

1. Funcția G calculează produsul primelor 2 elemente ale unei liste.

(setg car 'G') \rightarrow car se va evalua ea G

Totuși, pentru a apela funcția G prin intermediul lui CAR , va fi nevoie de a evalua:

- una pt. a ajunge de la CAR la G
- una pt. a evalua G ca funcția pe care o reprezintă

Dacă în locurile lui CAR ar fi fost o denumire ce nu coincide cu denumirea unei funcții predefinite, apăsarea din cerință ar fi dat eroare, negăsind o funcție cu numele respectiv (s-ar fi evaluat CAR la G , dar ar mai fi fost nevoie de încă o evaluare pentru apel). Totuși, există o funcție cu numele CAR și atunci se va repeta definiția acesteia.

\Rightarrow Rezultatul lui: $(CAR (2 3 5 6))$ este 2.

3. Aranjamente de k elemente cu o sumă dată.
(PROLOG)

Modele matematice:

inserează($e_1 \dots e_n, x$) =

1. $x \oplus e_1 \dots e_n$

2. $e_1 \oplus \text{inserează}(e_2 \dots e_n, x)$, $n > 1$

aranj($e_1 \dots e_n, k, S$) =

1. (e_1) , dacă $k=1$ și $S=e_1$

2. aranj($e_2 \dots e_n, k, S$)

3. inserează(aranj($e_2 \dots e_n, k-1, S-e_1$), e_1),
dacă $e_1 < S$ și $k > 1$

$$\text{main}(e, R, S) = \bigcup \text{aranj}(e, R, S)$$

$\text{insereaza}(L, E, [E \setminus L])$.

$\text{insereaza}([H \setminus T], E, R) :-$

$\text{insereaza}(T, E, R_1),$

$R = [H \setminus R_1]$.

model de flux: (i, i, α)

non-determinist

$\text{aranj}([H \setminus T], 1, H, [H \setminus T])$.

$\text{aranj}([T \setminus I], K, S, R) :-$

$\text{aranj}(T, K, S, R_1)$.

$\text{aranj}([H \setminus T], K, S, R) :-$

$K > 1,$

$S > H,$

R_1 is $K - 1,$

S_1 is $S - H,$

$\text{aranj}(T, K_1, S_1, R_1),$

$\text{insereaza}(R_1, H, R)$.

model de flux:

(i, i, i, α)

non-determinist

$\text{main}(L, K, S, R) :-$

$\text{findare}(R, \text{aranj}(L, K, S, R_1), R)$.

model de flux: $(i, i, i, \alpha) \rightarrow$ determinist

4. Nr. de moduri de pe nivelul R. (Lisp)

(nivel superficial = 1)

modele matematice:

$$\text{numara}(e, R, n) = \begin{cases} 1, & e \text{ atom si } n = R \\ 0, & e \text{ atom si } n \neq R \\ \sum_{i=1}^n \text{numara}(e_i, R, n_i + 1), & \text{altfel} \end{cases}$$

$\text{maim}(e, l_2) = \text{numara}(e, l_2, 0)$

↳ pt. ca numărul prezintă
lista întreagă (niv=0), iar
pt. a ajunge la niv. superioară
trebuie să trecem la un niv.
superior (niv=1)

(defun numara (e l_2 niv)

(cond

((and (atom e) (equal l_2 niv)) 1)

((atom e) 0)

(+ (appex #) + (mapcar #) (cmeda l_2)

(numara y l_2 (+ niv 1))

)

)

)

)

(defun maim (e l_2)

(numara e l_2 0)

)

5. Nr. de succese pt. care primul atom numeric este
impar.

Modele matematice:

$$\text{prim}(e_1 \dots e_n) = \begin{cases} \emptyset, n = 0 \\ e_1, e_1 \text{ e nr.} \\ \text{prim}(e_1), e_1 \text{ e lista \à prim}(e_1) \neq \emptyset \\ \text{prim}(e_2 \dots e_n), \text{ altfel} \end{cases}$$

$$\text{succese}(e) = \begin{cases} 0, e \text{ e atom} \\ 1 + \sum_{i=1}^3 \text{succese}(e_i), \text{ prim}(e) \text{ este nr. impar} \\ \sum_{i=1}^3 \text{succese}(e_i), \text{ altfel} \end{cases}$$

```
(defun prim (e)
```

```
  (cond
```

```
    ((null e) nil)
```

```
    ((numberp (car e)) (car e))
```

```
    ((and (listp (car e)) (prim (car e)))  
     (prim (car e)) )
```

```
    (+ (prim (car e)) )
```

```
  )
```

```
)
```

```
(defun sieve (e)
```

```
  (cond
```

```
    ((atom e) 0)
```

```
    ((and (numberp (prim e)) (oddp (prim e)))  
     (+ 1 (apply #' + (mapcar #' sieve e) ) ) )
```

```
    (+ (apply #' + (mapcar #' sieve e) )
```

```
  )
```

```
)
```