1.

(set 'L '((T NIL T)) -> îi evaluează pe L ca lista (T NIL T)

Ceea ce ar fi de așteptat să se întâmple atunci când aplicăm OR pe lista din L, ar fi să se returneze T. Acest lucru nu se va întâmpla însă, deoarece OR nu este considerată o funcție în LISP, ci doar un operator special => vom primi o eroare care ne va spune că nu există o funcție cu numele OR.

Pentru a rezolva problema, va trebui să ne definim o funcție separată care aplică OR:

```
(defun f (e)
    (cond
        ((nuce e) nil)
        ((car e) +)
        (+ (f (cdr e)))
    )
)

(apply #'f (list L))
```

← în apply, parametrii trebuie transmiși într-o singură listă

3. Lista subsumelor care au o dimensiune pară.

Model matematic:

subsum(e1 ... en, eg) =

1. (e1), eg e impar
2. subsum(e2 ... en, eg)
3. e1 ⊕ subsum(e2 ... en, eg+1), n ≥ 1

main(e) = φ ⊕ ∪ subsum(e, 0)

```
suma([H|-],Lg,[H]):- mat(Lg mod 2 =:= 0).
suma([-|T],Lg,R):-
    suma(T,Lg,R).

suma([H|T],Lg,R):-
    Lg1 is Lg+1,
    suma(T,Lg1,R1),
    R=[H|R1].

main(L,R):-
    findall(R1,suma(L,0,R1),R2),
    R=[[]|R2].
```

modele de flux: (i, i, o)
nedeterminist

modele de flux: (i, o)
determinist

5. Nr. de subliste care au un nr. impar de atomi nenumerici pe niv. pare. (nivelul superficial = 1)

Modele matematice:

$$numara(e, niv) = \begin{cases} 1, & e \text{ atom nenumeric si } niv \text{ par} \\ 0, & e \text{ atom nenum. si } niv \text{ impar} \\ & \text{sau } e \text{ e numar} \\ \sum_{i=1}^{m} numara(e_i, niv+1), & altfel \end{cases}$$

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

```
(defun numera (e niv)
  (cond
    ((and (atom e) (not (numberp e))) (evenp niv)) 1)
    ((atom e) 0)
    (+ (apply #'+ (mapcar #'(lambda (y)
                              (numera y (+ niv 1)))
                          e)))))

(defun sublista (e)
  (cond
    ((atom e) 0)
    ((oddp (numera e 0)) (+ 1 (apply #'+
                                (mapcar #'sublista e))))
    (t (apply #'+ (mapcar #'sublista e)))))
```