Calculabilitate & Complexități  
Subiectul 3

REC, RE, NRE

horizontal line

# Ce tre să știi?

Multimi recursive, recursiv enumerabile, nerecursiv enumerabile.

Nota 6:

- definitiile celor 3 tipuri de multimi

- relatiile intre ele (enunturi si exemple de limbaje, fara demonstratii)

Fiecare demonstratie, la alegere: 2p

# Definiții



* Spunem că limbajul L este **recursiv (REC)** dacă una dintre următoarele condiții este îndeplinită:
  + Fie . Atunci, avem două opțiuni:
    - Funcția f este recursivă (total)
    - Funcția f este calculabilă cu Programe Standard
  + Există o mașină Turing M care se oprește pe fiecare intrare și acceptă L
* Spunem că limbajul L este **recursiv enumerabil (RE)** dacă există o mașină Turing care acceptă L (dar care poate să nu se oprească pe fiecare intrare)
* Spunem că limbajul L este **nerecursiv enumerabil (NRE)** dacă nu există nicio mașină Turing care acceptă L.

Atunci, putem defini cele 3 mulțimi din diagramă:

# Proprietăți și relații între ele

Notăm C(A) = complementul mulțimii A.

1. REC este închisă la
2. RE este închisă la
3. RE nu este închisă la C

# Demonstrații

## REC este închisă la

Fie .

Atunci, există mașina Turing Mi cu L(Mi) = Li care se oprește pe fiecare input, unde .

### Construim mașina Turing M:

* Input: w
* Simulează mașina M1 pe intrarea w:
  + M1 respinge => M respinge
  + M1 acceptă => Simulează M2 pe intrarea w:
    - M2 acceptă => M acceptă
    - M2 respinge => M respinge

Fie L(M) ⇔ w e acceptat și de M1, și de M2 ⇔ L(M1)L(M2) ⇔ L1L2.

Deci, L1L2 este limbaj recursiv.

### Construim mașina Turing M:

* Input: w
* Simulează mașina M1 pe intrarea w:
  + M1 acceptă => M acceptă
  + M1 respinge => Simulează M2 pe intrarea w:
    - M2 acceptă => M acceptă
    - M2 respinge => M respinge

Fie L(M) ⇔ w e acceptat de M1 sau de M2 ⇔ L(M1)L(M2) ⇔ L1L2.

Deci, L1L2 este limbaj recursiv.

### Construim mașina Turing M:

* Input: w
* Simulează mașina M1 pe intrarea w:
  + M1 acceptă => M respinge
  + M1 respinge => M acceptă

Fie L(M) ⇔ w nu e acceptat de M1 ⇔ L(M1) ⇔ C(L1).

Deci, C(L1) este limbaj recursiv.

## RE este închisă la

Fie .

Atunci, există mașina Turing Mi cu L(Mi) = Li care se pot să nu se oprească pe input-urile respinse, unde .

### Construim mașina Turing M:

* Input: w
* Simulează mașina M1 pe intrarea w:
  + M1 se oprește și respinge => M se oprește și respinge
  + M1 nu se oprește => M nu se oprește
  + M1 se oprește și acceptă => Simulează M2 pe intrarea w:
    - M2 se oprește și acceptă acceptă => M acceptă
    - M2 se oprește și respinge => M se oprește și respinge
    - M2 nu se oprește => M nu se oprește

Fie L(M) ⇔ w e acceptat și de M1, și de M2 ⇔ L(M1)L(M2) ⇔ L1L2.

Deci, L1L2 este limbaj recursiv enumerabil.

### Construim mașina Turing M:

* Input: w
* Simulează alternativ mașinile M1 și M2 - efectuează un pas al mașinii M1, un pas al mașinii M2:
  + M1 sau M2 se oprește si acceptă => M acceptă
  + M1 sau M2 se oprește și respinge => așteaptă rezultatul celeilalte
    - Și cealaltă mașină s-a oprit:
      * A acceptat => M acceptă
      * A respins => M respinge
    - Cealaltă mașină nu se oprește => M nu se oprește
  + Nici M1, nici M2 nu se oprește => M nu se oprește

Fie L(M) ⇔ w e acceptat de M1 sau de M2 ⇔ L(M1)L(M2) ⇔ L1L2.

## 

### „=>”

L și C(L) sunt recursiv enumerabile, deci există mașinile Turing M1 și M2 cu L(M1) = L și L(M2) = C(L) care pot să nu se oprească pe intrările ce nu sunt acceptate.

### Construim mașina Turing M:

* Input: w
* Simulează alternativ mașinile M1 și M2 - efectuează un pas al mașinii M1, un pas al mașinii M2:
  + M1 se oprește:
    - Acceptă => Macceptă
    - Respinge => Mrespinge
  + M2 se oprește
    - Acceptă => Mrespinge
    - Respinge => Macceptă
  + M1 nu se oprește - o să se oprească M2 pentru că C(L)
  + M2 nu se oprește - o să se oprească M1 pentru că L.

Fie L ⇔ w e acceptat de M1 sau respins de M2 ⇔ L și .

Deci, există o mașină Turing care acceptă L și se oprește pe fiecare input => L limbaj recursiv.

### „<=”

REC este închis la C, deci și C(L) aparține lui REC, iar REC este inclus în RE.

## 

Pornim de la faptul că (reiese din definiții, din diagramă). Rămâne să demnstrăm că incluziunea e strictă.

Considerăm mulțimea {0, 1}\* (toate șirurile de lungime finită formate din 0 și 1) și le ordonăm mai întâi după lungime și, în caz de lungime egală, lexicografic:

0, 1, 00, 01, 10, 11, 100 etc.

Notăm cu numărul de ordine al lui x în mulțimea ordonată. (= 6).

#(M) = codificarea mașinii M peste {0, 1, 2, (, ), L, R} (e explicată în cursul cu mașina universală).

= numărul de ordine al mașinii M în enumerarea tuturor cuvintelor peste {0, 1, 2, (, ), L, R}.

### 

Considerăm limbajul Ld.

**Ld = {w {0, 1}\* | w L(M) unde = } - exemplu de limbaj NRE.**

Presupunem că Ld este recursiv enumerabil. Atunci, există o mașină Turing M care să îl accepte.

Alegem w cu **=** .

* M acceptă w => w este acceptat de M cu **=** => wLd = L(M)
* M respinge w => w nu este acceptat de M cu **=** => wLd = L(M)

Deci, avem contradicție.

=> Nu există o mașină M care să accepte Ld => Ld .

### 

Considerăm limbajul Lu.

**Lu = {w#(M) |w {0, 1}\* și w L(M) } - exemplu de limbaj RE.**

(w concatenat cu #(M))

Presupunem că Lu este recursiv. Atunci, există o mașină Turing Mu care se oprește pe fiecare intrare cu L(Mu) = Lu.

Atunci, putem construi următoarea mașină M:

* Input: w din {0, 1}\*
* Găsește
* Găsește M cu **=**
* Simulează Mu pe intrarea w#(M):
  + Mu acceptă => M respinge
  + Mu respinge => M acceptă

Din moment ce Mu se oprește pe fiecare intrare, și M se oprește pe fiecare intrare, deci L(M) este recursiv.

Dar L(M) = {w {0, 1}\* | w L(M) unde = } = Ld, iar Ld => contradicție.

Deci, nu putem construi mașina Mu, iar Lu .