona.

write ozn. wypisanie termu
nl ozn. przejście do nowej linii

Przykłady predykatów wbudowanych działających na listach

is_list (L)- sprawdza, czy L jest listą

Przykład

?- is_list([1,2,a,b]).

true.

append (L1,L2,L3) – łączy listy L1 i L2 w listę L3

Przykład

?- append([1,2],[3,4],X).

X=[1,2,3,4].

?- append([1,2],X,[1,2,3,4]).

X=[3,4].

?- append([X,[1,2],[1,2,3,4]).

false.

Przykłady predykatów wbudowanych działających na listach

member(E,L) – sprawdza, czy element E należy do listy L lub wypisuje elementy listy L

Przykład

```
?- member(5,[3,6,5,7,6]).
    true.
?- member(X,[2,3,4,9]).
    X = 2 ;
    X = 3 ;
    X = 4 ;
    X = 9 ;
    false.
```

memberchk(E,L) - równoważny predykatowi member, ale podaje tylko jedno rozwiązanie (pierwsze)

Przykłady predykatów wbudowanych działających na listach

nextto(X,Y,L) – predykat spełniony, gdy Y występuje bezpośrednio po X w liście L

Przykład

```
?- nextto(X,Y,[2,3,4,5]).
    X = 2,
    Y = 3 ;
    X = 3,
    Y = 4 ;
    X = 4,
    Y = 5 .
?- nextto(3,Y,[2,3,4,5]).
    Y = 4.
?- nextto(X,4,[2,3,4,5]).
    X = 3.
```

Przykłady predykatów wbudowanych działających na listach

delete(L1,E,L2) – z listy L1 usuwa wszystkie wystąpienia elementu E, wynik uzgadnia z listą L2

Przykład

```
?-delete([1,2,3,2,5,3],3,X).
    X = [1, 2, 2, 5].
```

select(E,L,R) – lista R jest uzgadniana z listą, która powstaje z L po usunięciu wybranego (jednego) elementu.

Przykład

```
?-select(3,[1,2,3,2,5,3],X).
    X = [1, 2, 2, 5, 3] ;
    X = [1, 2, 3, 2, 5] ;
    false.
```

Przykłady predykatów wbudowanych działających na listach

nth1(N,L,E) – predykat spełniony, jeśli element listy L o numerze N daje się uzgodnić z elementem E

Przykład

```
?-nth1(2,[a,b,c,d],Y).
    Y = b.
?-nth1(X,[a,d,b,c,d],d).
    X = 2 ;
    X = 5.
```

last(L,E) – ostatni element listy L

Przykład

```
?-last([a,b,c,d],Y).
    Y = d.
```

Przykłady predykatów wbudowanych działających na listach

reverse(L1,L2) – odwraca porządek elementów listy L1 i unifikuje rezultat z listą L2

Przykład

?-reverse([a,b,c,d],Y).

Y= [d,c,b,a].

permutation(L1,L2) – lista L1 jest permutacją listy L2

Przykład

?- permutation([1,2,3],L).

L = [1, 2, 3] ;

L = [2, 1, 3] ;

L = [2, 3, 1] ;

L = [1, 3, 2] ;

L = [3, 1, 2] ;

L = [3, 2, 1] ;

false.

Przykłady predykatów wbudowanych działających na listach

sumlist(L,S) – suma listy liczbowej L

Przykład

?-sumlist([1,4,7,9],S).

S=21.

length(L,N) – liczba elementów listy L

Przykład

?-length([b,2,a,0],N).

N=4.

?-length([[a,b,c],[1,2],prolog],M).

M=3.

Operacje na listach

Sprawdzenie, **czy element jest na liście**
Predykat wbudowany: **member**

Procedura: X jest elementem listy L, jeżeli X jest głową listy L lub X jest elementem ogona listy L.

```
element(X,[X|_]).
element(X,[_|Ogon]) :- element(X,Ogon).
```

Przykład

?-element(a,[p,r,o,l,o,g]).
    false.
?-element(p,[p,r,o,l,o,g]).
    true.

Operacje na listach

Łączenie list

Predykat wbudowany: **append**

Procedura:

- Jeżeli pierwszy element listy jest pusty [], to drugi i trzeci element muszą być takie same (L = L).
- Jeżeli pierwszy element nie jest pusty, to głową listy L3 staje się głowa listy L1, a ogonem listy L3 jest ogon listy L1 złączony z listą L2.

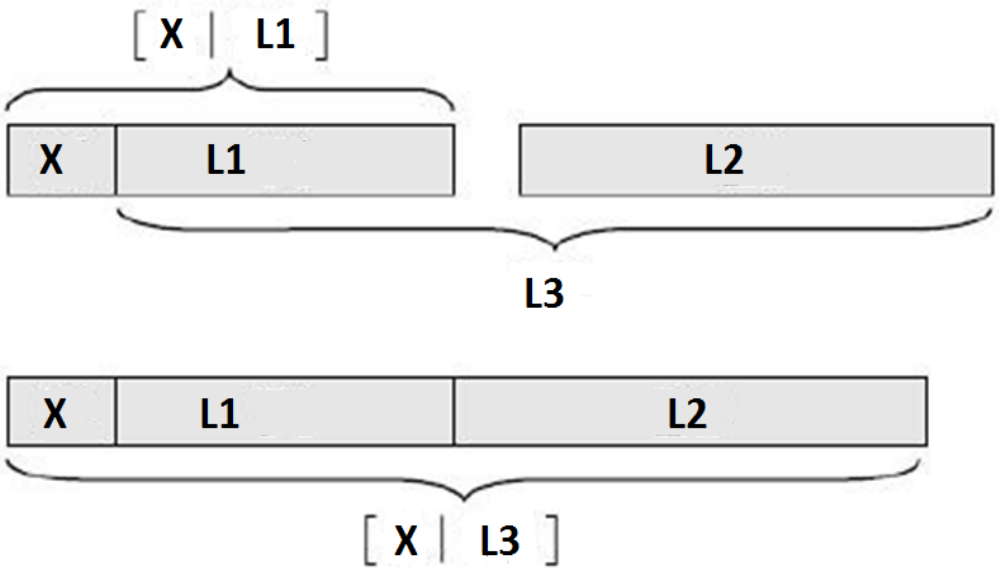
```
polacz([ ],L,L).
polacz([X|L1],L2,[X|L3]):- polacz(L1,L2,L3).
```

Operacje na listach

Przykład

?-polacz([1,2,3],[2,3,4],L).

L=[1,2,3,2,3,4].



polacz([X|L1],L2,[X|L3]):- polacz(L1,L2,L3).

Operacje na listach

Liczba elementów listy liczbowej
Predykat wbudowany: **length**

Procedura:

- Długość listy pustej jest równa 0 (fakt).
- Długość listy, to długość jej ogona plus jeden (reguła).

```
dlugosc([],0).
dlugosc([_|O],N):- dlugosc(O,N1),N is N1+1.
```

Przykład

?-dlugosc([p,r,o,l,o,g],K).
K=6.

Operacje na listach

Odwracanie kolejności elementów listy
Predykat wbudowany: **reverse**

Procedura:

- Odwrotna do listy pustej jest lista pusta (fakt).
- Odwrotnością listy jest połączenie odwróconego ogona listy z listą złożoną z głowy listy wejściowej (reguła).

```
odwracanie([],[]).
odwracanie([A|B],C):-odwracanie(B,D), append(D,[A],C).
```

Przykład
?-odwracanie([g,o,l,o,r,p],X).
X=[p,r,o,l,o,g].

Operacje na listach

n-ty element listy, element na n-tym miejscu w liście

Predykat wbudowany: **nth1**

Procedura:

- Głowa listy jest pierwszym elementem listy (fakt).
- n-ty element listy jest n-1-szym elementem ogona (reguła).

```
nty(1,[E|_],E).
nty(N,[_|X],E):- nty(N1,X,E), N is N1+1.
```

Przykłady

```
3 ?- nth1(3,[a,b,c,d,c],X).
X = c.

4 ?- nth1(Y,[a,b,c,d,c,e],c).
Y = 3 ;
Y = 5 ;
false.
```

```
1 ?- nth1d(3,[a,b,c,d,c],X).
X = c ;
false.

2 ?- nth1d(Y,[a,b,c,d,c],c).
Y = 3 ;
Y = 5 ;
false.
```

Operacje na listach

Porównaj:

```
nth1a(1,[E|_],E).
nth1a(N,[_|T],E):-N is N-1,nth1a(N,T,E).

nth1b(1,[E|_],E).
nth1b(N,[_|T],E):-N1 is N-1, nth1b(N1,T,E).

nth1c(1,[E|_],E).
nth1c(N,[_|T],E):-N is N1+1,nth1c(N1,T,E).

nth1d(1,[E|_],E).
nth1d(N,[_|T],E):- nth1d(N1,T,E),N is N1+1.

nth1e(1,[E|_],E).
nth1e(N,[_|T],E):-nth1e(N1,T,E),N1 is N-1.
```

Operacje na listach

```
nth1a(1,[E|_],E).
nth1a(N,[_|T],E):-N is N-1,nth1a(N,T,E).
```

```
1 ?- nth1a(3,[a,b,c,d,c,e],X).
false.

2 ?- nth1a(Y,[a,b,c,d,c,e],c).
ERROR: is/2: Arguments are not sufficiently instantiated
```

Operacje na listach

```
nth1b(1,[E|_],E).
nth1b(N,[_|T],E):-N1 is N-1,nth1b(N1,T,E).
```

```
5 ?- nth1b(3,[a,b,c,d,e],X).
X = c ;
false.

6 ?- nth1b(Y,[a,b,c,d,e],c).
ERROR: nth1b/3: Arguments are not sufficiently instantiated
```

Operacje na listach

```
nth1c(1,[E|_],E).
nth1c(N,[_|T],E):-N is N1+1,nth1c(N1,T,E).
```

```
1 ?- nth1c(3,[a,b,c,d,e,f],L).
ERROR: is/2: Arguments are not sufficiently instantiated
2 ?- nth1c(Y,[a,b,c,d,e,f],c).
ERROR: is/2: Arguments are not sufficiently instantiated
3 ?- nth1d(3,[a,b,c,d,e,f],L).
L = c ;
false.

4 ?- nth1d(Y,[a,b,c,d,e,f],c).
Y = 3 ;
false.

5 ?- nth1d(Y,[a,b,c,d,e,f,c],c).
Y = 3 ;
Y = 7 ;
false.
```

```
nth1d(1,[E|_],E).
nth1d(N,[_|T],E):-nth1d(N1,T,E),N is N1+1.
```

Operacje na listach

```
nth1e(1,[E|_],E).
nth1e(N,[_|T],E):-nth1e(N1,T,E),N1 is N-1.
```

```
6 ?- nth1e(3,[a,b,c,d,e,f],L).
ERROR: is/2: Arguments are not sufficiently instantiated
7 ?- nth1e(Y,[a,b,c,d,e,f,c],c).
ERROR: is/2: Arguments are not sufficiently instantiated
```

Literatura

- W. Clocksin, C. Mellish, *Prolog. Programowanie*, Wyd. Helion.
- E.Gatnar, K.Stąpor, *Prolog*, Wyd. PLJ.
- G.Brzykcy, A.Meissner, *Programowanie w Prologu i programowanie funkcyjne*,Wyd.PP.
- M. Ben-Ari, *Logika matematyczna w informatyce*, WNT.
- http://lpn.swi-prolog.org/lpnpage.php?pageid=online