Programowanie deklaratywne

Artur Michalski
Informatyka II rok

Plan wykładu

- Wprowadzenie do języka Prolog
- Budowa składniowa i interpretacja programów prologowych
- *Listy,* operatory i operacje arytmetyczne
- Sterowanie mechanizmem nawrotów
- Predefiniowane procedury prologowe
- Styl i technika programowania w Prologu

Listy, operatory i operacje arytmetyczne

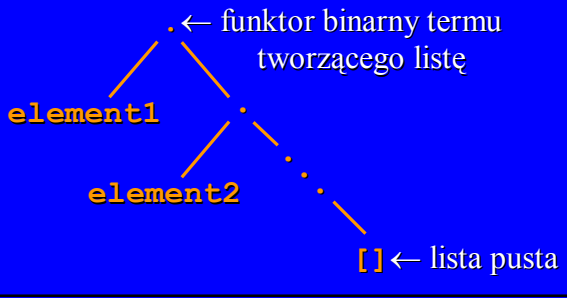
- Reprezentacja list w Prologu
- Wybrane operacje na listach w Prologu
- Notacja operatorów
- Operacje arytmetyczne w Prologu

Reprezentacja list w Prologu

Lista to dowolnej długości ciąg termów zapisywany w postaci:

```
[element1,element2,...]
```

Reprezentacja wewnętrzna listy odpowiada strukturze drzewiastej:



Reprezentacja list w Prologu

Budowa listy (niepustej):

- *głowa* - pierwszy element listy,
- *ogon* - pozostałe elementy listy

Własności:

- Głową listy może być dowolny obiekt języka Prolog np. inna lista, zmienna, czyli dowolny term
- Ogon listy jest zawsze listą i może być listą pustą
- Lista jest strukturą rekurencyjną - jeżeli ogon jest niepusty, to również on składa się z głowy i z ogona

Reprezentacja list w Prologu

Przykładowe listy:

```
?-Hobby1=[muzyka,kuchnia],
  Hobby2=[narty,taniec],
  Lista=[tenis,Hobby1,film,Hobby2].

Hobby1=[muzyka,kuchnia],
Hobby2=[narty,taniec],
Lista=[tenis,[muzyka,kuchnia],film,[narty,
  taniec]].
```

Elementem składowym listy może być inna lista.

```
?- Pets = .(dogs,.(cats,[])).
Pets = [dogs, cats]
```

Funktor '.' może być wykorzystywany jawnie przy tworzeniu list.

Reprezentacja list w Prologu

Alternatywna notacja list

W prologu listy mogą być zapisywane w sposób odzwierciedlający ich budowę:

```
[Head | Tail]
```

gdzie:
`Head` - może być ciągiem dowolnych elementów (termów) oddzielonych przecinkiem, a `Tail` - dowolna listą elementów

Przykład

```
[a,b,c] = [a|[b,c]] = [a,b|[c]] = [a,b,c|[]]
```

Wybrane operacje na listach w Prologu

Operacja sprawdzania przynależności elementu do listy

Klauzula `member(X,L)` ma być prawdziwa, jeżeli element `X` należy do listy `L`. Przykładowo:
`member(b,[a,b,c])` - powinno być prawdziwe
`member(b,[a,[b,c]])` - powinno być fałszywe, ale
`member([b,c],[a,[b,c]])` - powinno być prawdziwe

Definicja
X należy do listy L, o ile
X jest głową listy L lub
X należy do ogona listy L.

Zapis w Prologu:

```
member(X,[X|Tail]).
member(X,[Head|Tail]):- member(X,Tail).
```

Wybrane operacje na listach w Prologu

Operacja łączenia (konkatenacji) list

Klauzula `conc(L1,L2,L3)` łączy listę `L1` z listą `L2` w listę wynikową `L3`.

Przykłady
`conc([a,b],[c,d],[a,b,c,d])` - jest prawdziwe
`conc([a,b],[c,d],[a,b,a,c,d])` - jest fałszywe

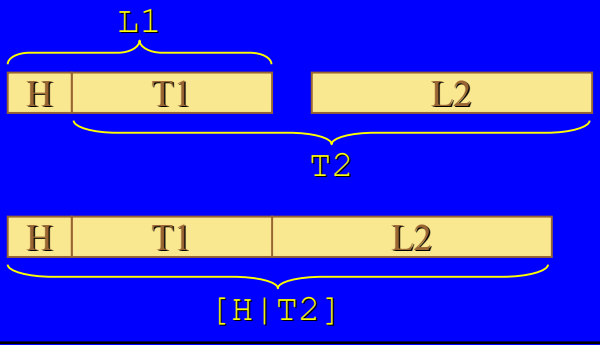
Definicja
Jeżeli lista L1 *jest pusta, to lista* L3 *jest taka sama jak lista* L2.
Jeżeli lista L1 *jest niepusta i ma postać* [H|T1], *to lista* L3 *ma postać* [H|T2] , *gdzie* T2 *jest połączeniem list* T1 *i* L2.

Wybrane operacje na listach w Prologu

Operacja łączenia (konkatenacji) list c.d.

Zapis konkatenacji w Prologu:

```
conc([],L,L).
conc([H|T1],L2,[H|T2]):- conc(T1,L2,T2).
```



Wybrane operacje na listach w Prologu

Operacja łączenia (konkatenacji) list c.d

Zastosowanie operacji konkatenacji - ***dekompozycja listy***:

```
?- conc(L1,L2,[a,b,c]).
L1=[]
L2=[a,b,c];

L1=[a]
L2=[b,c];

L1=[a,b]
L2=[c];

L1=[a,b,c]
L2=[];

No
```

Wybrane operacje na listach w Prologu

Operacja łączenia (konkatenacji) list c.d.

Zastosowanie operacji konkatenacji - ***szukanie podlist***:

```
?- conc(Przed,[sr|Po],
        [pon,wt,sr,czw,pt,sob,nd]).

Przed=[pon,wt]
Po=[czw,pt,sob,nd]
```

Zastosowanie operacji konkatenacji - ***szukanie poprzednika i następnika***:

```
?- conc(_,[Przed,sr,Po|_],
        [pon,wt,sr,czw,pt,sob,nd]).

Przed=wt
Po=czw
```

Wybrane operacje na listach w Prologu

Operacja łączenia (konkatenacji) list c.d.

Zastosowanie operacji konkatenacji - ***usuwanie podlisty***:

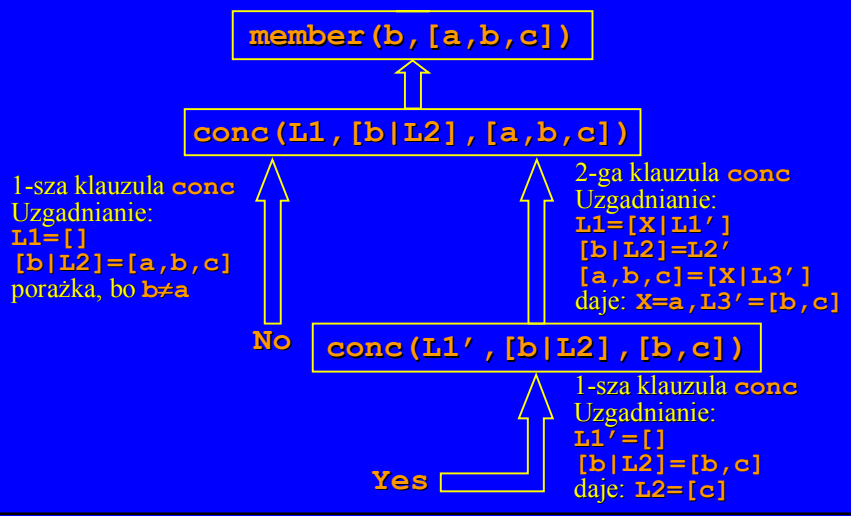
```
?- L1=[a,b,z,z,c,z,z,z,d,e],
conc(L2,[z,z,z|_],L1).

L1=[a,b,z,z,c,z,z,z,d,e]
L2=[a,b,z,z,c]
```

Zastosowanie operacji konkatenacji - ***inna wersja*** `member`:

```
member(X,L):- conc(L1,[X|L2],L).
```
albo:
```
member(X,L):- conc(_,[X|_],L).
```

Wybrane operacje na listach w Prologu

Zastosowanie operacji konkatenacji - ***inna wersja*** `member` :



Wybrane operacje na listach w Prologu

Operacja dodawania elementu do listy

Najprostszy wariant - dodać element na początku listy (nowa głowa listy).

Zapis w Prologu:

```
add(X,L,[X|L]).
```

Definiowanie tego predykatu nie jest konieczne, gdyż zawsze można otrzymać nową listę `Y`, powiększając daną listę `L` o nowy element `X`, dzięki jawnemu uzgadnianiu termów:

```
Y=[X|L]
```

Wybrane operacje na listach w Prologu

Operacja usuwania elementu z listy

Klauzula `del(X,L,L1)` ma być prawdziwa, jeżeli lista `L1` jest równa liście `L` pomniejszonej o element `X`. Położenie elementu `X` jest dowolne.

Definicja

Jeżeli X jest głową listy L, to lista L1 jest ogonem listy L.
Jeżeli X należy do ogona listy L, to usuń stamtąd X.

Zapis w Prologu:

```
del(X,[X|Tail],Tail).
del(X,[H|Tail],[H|Tail1]):-
                    del(X,Tail,Tail1).
```

Wybrane operacje na listach w Prologu

Operacja usuwania elementu z listy c.d.

Operacja `del(X,L,L1)` usuwa dowolne, ale tylko jedno (a nie wszystkie!) wystąpienie `X` z listy `L`. Działanie takie określamy mianem *niedeterminizmu* predykatu.

Przykład

```
?- del(a,[a,b,a,a],L).
L=[b,a,a];
L=[a,b,a];
L=[a,b,a];
No
```

Inna niedeterministyczna klauzula: `member(X,L)` .

Wybrane operacje na listach w Prologu

Operacja usuwania elementu z listy c.d.

Zastosowanie operacji `del(X,L,L1)` - ***wstawianie elementu do listy***:

```
?- del(a,L,[1,2,3]).
L=[a,1,2,3];
L=[1,a,2,3];
L=[1,2,a,3];
L=[1,2,3,a];
No
```

Zatem definicja w Prologu:

```
insert(X,L,BiggerL):- del(X,BiggerL,L).
```

Wybrane operacje na listach w Prologu

Operacje na podlistach

Klauzula `sublist(S,L)` jest prawdziwa, jeśli lista `S` zawiera się w liście `L`.

Przykłady

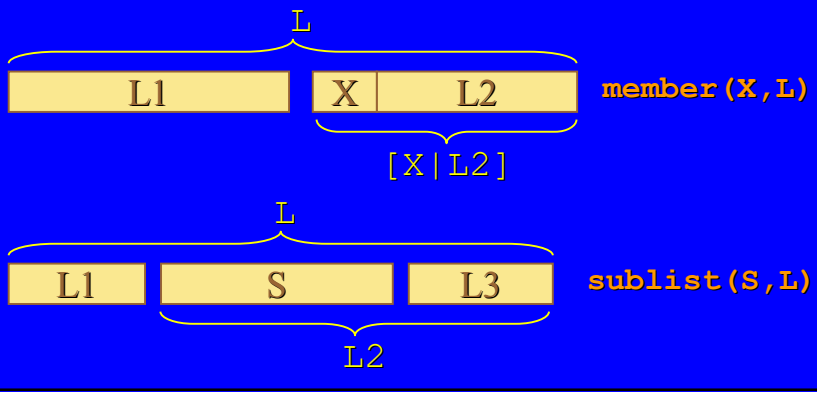
`sublist([c,d,e],[a,b,c,d,e,f])` - jest prawdziwe

`sublist([c,e],[a,b,c,d,e,f])` - jest fałszywe

Wybrane operacje na listach w Prologu

Operacje na podlistach c.d.

Analogia między klauzulą `member(X,L)` (w wersji z `conc`) a klauzulą `sublist(S,L)`



Wybrane operacje na listach w Prologu

Operacje na podlistach c.d.

Definicja
Lista S *należy do listy* L, *o ile lista* L *składa się z dwóch list* L1 *i* L2, a lista L2 *jest połączeniem list* S *i* L3.

Zapis w Prologu

```
sublist(S,L):- conc(L1,L2,L),conc(S,L3,L2).
```

Przykład

```
?- sublist(S,[a,b,c]).
S=[];
S=[a];
S=[a,b];
S=[a,b,c];
S=[b];
...
```