

SEMINAR 1 - Recursivitate

I Alg. recursiv cu numere

II Alg. recursiv cu liste

I Algoritm recursiv cu numere

1 Să se verifice, dacă un număr natural este „monocic”

Se consideră monocic dacă e format exclusiv din cifre 2 și 6

$$\text{monocic}(n) = \begin{cases} A, & n=2 \vee n=6 \\ F, & n < 10 \wedge n \neq 2 \wedge n \neq 6 \\ F, & n \geq 10 \wedge n \% 10 \neq 2 \wedge n \% 10 \neq 6 \\ \text{monocic}([n/10]), & \text{altfel} \end{cases}$$

Implementare python

```
def monocic(n):
```

```
    if n == 2 or n == 6:
```

```
        return True
```

```
    elif n < 10 and n != 2 and n != 6:
```

```
        return False
```

```
    elif n >= 10 and n % 10 != 2 and n % 10 != 6:
```

```
        return False
```

```
    else:
```

```
        return monocic(n // 10)
```

2 Să se calculeze suma divizorilor proprii și improprii ai unui număr natural săpată dat

$$\text{ex. } 6 \Rightarrow 1+2+3+6 = 12$$

$$\text{suma}(n, \text{div}) = \begin{cases} \text{div} + \text{suma}(n, \text{div}-1), & n \% \text{div} = 0 \\ > 1 \\ \text{suma}(n, \text{div}-1), & n \% \text{div} \neq 0 \\ > 1 \\ 1, & \text{div} = 1 \end{cases}$$

$$\text{sumaMaior}(n) = \text{suma}(n, n)$$

II Algoritm recursiv cu liste

$l_1, l_2 \dots l_m$ - lista cu n elemente
 \mapsto completă înțărtită

DA - se poate direct

NU - se poate cu recursivitate

- $n=0$
- n ? constantă
- l_1, l_2 (un m constant de elem de la început)
- $l_2 \dots l_m$
- $e \cdot l_1, l_2 \cdot l_m$

- $n > k$ (var)
- n par ?
- l_k (k - variabilă)
- $l_k \cdot l_{k+1} \dots l_m$
- $l_1, l_2 \dots e \cdot l_m$
- $l_1, l_2 \dots l_m \in$

1 Se calculează produsul elem. pere ale unor liste numerice formate din nr. naturale

$$\text{prod}(l_1, l_2 \dots l_m) = \begin{cases} 1, & n=0 \\ l_1 \cdot \text{prod}(l_2 \dots l_m), & n>0 \text{ și } l_1 \% 2 = 0 \\ 1 \cdot \text{prod}(l_2 \dots l_m), & n>0 \text{ și } l_1 \% 2 \neq 0 \end{cases}$$

eVidă $(l_1, l_m) \Rightarrow A/F$

prum $(l_1, l_m) \Rightarrow l_1$

sublista $(l_1, l_m) \Rightarrow l_2 \dots l_m$

crecasă ()

adaugăÎnceput(l_1, l_m, e)

Implementare

```

def par(l)
    if eVidă(l):
        return 1
    elif par(l) % 2 == 0:
        return par(mulista(l))
    else:
        return par(mulista(l))

```

2 Să se implementeze un elemente pe o poziție m , nr. natural,
într-o listă dată

$$\text{ex: } l = [1, 2, 3, 4]$$

$$e = 0$$

$$m = 3$$

$$l = [1, 2, 0, 3, 4]$$

$$m = 5$$

$$l = [1, 2, 3, 4, 0]$$

$$m = 100$$

$$l = [1, 2, 3, 4]$$

$$\text{imr}(l_1 \dots l_m, e, m) = \begin{cases} [e], & m=0 \text{ și } m=1 \\ l_1 \cup \text{imr}(l_2 \dots l_m, e, m-1), & m>1 \\ e \cup l_1 \dots l_m, & m=1 \end{cases}$$

Exemplu.

$$\begin{aligned}
\text{imr}([1, 2, 3], 0, 5) &= 1 \cup \text{imr}([2, 3], 0, 4) \\
&= 1 \cup 2 \cup \text{imr}([3], 0, 3) \\
&= 1 \cup 2 \cup 3 \cup \text{imr}([], 0, 1) \\
&= 1 \cup 2 \cup 3 \cup [0] \\
&\Rightarrow 1 \cup 2 \cup [3, 0] \Rightarrow \dots \Rightarrow 1, 2, 3, 0
\end{aligned}$$

3) La fel ca la 2, dar să se impună dim m în m.

$$\text{ims}(l_1..l_m, e, m, m) = \begin{cases} -\text{II}- \\ -\text{II}- \end{cases}$$

$$\begin{cases} l_1 \vee \text{ims}(l_2..l_m, l, m-1, m), -\text{II}- \\ e \vee (l_1..l_m, e, m, m), -\text{II}- \end{cases}$$

$$\text{imsMam}(l_1..l_m, e, m) = \begin{cases} \text{ims}(l_1..l_m, e, m, m), m > 1 \\ l_1..l_m, altfel \end{cases}$$

SEMINAR 2

Liste liniare în prolog

1. Dămdu-se o listă liniară, să se eliminate toate elem. care apar o singură dată

$$\text{ex } [1, 2, 3, 2, 1, 5, 1] \Rightarrow [1, 2, 2, 1, 1]$$

$$\text{mnAparițuElement}(l_1, l_2, l_m, e) = \begin{cases} 0, & m=0 \\ 1 + \text{mnAparițuElement}(l_2, l_m, e), \\ m>0 \wedge l_i = e \\ \text{mnAparițuElement}(l_2..l_m, e), \\ m>0 \wedge l_i \neq e \end{cases}$$

% mnAparițuElement (l: lista, E element, R. int)

% model flux

% l lista în care numărăm ap el E

% E el ale cărui ap le numărăm

% R. nr. de apariții ale lui E în l

cu literă mare
⇒ variabilă
cu literă mică
⇒ constantă

$$\text{mnAparițuElement}([], \underline{E}, 0) \rightarrow \text{nu mă interesează valoarea el.}$$

$$\text{mnAparițuElement}([+|T], E, \underline{R}) \cdot -H = E, \text{mnApEl}(T, E, R1), R \text{ is } R1 + 1$$

nu e legat

explicație dacă am găsit primul el. din listă - el și după am găsit R1 apariții în restul listei, îl legăm pe R, acesta devine R1 + 1

$$\text{mnAparițuElement}([+|T], E, R) - (H) = E, \text{mnApEl}(T, E, R)$$

$$\text{elumină } (l_1 l_2 \dots l_m, m) = \begin{cases} \emptyset, & m=0 \\ l_1 \cup \text{elumină } (l_2 \dots l_m, m), & m \neq 0 \text{ și element } (m, l_1) > 1 \\ \text{elumină } (l_2 \dots l_m, m), & altfel \end{cases}$$

$$\text{elumină-pp}(l) = \text{elumină } (l, l)$$

% elumină (λ linișt, M linișt, λ_{neq} linișt)

% modelul de flux $\cdot (\lambda, \epsilon, \sigma)$

elumină $([], \dots, [])$

elumină $([H|T], M, [H|L_{\text{neq}}]) = m \wedge A_p E_l (H, M, A_p),$
 $A_p > 1,$
 $|$
 $\text{elumină } (T, M, L_{\text{neq}})$

elumină $([-|T], M, L_{\text{neq}}) = \text{elumină } (T, M, L_{\text{neq}})$

% elumină-pp (λ . linișt, λ_{neq} linișt)

% modelul de flux $\cdot (\lambda, \epsilon)$

elumină-pp $(\lambda, \lambda_{\text{neq}}) = \text{elumină } (\lambda, \lambda, \lambda_{\text{neq}})$

2.

adăugăFinal ($l_1 \cdot l_m, e$) = $\begin{cases} [e], m=0 \\ l_1 \oplus \text{adăugăFinal } (l_2 \cdot l_m, e), m>0 \end{cases}$
 (asta e un predicat)

% adăugăFinal (λ list, E · element, lneș list)

% model flux (i, α, \circ)

adăugăFinal ([] , E , [E])

adăugăFinal ([H \ T] , E , lneș) - adăugăFinal (T , E , Lneș),
 lneș = [H \ lneș]

elimină2 ($l_1 \cdot l_m, m, C$) = $\begin{cases} c, m=0 \\ \text{elimină2 } (l_2 \cdot l_m, m, \text{adăugăFinal } (c, l_1)), \\ m>0 \wedge \text{mApant } (m, l_1) > 1 \\ \text{elimină2 } (l_2 \cdot l_m, m, C), \text{mAp } (m, l_1) = 1 \end{cases}$

elimină2_pp (l) = elimină2 (l, l, Ø)

% elimină2 (λ list, M list, C · list, lneș · list)

% model flux (i, α, α, \circ)

elimină2 ([] , ⊕ , C , C) → putem să punem M, dar trebuie
 să sună ca nu mai e folosit în
 acel apel, deoarece sună

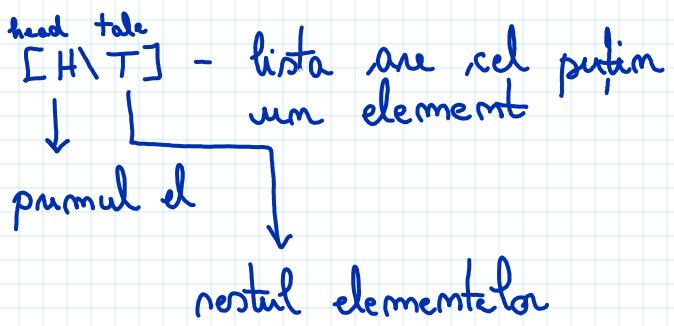
elimină2 ([H \ T] , M , C , lneș) - mAp (H , H , R), R > 1 ,
 adăugăFinal (C , H , C),
 elimină2 (T , M , C , lneș).

elimină2 ([T] , M , C , lneș) - elimină2 (T , M , C , lneș)

nu mai punem cond.
 la următoarea

NOTIȚE

8



\mid (cut)

- opriște backtrackingul pentru acelasi predicat
- se contamă și avem în clauză, dar nu mai intră în următoarele clauze de acelasi tip
- îl punem după condiție

$=$ (compară. von - comst, 2 von core
 an o valoare ex: $E = V$ ✓
 $E = V + 1$ ✗

$\mid\mid$ (leg o variabila de o valoare / expr)
 ex V is $B+1$

$= . =$ (fătează evaluarea a ce avem în stânga, apoi dr,
 apoi verifică egalitatea)

ex $A + B = . = B - 2$

$A = = B$ e ca $A = B$

$\backslash =$ (mot $E = H$)

Pentru liste funcționează doar " $=$ ",
 nu funcționează " $\mid\mid$ "

TEMA LAB 1

6.

- a. Sa se scrie un predicat care elimina dintr-o lista toate elementele care se repeta (ex: $l=[1, 2, 1, 4, 1, 3, 4] \Rightarrow l=[2, 3]$)
- b. Sa se eliminate toate aparitiile elementului maxim dintr-o lista de numere intregi.

6

a)

$$m \text{ Apariții Element}(l_1, l_2, l_m, e) = \begin{cases} 0, & m=0 \\ 1 + m \text{ Apariții Element}(l_2..l_m, e), \\ & m>0 \quad \exists i \quad l_i = e \\ m \text{ Apariții Element}(l_2..l_m, e), \\ & m>0 \quad \nexists i \quad l_i \neq e \end{cases}$$

$$\text{elimină}(l_1, l_2..l_m, m) = \begin{cases} \emptyset, & m=0 \\ \text{copie lista } \leftarrow \text{elimină}(l_2..l_m, m), & \text{dacă} \\ & m \text{ Apariții Element}(m, l_1) > 1 \\ & l_1 \cup \text{elimină}(l_2..l_m, m), \text{ altfel} \end{cases}$$

$$\text{elimină_pp}(l) = \text{elimină}(l, l, \emptyset)$$

b)

$$\text{elMax}(l_0, l_1, \dots, l_m) = \begin{cases} l_0, & m=1 \\ l_0, & m>0 \text{ și } l_0 > \text{elMax}(l_1, l_2, \dots, l_m) \\ \text{elMax}(l_1, l_m), & m>0 \text{ și } l_0 < \text{elMax}(l_1, l_2, \dots, l_m) \end{cases}$$

$$\text{eliminăM}(l_0, l_1, \dots, l_m, el) = \begin{cases} \emptyset, & m=0 \\ \text{eliminăM}(l_1, \dots, l_m, el), & \text{dacă } l_0 = el \\ l_0 \cup \text{eliminăM}(l_1, \dots, l_m, el), & \text{altfel} \end{cases}$$

$$\text{elMaxime}(l) = \text{eliminăM}(l, \text{elMax}(l, el))$$

SEMINAR 5

Lista extogeme în prolog

+

Se dă o listă formată din numere și liste de numere. Se cere să se determine nr. el. de tip lista având aspect de număr.

ex $[1, 3, 2, [1, \overbrace{3}, 5], 2, [5, 3, 1], 0, [9, \overbrace{10}, 1], 6, [6, \overbrace{6}, 7, 5], [1, 3, 2, 2], [11, 12, 10, 8], 6] \Rightarrow 2$

Varianta 1.

- merg cu un predicat care verifică în cînd ajung la 2 nr. descr. apelăz alt predicat care verifică dacă restul sunt

Varianta 2

- pe part. cercatoare merg cu un flag = 0 și când, dacă de 2 el. descr. flag = 1

V2

$\text{isMountain}(l_1..l_m, \text{flag}) = \begin{cases} \text{true} , m=1 \wedge \text{flag} = 1 \\ \text{isMountain}(l_1..l_m, 0) , m>1 \wedge l_1 < l_2 \wedge \text{flag} = 0 \\ \text{isMountain}(l_2..l_m, 1) , m>1 \wedge l_1 > l_2 \\ \text{false} , \text{altfel} \end{cases}$

numere ($l_1..l_m$) = $m > 2 \wedge l_1 < l_2 \wedge \text{isMountain}(l_2..l_m, 0)$

SAU

-1 (neutral) 0 (\uparrow) 1 (\downarrow)

numere ($l_1..l_m$) = $\text{isMountain}(l_1..l_m, -1)$

2.

- a) Sa se sorteze o lista cu pastrarea dublurilor. De ex: [4 2 6 2 3 4] => [2 3 4 4 6]
 b) Se da o lista eterogena, formata din numere intregi si liste de numere. Sa se sorteze fiecare sublista cu pastrarea dublurilor. De ex:
 [1, 2, [4, 1, 4], 3, 6, [7, 10, 1, 3, 9], 5, [1, 1, 1], 7] =>
 [1, 2, [1, 4, 4], 3, 6, [1, 3, 7, 9, 10], 5, [1, 1, 1], 7].

$$a) \text{elMin}(l_0..l_m, e) = \begin{cases} l_0, & m=1 \\ l_0, & m>0 \wedge l_0 \leq \text{elMin}(l_1..l_m) \\ \text{elMin}(l_1..l_m), & l_0 > \text{elMin}(l_1..l_m) \end{cases}$$

$$\text{adF}(e, l_0..l_m) = \begin{cases} e, & m=0 \\ \text{adF}(e, l_1..l_m), & \text{altfel} \end{cases}$$

$$\text{sortare}(l_0..l_m) = \begin{cases} \emptyset, & m=0 \\ l_0, & m=1 \\ \text{sortare}(l_1..l_m), & \text{daca } l_0 \leq \text{elMin}(l_1..l_m) \\ \text{sortare}(\text{adF}(e, l_1..l_m)), & \text{altfel} \end{cases}$$

b)

$$\text{listates}(l_0..l_m) = \begin{cases} \emptyset, & m=0 \\ l_0, & m=1 \\ \text{listates}(\text{sortare}(l_0..l_m)), & l_0 \in \text{listate} \\ \text{listates}(l_1..l_m), & \text{altfel} \end{cases}$$

SEMINAR 1

Backtracking în prolog

Recapitulare utilizare întăritură

impar (1)

impar (3)

par (2)

par (4)

impredică back pt clauzele care urmărează pt predicatul p respectiv, dar clauza în care suntem se efectuează

parYimpar (x,y) - !; impar (x), par (y). → asta tot se face

parYimpar (x,y) - par (x), impar (y) → asta nu se mai face

fără ! $\Rightarrow (1,2), (1,4), (3,2), (3,4), (2,1), (2,3), (4,1), (4,3)$

$\xrightarrow{x \text{ se leagă de cîte trb, dar nu poate revinere pt } y, \text{ ptc am opit bîng}}$

parYimpar (x,y) - impar (x), !; par (y). $\Rightarrow (1,2), (1,4)$

parYimpar (x,y) - par (x), impar (y)

parYimpar (x,y) - impar (x), par (y), !. $\Rightarrow (1,2)$

parYimpar (x,y) - par (x), impar (y)

parYimpar (x,y) - impar (x), par (y).

parYimpar (x,y) - !; par (x), impar (y) \Rightarrow toate cele 8 soluții

parYimpar (x,y) - impar (x), par (y).

parYimpar (x,y) - par (x), !; impar (y) \Rightarrow primele 6 soluții

parYimpar (x,y) - !; impar (x), par (y).

parYimpar (x,y) - par (x), impar (y), !. \Rightarrow primele 5 soluții

generare prima
soluție pt. x

$\Rightarrow (1,2)$

mă întorc în
generare a 2-a
soluție pt x
 $\Rightarrow (1,4)$

apei avem !,
deci me opresc
dîn a mai ge-
meră

① Se dă o listă formată din numere întregi distincte

Se cere să se genereze toate sublăstile formate din elemente ale listei date, având aspect de vale.

$$\text{ex } [3, 9, 6, 2, 1, 7] \Rightarrow [6, 2, 1, 7, 9]$$

$$[3, 1, 2]$$

[3, 1, 3] × nu e valoare să se repete

! Sealătă metodă nu e acceptată la colocviu!

$$\text{candidat } (l_1..l_m) = \perp l_1, m > 0$$

$$2. \text{ candidat } (l_2..l_m), m > 0$$

$$\% \text{ candidat } (l \text{ list}, e \text{ int})$$

$$\% \text{ model de flux } (i, o) \text{ medet}$$

$$\text{candidat } ([H..], H)$$

$$\text{candidat } ([-IT], e) - (\text{candidat } (T, e)).$$

$$\text{generare } (l, c_1..c_m, f) = \perp c_1..c_m, f=0$$

$$2. \text{ generare } (l, c_1..c_m, 1), f=1, e < c_1, \text{ unde } e = \text{candidat } (l)$$

$$3. \text{ generare } (l, c_1..c_m, 0), e > c_1,$$

$$\text{candidat } (c_1..c_m) = e, \text{ unde } e = \text{candidat } (l)$$

→ prediciț medet c.m.m.t

- numerotăm ramurile, nu punem același
- ramurile nu se exclude reciproc, se poate intra pe amândouă

- nu punem altfel la ultima ramură

$$\% \text{ generare } (l \text{ list}, C: \text{list}, F: \text{int}, R: \text{list})$$

$$\% \text{ model de flux } (i, i, i, o) \text{ medet}.$$

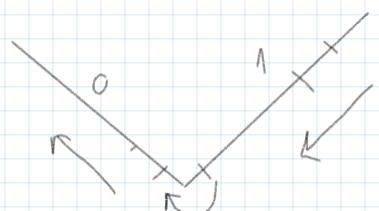
$$\text{generare } (-, c, 0, c).$$

$$\text{generare } (l, [H..T], 1, R). -$$

$$\text{candidat } (\perp, \varepsilon),$$

$$\varepsilon < H,$$

$$\text{generare } (\perp, [\varepsilon | [H..T]], 1, R)$$



generare ($L, [H \sqcup T], -, R$) :-
 candidat (L, E),
 NOT (candidat ($[H \sqcup T], E$)),
 $E > H$,
 generare ($L, [E, H \sqcup T], 0, R$)

generareMain (L) = generare ($L, [l_1 l_2], 1$), unde $l_1 =$ candidat (L)
 $l_2 =$ candidat (L)
 $l_1 < l_2$

% generareMain ($L: list, R: list$)

% model (L, R) mediat.

generareMain (L, R) :-

candidat (L, E_1),
 candidat (L, E_2),
 $E_1 < E_2$,
 generare ($L, [E_1, E_2], 1, R$).

Pentru cazul în care returnăm o listă finală cu toate rezultatele bune

main (L) = \cup generareMain (L)

% main ($L: list, R: list$)

main (L, R) :-

fndall (R_1 , generareMain (L, R_1, R))

predicat din prolog variable pe care o colectăm colecție

SEMINAR 5

Recursivitate în lisp

1 Se dau 2 liste numerice, sortate, cu elemente distincte

Se cere să se întrelasă cele două liste fără păstrarea dublurilor

$$\text{ex. } l_1 = (1, 3, 5, 7, 9)$$

$$l_2 = (2, 3, 4, 5)$$

$$\Rightarrow (1, 2, 3, 4, 5, 7, 9)$$

întrelasare ($l_1, l_m, p_1..p_m$) =

$$l, m=0$$

$$p, m=0 \quad \text{și} \quad m \neq 0$$

$l_1 \cup$ întrelasare ($l_2..l_m, p_1..p_m$), dacă
 $l_1 < p_1$ și $m \neq 0$ și $m \neq 0$

$p_1 \cup$ întrelasare ($l_1, l_m, p_2..p_m$), dacă
 $p_1 < l_1$ și $m \neq 0$ și $m \neq 0$

$l_1 \cup$ întrelasare ($l_2..l_m, p_2..p_m$), dacă
 $l_1 = p_1$, $m \neq 0$ și $m \neq 0$

nu se punne virgulă

(defun întrelasare ($l p$)

un fel de switch

(.cond \rightarrow *un fel de switch* \rightarrow dacă $p \in$ vidă, returnează l

((null p) l)

((and (null l) (not (null p))) p)

dacă $a \in A$, returnează b

((< (car l) (car p)) (comp b (car l) (întrelasare (cdr l) p)))

((> (car l) (car p)) (comp (car p) (întrelasare l (cdr p))))

((+ (comp (car l) (întrelasare (cdr l) (cdr p))))

)

)

operare (întrelasare '(.)'(''))

2. Se dă o listă melimică.

Să ne întreagă toate operațiile unui element din listă cu toate sublăncile

$$\text{ex } (1 \ A \ (2 \ B \ 1) \ (1) \ 2 \ 1) \Rightarrow (1 \ A \ (2 \ B) \ (1) \ 2 \ 1)$$

\perp

$$\text{listRemove}(l_1..l_m, e) = \begin{cases} () & m=0 \\ \text{listRemove}(l_2..l_m, e) & m>0 \text{ și } l_1 = e \\ l_1 \cup \text{listRemove}(l_2..l_m, e) & m>0 \text{ și } l_1 \neq e \end{cases}$$

(defun listRemove (l e)

(.cond

$$((\text{null } l) ())$$

$$((\text{equal } (\text{car } l) e) (\text{listRemove } (\text{cdr } l) e))$$

$$((\text{not } (\text{equal } (\text{car } l) e)) (\text{cons } (\text{car } l) (\text{listRemove } (\text{cdr } l) e)))$$

)

)

$$\text{remove}(l_1..l_m, e) = \begin{cases} () & m=0 \\ l_1 \cup \text{remove}(l_2..l_m, e) & m>0 \text{ și } l_1 \text{ nu e lista} \\ \text{listRemove}(l_1, e) \cup \text{remove}(l_2..l_m, e) & m>0 \text{ și} \\ l_1 \text{ e lista} \end{cases}$$

(defun remove (l e)

(.cond

$$((\text{null } l) ())$$

$$((\text{listp } l) (\text{cons } (\text{listRemove } (\text{car } l) e) (\text{remove } (\text{cdr } l) e)))$$

$$(\tau (\text{cons } (\text{car } l) (\text{remove } (\text{cdr } l) e)))$$

))

3) Se dă o listă liniică

Se cere să se determine lista pozițiilor elementului numărul minim.

1 2 3 4 5 6

ex. ($A[2..1\ 3\ 5\ 3\ 1]$) $\Rightarrow (3\ 6)$

$posMin(l_1..l_m, m, pos, col) = \begin{cases} col, & m=0 \\ posMin(l_2..l_m, l_1, pos+1, [pos]), & m>0 \text{ și} \\ & \text{în murmări}(l_1) \text{ și } l_1 < min \\ posMin(l_2..l_m, l_1, pos+1, pos \cup col), & m>0 \\ & \text{în murmări}(l_1) \text{ și } l_1 = min \\ posMin(l_2..l_m, m, pos+1, col), & \text{altfel} \end{cases}$

comentariu:

După numele $f(x)$, nu se deschide $()$

coms - adăugare el. la începutul linte

cdr - restul listei

car - primul el. din linta

mil = linta goală / vădu = fals

$\hookrightarrow \text{mil} / (1) / \rightarrow \text{mil} / '()$

= pt m. doar, altfel, lelenim equal

lisp

number

odd

minus

atom

} le putem folosi,
au complexitate
limiară

ad. el. început face linta

coms (2 ang)

lint

Ang

concatenare linta
append

$'A 'B$

$(A B)$

$(A B)$

erorare

$'A '(B)$

$(A B)$

$(A (B))$

erorare

$'(A) 'B$

$((A) B)$

$((A) B)$

$(A B)$

$'(A) '(B)$

$((A) B)$

$((A) (B))$

$(A B)$

$'(A) C '(B)$

erorare

$((A) C (B))$

erorare

Dacă vrem ea ceva să mu fie evaluat, punem ' λ ' în față.

$\lambda \rightarrow$ linta

Primul el. din linta e fct / operator.

(introducere $\lambda^1((1\ 2\ 3)^1\ \underline{1}\ (2\ 5))$)

punem apostrof pt că altfel incercă să caute o fct murmărită 1, resp. 2