

# Gramatici independente de context (GIC) (CFG – context free grammars)

derivari, arbori de derivare,  
tipuri de gramatici independente de context,  
gramatici echivalente - constructii

# Ne reamintim: Gramatica

O gramatica este un cvadruplu  $\mathbf{G} = (\mathbf{N}, \Sigma, P, S)$

- $\mathbf{N}$  este un alfabet de simboluri **neterminale**
- $\Sigma$  este un alfabet de simboluri **terminale**
- $\mathbf{N} \cap \Sigma = \emptyset$
- $P \subseteq (\mathbf{N} \cup \Sigma)^* \mathbf{N} (\mathbf{N} \cup \Sigma)^* \times (\mathbf{N} \cup \Sigma)^*$ 
  - $\mathbf{P}$  multime finită (multimea regulilor de productie)
  - $S \in \mathbf{N}$  (simbolul de start - simbolul initial)

Notatie:

$(\alpha, \beta) \in P$  se noteaza:  $\alpha \rightarrow \beta$   
( $\alpha$  se înlocuieste cu  $\beta$ )

# Ne reamintim: clasificarea Chomsky

- Gramatici de tip 0:  
nici o restrictie (*suplimentara*) referitoare la forma regulilor de productie
- Gramaticile de tip 1  
(gramatici monotone)
  - $\forall \alpha \rightarrow \beta \in P: |\alpha| \leq |\beta|$
  - caz special:  $S \rightarrow \epsilon$  poate  $\in P$ . In acest caz S nu apare în membrul drept al nici unei reguli de productie.

## • Gramatici independente de context:

**reg. productie sunt de forma  $A \rightarrow \alpha$ ,  $A \in N$ ,  $\alpha \in (N \cup \Sigma)^*$**   
**(gramatici de tip 2)**

- Gramaticile de tip 3:  
reg. prod. sunt de forma
  - $A \rightarrow aB$
  - $A \rightarrow b$unde  $A, B \in N$  si  $a, b \in \Sigma$   
caz special:  $S \rightarrow \epsilon$  poate  $\in P$ . In acest caz S nu apare în membrul drept al nici unei reguli de productie.

# derivari de stanga/ dreapta

- *derivare de stânga*  $\Rightarrow_{st}$   
o derivare directă în care se înlocuiește cel mai din stânga neterminál
  - *derivare de dreapta*  $\Rightarrow_{dr}$   
o derivare directă în care se înlocuiește cel mai din dreapta neterminál

# Analiza sintactica

- *analiză sintactică* pt. cuvantul w  
succesiunea de derivări directe:
  - $S \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \dots \Rightarrow \alpha_n = w$   
altfel spus: reprezintă o derivare pentru cuvântul w
- *analiză sintactică descendentală*  
dacă această succesiune de derivări directe se obtine pornind de la S și terminând cu w
- *analiză sintactică ascendentă*  
dacă această succesiune de derivări directe se obtine pornind de la w și terminând cu S

# Arbore de derivare

- Fie  $G = (N, \Sigma, P, S)$  o gramatică independentă de context. Numim *arbore de derivare* sau *arbore de analiză sintactică* un arbore cu radacina, ordonat, cu urmatoarele proprietati:
  1. Orice nod interior - o eticheta din  $N$ ;
  2. Orice nod frunza - o *etichetă* din  $\Sigma \cup \{\varepsilon\}$
  3. Eticheta rădăcinii este  $S$ ;
  4. Dacă un nod are eticheta  $A$  iar nodurile succesoare acestuia, în ordine de la stânga la dreapta sunt etichetate cu  $X_1, X_2, \dots, X_n$  atunci  $A \rightarrow X_1 X_2 \dots X_n$  trebuie să fie o producție din  $P$ .

# Arbore de derivare

- **frontiera (frontul)**: nodurile terminale, în ordine de la stânga la dreapta
- etichetele lor formează o secvență peste  $\Sigma^*$
- obs: denumirea de frontiera (front) se folosește și pentru a denumi succesiunea etichetelor nodurilor terminale

## Teoremă.

Fie  $G = (N, \Sigma, P, S)$  o gramatică independentă de context. Un cuvânt  $w$  peste alfabetul  $\Sigma$ , deci din  $\Sigma^*$ , aparține limbajului generat de  $G$ , adică  $w \in L(G)$ , dacă și numai dacă  $w$  este frontul unui arbore de analiză sintactică.

# Gramatica ambigua

O gramatică  $\mathbf{G} = (\mathbf{N}, \Sigma, P, S)$  independentă de context este *ambiguă*

dacă și numai dacă există cel puțin un cuvânt  $w$  care admite doi arbori de derivare distincti; în caz contrar gramatica este *neambiguă*.

- $\Leftrightarrow \exists 2$  analize sintactice care folosesc numai derivari de stanga, diferite
- $\Leftrightarrow \exists 2$  analize sintactice care folosesc numai derivari de dreapta, diferite

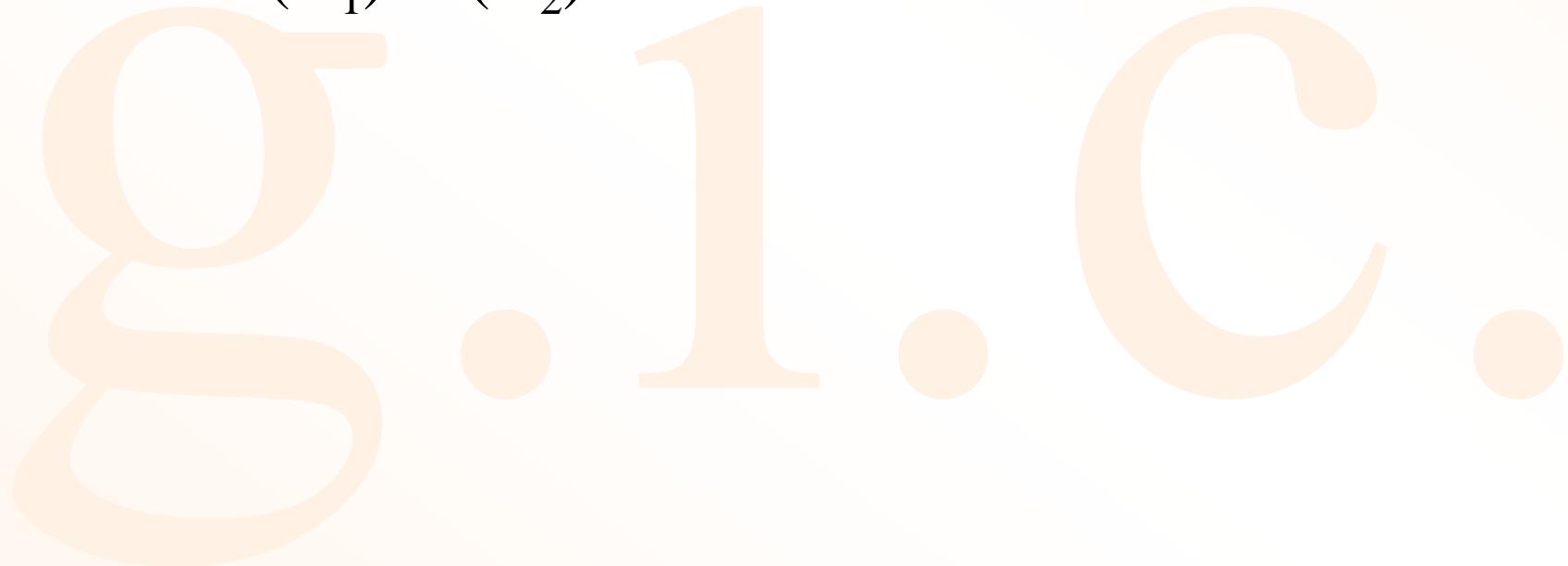
# Descrieri echivalente. Forme normale

**Ne reamintim :**

O gramatica  $G_1$

este echivalenta cu gramatica  $G_2$

daca  $L(G_1) = L(G_2)$



# $\epsilon$ -productii si gram. $\epsilon$ -independente

- **$\epsilon$ -productie** : o productie de forma  $A \rightarrow \epsilon$
- Gramatica  $G = (N, \Sigma, P, S)$  este  **$\epsilon$ -independentă** daca:
  - a) dacă  $\epsilon \notin L(G)$  atunci  $G$  nu are  $\epsilon$ -productii
  - b) dacă  $\epsilon \in L(G)$  atunci avem o singură productie  $S \rightarrow \epsilon$  iar celelalte productii nu-l contin în membrul drept pe  $S$
- Teorema
$$\forall G = (N, \Sigma, P, S)$$
$$\exists G' = (N', \Sigma', P', S)$$
 echivalentă,  $\epsilon$ -independentă

# Redenumiri. Cicluri.

- *redenumire*: reg.prod. de forma  $A \rightarrow B$
- *Gramatica fără redenumiri*: fara r.p. de redenumire

Teorema:

$\forall G = (N, \Sigma, P, S) \exists G' = (N', \Sigma', P', S)$  echivalentă fără redenumiri

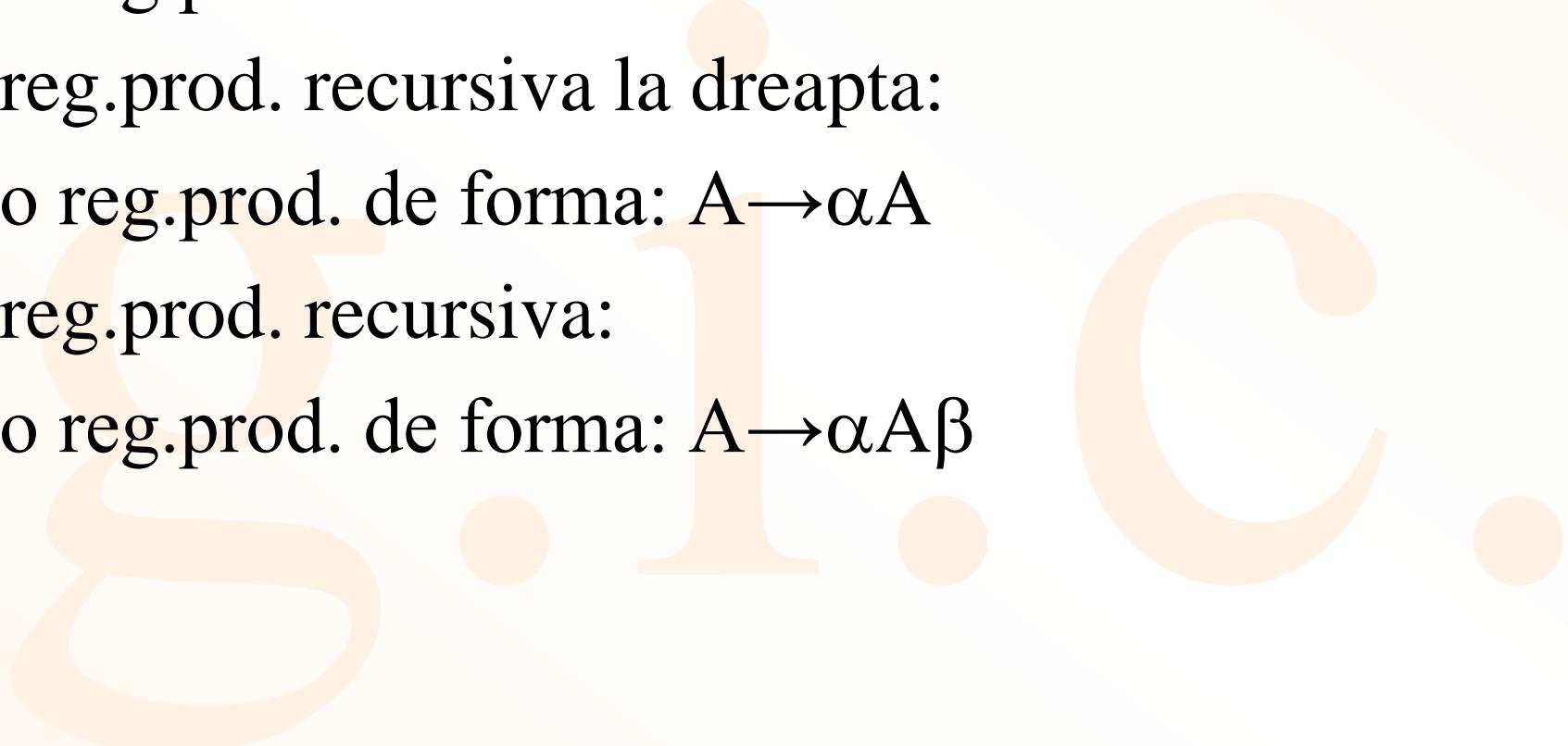
- *ciclu*: o \* derivare de forma  $A \Rightarrow^* B$
- *Gramatica fără cicluri*: nu se pot obtine cicluri (la derivare)

Teorema:

$\forall G = (N, \Sigma, P, S) \exists G' = (N', \Sigma', P', S)$  echivalentă fără cicluri

# Recursivitate

- reg.prod. recursiva la stanga:
  - reg.prod. de forma:  $A \rightarrow A\alpha$
- reg.prod. recursiva la dreapta:
  - reg.prod. de forma:  $A \rightarrow \alpha A$
- reg.prod. recursiva:
  - reg.prod. de forma:  $A \rightarrow \alpha A \beta$



# Recursivitate

- **neterminal recursiv la stanga:**  
 $A \in N$  daca  $\exists$  o derivare de forma:  $A =^+> A\alpha$
- **neterminal recursiv la dreapta:**  
 $A \in N$  daca  $\exists$  o derivare de forma:  $A =^+> \alpha A$
- **neterminal recursiv:**  
 $A \in N$  daca  $\exists$  o derivare de forma:  $A =^+> \alpha A \beta$
- **gramatica recursiva la stanga:**  
are cel putin un neterminal recursiv la stanga
- **gramatica recursiva la dreapta:** ...

# Forma normală Chomsky

O gramatică independentă de context  $G = (N, \Sigma, P, S)$  este în *forma normală Chomsky (FNC)*

dacă orice regula de producție din  $P$  este de una din formele:

a)  $A \rightarrow BC$        $A, B, C \in N;$

b)  $A \rightarrow a$        $a \in \Sigma, A \in N;$

Si un caz special:  $S \rightarrow \epsilon$  poate  $\in P$ . În acest caz  $S$  nu apare în membrul drept al nici unei reguli de producție.

**Teoremă.** Oricare ar fi  $G = (N, S, P, S)$  o gramatică independentă de context, întotdeauna există o gramatică în forma normală Chomsky  $G'$ , astfel încât  $L(G) = L(G')$ .

# Forma normală Greibach

O gramatică  $G = (N, \Sigma, P, S)$

este în *forma normală Greibach (FNG)*

dacă  $P$  are productii numai de forma:

$$A \rightarrow a\alpha, \quad A \in N, a \in \Sigma, \alpha \in N^*;$$

Si un caz special:  $S \rightarrow \varepsilon$  poate  $\in P$ . In acest caz  $S$  nu apare în membrul drept al nici unei reguli de productie.

**Teorema.** Oricare ar fi  $G = (N, \Sigma, P, S)$  o gramatică independentă de context, întotdeauna există o gramatică în forma normală Greibach, astfel încât  $L(G) = L(G')$ .

# Simplificarea GIC

- *simbol neproductiv*

Un simbol  $A \in N$  este *neproductiv* dacă nu există nici o derivare de forma  $A =^* > x \quad (x \in \Sigma^*)$

- În caz contrar  $A$  este *simbol productiv*

- Teorema

$\forall G = (N, \Sigma, P, S) \exists G' = (N', \Sigma', P', S)$  echivalentă, fără simboluri neproductive

# Simplificarea GIC

(transformari echivalente)

- *simbol inaccessible*

Un simbol  $X \in N \cup \Sigma$  este *simbol inaccessible* dacă nu există nici o  $*$  derivare:  $S \Rightarrow^* \alpha X \beta \quad (\alpha, \beta \in (N \cup \Sigma)^*)$

- În caz contrar simbolul este *accessible*
- Teorema

$\forall G = (N, \Sigma, P, S) \exists G' = (N', \Sigma', P', S)$  echivalentă, fără simboluri inaccessible

# Determinarea simbolurilor productive

≈ algoritm AF determ. stari productive

$$\begin{aligned} A &\rightarrow BC \\ B &\rightarrow bB \\ C &\rightarrow c \end{aligned}$$

1.  $i := 0 ; V_0 := \Phi$

2. Repeta

$$V_{i+1} := V_i \cup \{A \in N \mid \exists A \rightarrow \alpha \in P, \alpha \in (V_i \cup \Sigma)^*\}$$

$$i := i + 1$$

pana cand  $V_i = V_{i-1}$

$\{V_i - multimea simbolurilor productive\}$

# Determinarea simbolurilor accesibile

≈ algoritm AF determ. stari accesibile

1.  $i := 0 ; V_0 := \{S\}$

2. Repeta

$$V_{i+1} := V_i \cup \{ B \in N \mid \exists A \in V_i, \alpha, \beta \in (N \cup \Sigma)^* \text{ a.i. } A \rightarrow \alpha B \beta \in P \}$$

$i := i + 1$

**pana cand**  $V_i = V_{i-1}$

$\{V_i - multimea simbolurilor neterminale accesibile\}$

\* analog pentru simboluri terminale accesibile

# Simplificarea GIC

- Un simbol este **neutilizabil** dacă el este fie inaccesibil, fie neproductiv
- Teorema

$\forall G = (N, \Sigma, P, S) \exists G' = (N', \Sigma', P', S)$  echivalentă fără simboluri neutilizabile



## Observatii:

Fie  $G = (N, \Sigma, P, S)$  o gramatica independenta de context:

- Fie un simbol neterminal  $A$  al gramaticii  $G$ .

Daca nu exista o regula de productie  $A \rightarrow \alpha$  in  $P$   
atunci  $A$  este neproductiv

- Fie un simbol terminal  $a$  al gramaticii  $G$ .

Daca nu exista o regula de productie de forma  
 $B \rightarrow \alpha a \beta$  in  $P$   
atunci  $a$  este inaccesibil

# *Eliminarea $\varepsilon$ -productiilor*

1. Construim multimea  $N_\varepsilon$  care are ca elemente acele neterminale care prin derivare conduc la  $\varepsilon$  adică :

- $N_\varepsilon = \{A \mid A \in N, A \Rightarrow^* \varepsilon\}$   
alg.  $\approx$  determinarea simb. productive

2. Determinam noile reguli de productie

- astfel incat productiile de forma  $A \rightarrow \varepsilon$  se elimina
- dar, daca  $\varepsilon \in L(G)$ , atunci  $\exists S \rightarrow \varepsilon$  si  $S$  nu apare în membrul drept al nici unei productii

# Determinarea lui $N_\varepsilon$

1.  $i := 0 ;$

$$V_0 := \{A \in N \mid \exists A \rightarrow \varepsilon \in P\}$$

2. **Repetă**

$$V_{i+1} := V_i \cup \{A \in N \mid \exists A \rightarrow \alpha \in P, \alpha \in (V_i)^*\}$$

$$i := i + 1$$

**pană cand**  $V_i = V_{i-1}$

# determinam noile reguli de productie

- productiile de forma  $A \rightarrow \varepsilon$  se elimina
- celelalte r.p. se rescriu astfel incat sa “suplineasca” eliminarea  $\varepsilon$ -productiilor astfel:

Fie r.p.  $A \rightarrow \alpha_0 B_1 \alpha_1 B_2 \alpha_2 \dots B_k \alpha_k$

unde:  $B_i \in N_\varepsilon$

$\alpha_j$  nu contine simb. din  $N_\varepsilon$

Se inlocuieste cu:

$A \rightarrow \alpha_0 X_1 \alpha_1 X_2 \alpha_2 \dots X_k \alpha_k$

unde  $X_i = \begin{cases} B_i & \text{este unul dintre } B_i \text{ sau } \varepsilon \\ \varepsilon & \text{(se fac toate inlocuirile posibile)} \end{cases}$

**Ce lipseste ???**

# determinam noile reguli de productie

- continuare

Dacă  $\varepsilon \in L(G)$  trebuie să avem o  $\varepsilon$ -productie

“atunci avem productia  $S \rightarrow \varepsilon$  și  $S$  nu apare în membrul drept al nici unei productii”  
(gram.  $\varepsilon$ -independenta)

- adaugam un nou simbol de start  $S'$  și productiile  $S' \rightarrow \varepsilon \mid S$

# Redenumiri. Cicluri.

- *redenumire*: reg.prod. de forma  $A \rightarrow B$
- *Gramatica fără redenumiri*: fara r.p. de redenumire

Teorema:

$\forall G = (N, \Sigma, P, S) \exists G' = (N', \Sigma', P', S)$  echivalentă fără redenumiri

- *ciclu*: o \* derivare de forma  $A \Rightarrow^* B$
- *Gramatica fără cicluri*: nu se pot obtine cicluri (la derivare)

Teorema:

$\forall G = (N, \Sigma, P, S) \exists G' = (N', \Sigma', P', S)$  echivalentă fără cicluri

# Eliminarea redenumirilor

**PP. G –  $\varepsilon$ -independenta** (daca nu , luam gr.echiv.  $\varepsilon$ -ind.)

Pentru fiecare  $A \in N$

se elimina redenumirile de forma  $A \rightarrow B$  ( $\forall B \in N$ )

- construieste multimile  $N_A = \{B \mid A =^* > B\}$ ;  
 $(\approx$  det. simb. accesibile)
- determinam noile reguli de productie

# Construieste $N_A = \{D \mid A =^* > D\}$

1.  $i := 0$  ;

$V_0 := \{A\}$

2. Repeta

$V_{i+1} := V_i \cup \{C \mid (B \rightarrow C) \in P, B \in V_i\}$

$i := i + 1$

pana cand  $V_i = V_{i-1}$

$N_A := V_i$

# determinam noile reguli de productie

A ∈ N:

- pentru fiecare  $A \rightarrow \alpha \in P$  execută
  - daca  $\alpha$  e format dintr-un singur neterminatunci il excludem din mult. noilor reg.prod  
altfel adaugam:  $B \rightarrow \alpha , \forall B \in N$  a.i.  $A \in N_B$
  - sf.daca
  - sf.pentru

# Eliminarea redenumirilor

## Exercitiu:

$$E \rightarrow E + T$$
$$E \rightarrow T$$
$$T \rightarrow T * F$$
$$T \rightarrow F$$
$$F \rightarrow (E)$$
$$F \rightarrow a$$

1.C

# Gramatica fara cicluri

Teorema:

$\forall G = (N, \Sigma, P, S) \exists G' = (N', \Sigma', P', S)$  echivalentă fără cicluri

Daca  $G - \varepsilon$ -independenta si fara redenumiri atunci este fara cicluri

# Gramatica propre:

este o gramatica

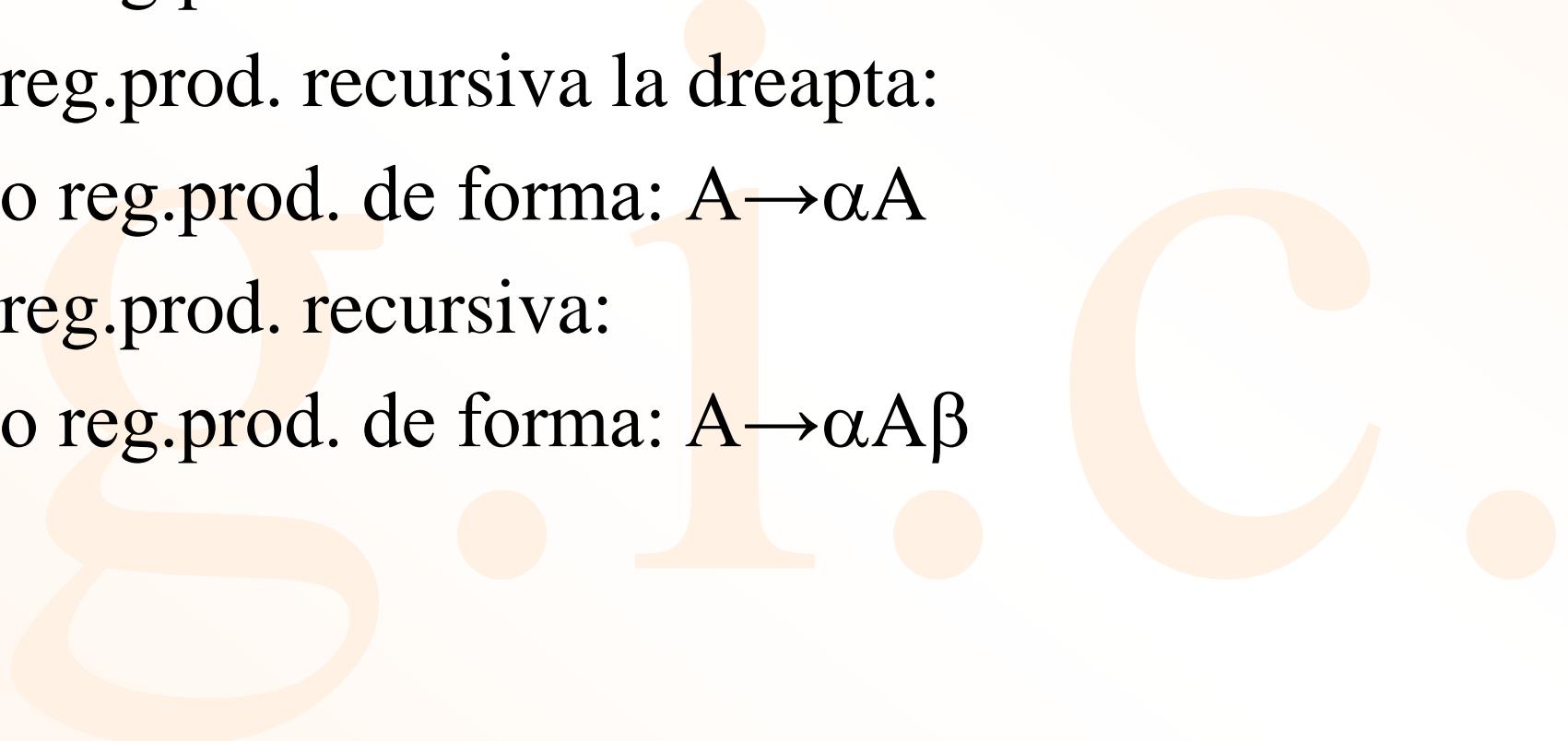
- fara simb. neutilizabile
- **$\epsilon$ -independenta**
- fara cicluri

Teorema:

$\forall G = (N, \Sigma, P, S) \exists G' = (N', \Sigma', P', S)$  propri echiv.

# Recursivitate

- reg.prod. recursiva la stanga:
  - reg.prod. de forma:  $A \rightarrow A\alpha$
- reg.prod. recursiva la dreapta:
  - reg.prod. de forma:  $A \rightarrow \alpha A$
- reg.prod. recursiva:
  - reg.prod. de forma:  $A \rightarrow \alpha A \beta$



# Reg. prod. recursive la stanga

- reg.prod. recursiva la stanga:  $A \rightarrow A\alpha$

Teorema:

$\forall G = (N, \Sigma, P, S) \exists G' = (N', \Sigma', P', S)$  echivalentă fără reg.prod. recursive la stanga

- **PP. G – gr. propre**  
(daca nu este, det. gr. propre echiv. si lucram cu ea)
- Obs.: vom obtine tot o gramatica propre

# Eliminarea r.p. recursive la stanga

pentru fiecare  $A \in N$ : reg.prod.cu m.s. A

- grupam r.p. in recursive la stng. si nerec. la stanga

$$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \dots \mid A\alpha_r \quad (\text{r.p. recursive})$$

$$A \rightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_s \quad (\text{r.p. ne-recursive})$$

- r.p. se transforma astfel:

$$A \rightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_s \mid \beta_1 A' \mid \beta_2 A' \mid \dots \mid \beta_s A'$$

$$A' \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_r \mid \alpha_1 A' \mid \alpha_2 A' \mid \dots \mid \alpha_r A'$$

(a fost introdus un net. nou:  $A'$ )

# Eliminarea r.p. recursive la stanga

## Observatii:

- Recursivitatea nu se poate elimina.
- Recursivitatea la stanga a fost transformata în recursivitate la dreapta.

Exercitiu

Eliminati recursivitatea la stanga:

$S \rightarrow Sa$

$S \rightarrow a$

# Recursivitate

- **neterminal recursiv la stanga:**  
 $A \in N$  daca  $\exists$  o derivare de forma:  $A =^+> A\alpha$
- **neterminal recursiv la dreapta:**  
 $A \in N$  daca  $\exists$  o derivare de forma:  $A =^+> \alpha A$
- **neterminal recursiv:**  
 $A \in N$  daca  $\exists$  o derivare de forma:  $A =^+> \alpha A \beta$
- **gramatica recursiva la stanga:**  
are cel putin un neterminal recursiv la stanga
- **gramatica recursiva la dreapta:** ...

# Eliminarea recurs. la stg. a neterm.

Teorema:

$\forall G = (N, \Sigma, P, S) \exists G' = (N', \Sigma', P', S)$  echivalentă fără neterminale recursive la stanga

- PP.  $G$  – gr. propriie  
(daca nu este, det. gr. propriie echiv. si lucram cu ea)
- impunem o ordine asupra neterminalelor  
 $N = \{A_1, A_2, \dots, A_n\}$   
si apoi modific r.p. a.i. sa nu existe  $A_i \rightarrow A_j \alpha$  cu  $j <= i$   
de aici  $\Rightarrow$  nu va exista recursivitate la stanga

# Eliminarea recurs. la stg. a neterm.

**pentru**  $A_i$  de la  $A_1$  la  $A_n$  **executa**

//se elimina r.p. de forma  $A_i \rightarrow A_j \alpha$  cu  $j <= i$  astfel:

\***repeta**

**pentru**  $j := 1, i - 1$  **executa**

\*  $A_i \rightarrow A_j \alpha$  ( $j < i$ ) se inlocuieste cu:  $A_i \rightarrow \beta \alpha$   
cu toti  $\beta$  cu proprietatea  $A_j \rightarrow \beta \in P_{inloc}$

**sf.pentru**

\* se elimina r.p. de forma  $A_i \rightarrow A_i \alpha$

(se inlocuiesc cf.alg. de elim.r.p.rec.stg)

\***pana cand** toate r.p. cu  $A_i$  in m.s. respecta:  $\nexists j < i : A_i \rightarrow A_j \alpha$

**sf.pentru**

# Eliminarea recurs. la stg. a neterm.

Exercitii:

(1)

$$A \rightarrow BC \mid a$$

$$B \rightarrow CA \mid b$$

$$C \rightarrow AB \mid c$$

(2)

$$A \rightarrow a \mid aB$$

$$B \rightarrow AC \mid b$$

$$C \rightarrow BA \mid c$$