

Limbaje Formale, Automate și Compilatoare

Curs 5

2014-15

Curs 5

- 1 Gramatici și limbaje independente de context
- 2 Forma redusă pentru gramatici independente de context
- 3 Eliminarea regulilor de ștergere și a redenumirilor
- 4 Forma normală Chomsky
- 5 Problema recunoașterii: algoritmul Cocke Younger Kasami
- 6 Eliminarea recursivității stângi în gramatici de tip 2

Curs 5

- 1 Gramatici și limbaje independente de context
- 2 Forma redusă pentru gramatici independente de context
- 3 Eliminarea regulilor de ștergere și a redenumirilor
- 4 Forma normală Chomsky
- 5 Problema recunoașterii: algoritmul Cocke Younger Kasami
- 6 Eliminarea recursivității stângi în gramatici de tip 2

Gramatici independente de context

- Gramatici de tip 2 (independente de context): $G = (N, T, S, P)$
 - N și T sunt mulțimi nevide, finite, disjuncte de neterminali (variabile), respectiv terminali
 - $S \in N$ este simbolul de start
 - $P = \{x \rightarrow u \mid x \in N, u \in (N \cup T)^*\}$ este mulțimea regulilor (producțiilor).
- Un limbaj L este de tip 2 (independent de context: $L \in \mathcal{L}_2$) dacă există o gramatică G de tip 2 astfel încât $L(G) = L$

Derivări extrem stângi/drepte

Fie $G = (N, T, S, P)$ și $w \in L(G)$

- **derivare extrem stângă pentru w :** derivarea în care, la orice pas se înlocuiește cel mai din stânga neterminal din cuvântul obținut
- **derivare extrem dreaptă pentru w :** derivarea în care, la orice pas se înlocuiește cel mai din dreapta neterminal din cuvântul obținut

Exemplu

$G = (\{E\}, \{a, b, +, *\}, \{\}, E, P)$ unde:

$$P : E \rightarrow E + E \mid E * E \mid (E) \mid a \mid b$$

Fie $a + (b * a)$

- Derivare extrem stângă:

$$E \Rightarrow E + E \Rightarrow a + E \Rightarrow a + (E) \Rightarrow a + (E * E) \Rightarrow a + (b * E) \Rightarrow a + (b * a)$$

- Derivare extrem dreaptă:

$$E \Rightarrow E + E \Rightarrow E + (E) \Rightarrow E + (E * E) \Rightarrow E + (E * a) \Rightarrow E + (b * a) \Rightarrow a + (b * a)$$

- Există derivări care nu sunt nici extrem drepte nici extrem stângi!

Arbori sintactici

Definiție 1

Un *arbore sintactic* (*arbore de derivare*, *arbore de parsare*) în gramatica G este un arbore ordonat, etichetat, cu următoarele proprietăți:

- rădăcina arborelui este etichetată cu S ;
- fiecare frunză este etichetată cu un simbol din T sau cu ϵ ;
- fiecare nod interior este etichetat cu un neterminal;
- dacă A etichetează un nod interior care are n succesori etichetați de la stânga la dreapta respectiv cu X_1, X_2, \dots, X_n , atunci $A \rightarrow X_1 X_2 \dots X_n$ este o regulă.

Dacă A are un succesori etichetat cu ϵ , atunci acesta este singurul său succesori.

Arbori sintactici

Definiție 2

- *Frontiera unui arbore de derivare* este cuvântul $w = a_1 a_2 \dots a_n$ unde a_i , $1 \leq i \leq n$ sunt etichetele nodurilor frunză în ordinea de la stânga la dreapta.
- *Arbore de derivare pentru un cuvânt w* : arbore de derivare cu frontiera w .
- Un *X-arbore de derivare* este un subarbore al unui arbore de derivare care are eticheta rădăcinii X . Un arbore de derivare este un *S-arbore de derivare*.

Arbori sintactici-exemplu

$G = (\{E\}, \{a, b, +, *\}, \{\}, E, P)$ unde:

$P : E \rightarrow E + E | E * E | (E) | a | b$

$a + (b * a)$

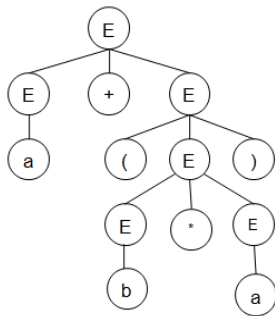
- Derivare extrem stângă:

$$\begin{aligned} E &\Rightarrow E + E \Rightarrow a + E \Rightarrow a + (E) \Rightarrow \\ &a + (E * E) \Rightarrow a + (b * E) \Rightarrow a + (b * a) \end{aligned}$$

- Derivare extrem dreaptă:

$$\begin{aligned} E &\Rightarrow E + E \Rightarrow E + (E) \Rightarrow E + (E * E) \Rightarrow \\ &E + (E * a) \Rightarrow E + (b * a) \Rightarrow a + (b * a) \end{aligned}$$

- Arbore de derivare pentru $a + (b * a)$:



Ambiguitate

Definiție 3

O gramatică G este ambiguă dacă există un cuvânt w în $L(G)$ care are 2 arbori de derivare distincți.

- Echivalent: w are 2 derivări extrem stângi(drepte) distincte.

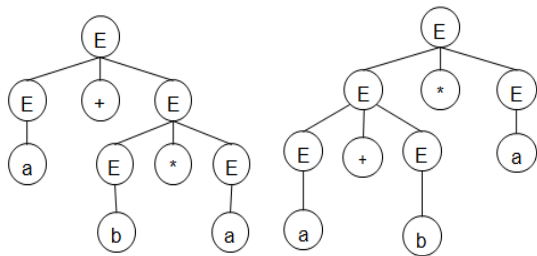
Ambiguitate

Definiție 3

O gramatică G este ambiguă dacă există un cuvânt w în $L(G)$ care are 2 arbori de derivare distincți.

- Echivalent: w are 2 derivări extrem stângi(drepte) distincte.

Gramatica precedentă este ambiguă: cuvântul $a + b * a$ are 2 arbori de derivare:



Ambiguitate

Definiție 3

O gramatică G este ambiguă dacă există un cuvânt w în $L(G)$ care are 2 arbori de derivare distincți.

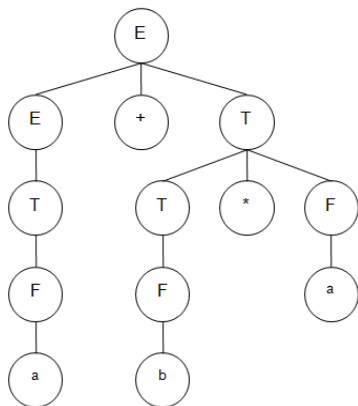
- Echivalent: w are 2 derivări extrem stângi(drepte) distincte.
- Problema ambiguității gramaticilor de tip 2 este nedecidabilă: nu există un algoritm care pentru o gramatică oarecare G să testeze dacă G este sau nu ambiguă

Exemplu: o gramatică echivalentă neambiguă

$G = (\{E, T, F\}, \{a, b, +, *\}, (\{, \}, E, P)$ unde P :

- $E \rightarrow E + T$
- $E \rightarrow T$
- $T \rightarrow T * F$
- $T \rightarrow F$
- $F \rightarrow (E)$
- $F \rightarrow a|b$

- Arbore de derivare pentru $a + b * a$:



Problema recunoașterii în gramatici independente de context

- Problema **recunoașterii** în gramatici independente de context:
Dată o gramatică $G = (N, T, S, P)$ și un cuvânt $w \in T^*$, să se decidă dacă $w \in L(G)$
- Problema **parsării (analizei sintactice)** este problema recunoașterii la care se adaugă: dacă $w \in L(G)$, se cere arborele sintactic (o reprezentare a sa) pentru w .

Curs 5

- 1 Gramatici și limbaje independente de context
- 2 Forma redusă pentru gramatici independente de context**
- 3 Eliminarea regulilor de ștergere și a redenumirilor
- 4 Forma normală Chomsky
- 5 Problema recunoașterii: algoritmul Cocke Younger Kasami
- 6 Eliminarea recursivității stângi în gramatici de tip 2

Simboluri inutile

- Un simbol X din $N \cup T$ este **accesibil** dacă există o derivare de forma $S \Rightarrow^+ \alpha X \beta$
- Un simbol A din N este **productiv** dacă există o derivare de forma $A \Rightarrow^+ w, w \in T^*$
- Un simbol este **inutil** dacă este inaccesibil sau neproductiv

Gramatici în formă redusă

Definiție 4

*O gramatică este în **formă redusă**, dacă nu conține simboluri inutile.*

- Orice limbaj independent de context poate fi generat de o gramatică în formă redusă.

Eliminarea simbolurilor inutile

- Pentru orice gramatică independentă de context G există o gramatică G' de același tip în formă redusă echivalentă cu G .
- Pentru eliminarea simbolurilor inutile:
 - Se determină și apoi se elimină simbolurile neproductive și toate regulile ce conțin măcar unul dintre acestea.
 - Se determină apoi se elimină simbolurile inaccesibile și toate regulile aferente.

Eliminarea simbolurilor neproductive - algoritm

- Intrare: $G = (N, T, S, P)$
- Ieșire: $G' = (N', T, S, P')$, $L(G') = L(G)$, N' conține doar simboluri productive

$N_0 = \emptyset; i = 0;$

do {

$i = i + 1;$

$N_i = N_{i-1} \cup \{A \mid A \rightarrow \alpha \in P, \alpha \in (N_{i-1} \cup T)^*\};$

} while $N_i \neq N_{i-1};$

$N' = N_i;$

$P' = \{A \rightarrow \alpha \in P \mid A \in N', \alpha \in (N' \cup T)^*\};$

- Un simbol A este productiv dacă $A \in N'$
- Consecință: $L(G) \neq \emptyset$ dacă $S \in N'$

Exemplu

$G = (\{S, A, B, C\}, \{a, b, c\}, S, P)$, unde P este:

- $S \rightarrow a|aA|bC$
- $A \rightarrow aAB$
- $B \rightarrow bac$
- $C \rightarrow aSb$

Gramatica G' cu toate simbolurile productive:

$G' = (\{S, B, C\}, \{a, b, c\}, S, P')$, unde P' este:

- $S \rightarrow a|bC$
- $B \rightarrow bac$
- $C \rightarrow aSb$

Eliminarea simbolurilor inaccesibile

- Intrare: $G = (N, T, S, P)$
- Ieșire: $G' = (N', T', S, P')$, $L(G') = L(G)$, N', T' conțin doar simboluri accesibile

$V_0 = \{S\}; i = 0;$

do {

$i = i + 1;$

$V_i = V_{i-1} \cup \{X | X \in N \cup T, \exists A \rightarrow \alpha X \beta \in P, A \in (V_{i-1} \cap N)\};$

} while $V_i \neq V_{i-1};$

$N' = V_i \cap N;$

$T' = V_i \cap T;$

$P' = \{A \rightarrow \alpha \in P | A \in N', \alpha \in (N' \cup T')^*\};$

- X accesibil ddacă $X \in V_i$

Exemplu

$G = (\{S, A, B, C\}, \{a, b, c\}, S, P)$, unde P este:

- $S \rightarrow a|aA|bC$
- $A \rightarrow aAB$
- $B \rightarrow bac$
- $C \rightarrow aSb$
- Eliminarea simbolurilor neproductive duce la:

$$G' = (\{S, B, C\}, \{a, b, c\}, S, \{S \rightarrow a|bC, B \rightarrow bac, C \rightarrow aSb\})$$

- Eliminarea simbolurilor inaccesibile duce la:

$$G' = (\{S, C\}, \{a, b\}, S, \{S \rightarrow a|bC, C \rightarrow aSb\})$$

- Ce se întâmplă dacă se aplică algoritmi în ordinea inversă?

Curs 5

- 1 Gramatici și limbaje independente de context
- 2 Forma redusă pentru gramatici independente de context
- 3 Eliminarea regulilor de ștergere și a redenumirilor**
- 4 Forma normală Chomsky
- 5 Problema recunoașterii: algoritmul Cocke Younger Kasami
- 6 Eliminarea recursivității stângi în gramatici de tip 2

Eliminarea regulilor de ștergere

- Intrare: $G = (N, T, S, P)$
- Ieșire: $G' = (N, T, S, P')$, $L(G') = L(G)$, P' nu conține reguli de ștergere (reguli de forma $A \rightarrow \epsilon$)

$N_0 = \{A | A \in N, A \rightarrow \epsilon \in P\}; i = 0;$

do {

$i = i + 1;$

$N_i = N_{i-1} \cup \{X | X \in N, \exists X \rightarrow \alpha \in P, \alpha \in N_{i-1}^*\};$

} while $N_i \neq N_{i-1};$

$N_\epsilon = N_i;$

Eliminarea regulilor de ștergere

- Intrare: $G = (N, T, S, P)$
- Ieșire: $G' = (N, T, S, P')$, $L(G') = L(G)$, P' nu conține reguli de ștergere (reguli de forma $A \rightarrow \epsilon$)

```

 $N_0 = \{A | A \in N, A \rightarrow \epsilon \in P\}; i = 0;$ 
do {
     $i = i + 1;$ 
     $N_i = N_{i-1} \cup \{X | X \in N, \exists X \rightarrow \alpha \in P, \alpha \in N_{i-1}^*\};$ 
} while  $N_i \neq N_{i-1};$ 
 $N_\epsilon = N_i;$ 

```

Are loc:

- $N_0 \subseteq N_1 \dots \subseteq N_i \subseteq N_{i+1} \subseteq \dots N_\epsilon \subseteq N$
- $A \in N_\epsilon \iff A \Rightarrow^+ \epsilon$

Eliminarea regulilor de ștergere

P' se obține din P astfel:

- în fiecare regulă $A \rightarrow \alpha \in P$ se pun în evidență simbolurile din N_ϵ ce apar în α :

$$\alpha = \alpha_1 X_1 \alpha_2 X_2 \dots \alpha_n X_n \alpha_{n+1}, \quad X_i \in N_\epsilon$$

- se înlocuiește fiecare regulă de acest fel cu mulțimea de reguli de forma

$$A \rightarrow \alpha_1 Y_1 \alpha_2 Y_2 \dots \alpha_n Y_n \alpha_{n+1} \text{ unde } Y_i = X_i \text{ sau } Y_i = \epsilon$$

în toate modurile posibile (2^n)

- se elimină toate regulile de ștergere
- pentru a obține cuvântul nul (dacă S este în N_ϵ) se adaugă S' simbol de start nou și regulile $S' \rightarrow S, S' \rightarrow \epsilon$

Exemplu

$G = (\{S, A, B, C\}, \{a, b, c\}, S, P)$, unde P :

- $S \rightarrow aAbC|BC$
- $A \rightarrow aA|aB$
- $B \rightarrow bB|C$
- $C \rightarrow cC|\epsilon$

$G' = (\{S', S, A, B, C\}, \{a, b, c\}, S', P')$ unde P' :

- $S' \rightarrow S|\epsilon$
- $S \rightarrow aAbC|aAb|B|C$
- $A \rightarrow aA|aB|a$
- $B \rightarrow bB|b|C$
- $C \rightarrow cC|c$

Eliminarea redenumirilor ($A \rightarrow B, A, B \in N$)

- Intrare: $G = (N, T, S, P)$
- Ieșire: $G' = (N, T, S, P'), L(G') = L(G), P'$ nu conține redenumiri

```

for( $A \in N$ ) {
   $N_0 = \{A\}; i = 0;$ 
  do {
     $i = i + 1;$ 
     $N_i = N_{i-1} \cup \{C | C \in N, \exists B \rightarrow C \in P, B \in N_{i-1}\};$ 
  } while  $N_i \neq N_{i-1};$ 
   $N_A = N_i; // N_A = \{X \in N | A \Rightarrow^+ X\}$ 
}
 $P' = \{X \rightarrow \alpha \in P | \alpha \notin N\}$ 
for( $X \rightarrow \alpha_1 | \alpha_2 | \dots | \alpha_n \in P'$ )
  for( $A \in N \ \&\& \ X \in N_A, X \neq A$ )
     $P' = P' \cup \{A \rightarrow \alpha_1 | \alpha_2 | \dots | \alpha_n\}$ 

```

Exemplu

$G = (\{x, y, z\}, \{a, b, c\}, x, P)$, unde P :

- $x \rightarrow y|ax|a$
- $y \rightarrow z|by|b$
- $z \rightarrow cz|c$

$N_x = \{x, y, z\}$, $N_y = \{y, z\}$, $N_z = \{z\}$

Gramatica echivalentă fără redenumiri $G' = (\{x, y, z\}, \{a, b, c\}, x, P')$
unde P' :

- $x \rightarrow ax|a|by|b|cz|c$
- $y \rightarrow by|b|cz|c$
- $z \rightarrow cz|c$

Curs 5

- 1 Gramatici și limbaje independente de context
- 2 Forma redusă pentru gramatici independente de context
- 3 Eliminarea regulilor de ștergere și a redenumirilor
- 4 Forma normală Chomsky**
- 5 Problema recunoașterii: algoritmul Cocke Younger Kasami
- 6 Eliminarea recursivității stângi în gramatici de tip 2

Forma normală Chomsky

Definiție 5

O gramatică este în *formă normală Chomsky* dacă regulile sale au forma:

$A \rightarrow BC, A \rightarrow a$ (și eventual $S \rightarrow \epsilon$) ($A, B, C \in N$ și $a \in T$).

Teorema 1

Orice limbaj independent de context poate fi generat de o gramatică în formă normală Chomsky.

Demonstrație

- Se elimină regulile de ștergere și redenumirile

Demonstrație

- Se elimină regulile de ștergere și redenumirile
- Se elimină regulile care nu sunt în formă normală Chomsky:
Dacă $A \rightarrow x_1 x_2 \dots x_n$, $n > 1$ este o astfel de regulă atunci o înlocuim cu $A \rightarrow Y_1 Y_2 \dots Y_n$ unde:
 - $Y_i = x_i$, dacă $x_i \in N$ (neterminalii rămân la fel)
 - $Y_i = x_a$ dacă $x_i = a \in T$ (x_a este neterminal nou) și se adaugă regula $x_a \rightarrow a$

Demonstrație

- Se elimină regulile de ștergere și redenumirile
- Se elimină regulile care nu sunt în formă normală Chomsky:
Dacă $A \rightarrow x_1 x_2 \dots x_n$, $n > 1$ este o astfel de regulă atunci o înlocuim cu $A \rightarrow Y_1 Y_2 \dots Y_n$ unde:
 - $Y_i = x_i$, dacă $x_i \in N$ (neterminalii rămân la fel)
 - $Y_i = x_a$ dacă $x_i = a \in T$ (x_a este neterminal nou) și se adaugă regula $x_a \rightarrow a$
- O regulă de forma $A \rightarrow Y_1 Y_2 \dots Y_n$, dacă $n > 2$, o înlocuim cu:
 - $A \rightarrow Y_1 Z_1$
 - $Z_1 \rightarrow Y_2 Z_2$
 - $\dots\dots\dots$
 - $Z_{n-3} \rightarrow Y_{n-2} Z_{n-2}$
 - $Z_{n-2} \rightarrow Y_{n-1} Y_n$,
 unde Z_1, Z_2, \dots, Z_{n-2} sunt neterminali noi.

Exemplu

$G = (\{S, A\}, \{a, b, c\}, S, P)$, unde P :

- $S \rightarrow aSb \mid cAc$
- $A \rightarrow cA \mid c$

Gramatica echivalentă în formă normală Chomsky

$G = (\{S, A, x_a, x_b, Z_1, Z_2\}, \{a, b, c\}, S, P')$, unde P' :

- $S \rightarrow x_a Z_1 \mid x_c Z_2$
- $Z_1 \rightarrow Sx_b$
- $Z_2 \rightarrow Ax_c$
- $A \rightarrow x_c A \mid c$
- $x_a \rightarrow a$
- $x_b \rightarrow b$
- $x_c \rightarrow c$

Curs 5

- 1 Gramatici și limbaje independente de context
- 2 Forma redusă pentru gramatici independente de context
- 3 Eliminarea regulilor de ștergere și a redenumirilor
- 4 Forma normală Chomsky
- 5 Problema recunoașterii: algoritmul Cocke Younger Kasami**
- 6 Eliminarea recursivității stângi în gramatici de tip 2

Algoritmul Cocke Younger Kasami (CYK)

- Problema recunoașterii în gramatici în formă normală Chomsky se poate rezolva cu algoritmul CYK în timp $O(n^3)$.
- Dacă $w = a_1 a_2 \dots a_n$ atunci se construiesc mulțimile

$$V_{ij} = \{A \mid A \Rightarrow^+ a_i a_{i+1} \dots a_{i+j-1}\}$$

inductiv pentru $j = 1, \dots, n$

$$w \in L(G) \Leftrightarrow S \in V_{1n}$$

Algoritmul Cocke Younger Kasami

- Pentru $j = 1$:

- $V_{i1} = \{A | A \Rightarrow^+ a_i\} = \{A | \exists A \rightarrow a_i \in P\}$

- Pentru $j > 1$, V_{ij} :

- Dacă $A \Rightarrow^+ a_i a_{i+1} \dots a_{i+j-1}$:

$$A \Rightarrow BC \Rightarrow^+ a_i a_{i+1} \dots a_{i+j-1} \text{ și}$$

$$B \Rightarrow^+ a_i a_{i+1} \dots a_{i+k-1} \quad (B \in V_{ik})$$

$$C \Rightarrow^+ a_{i+k} a_{i+k+1} \dots a_{i+j-1} \quad (C \in V_{i+k, j-k})$$

$$\text{unde } 1 \leq i \leq n+1-j, 1 \leq k \leq j-1$$

- $V_{ij} = \bigcup_{k=1}^{j-1} \{A | A \rightarrow BC \in P, B \in V_{ik}, C \in V_{i+k, j-k}\}$

Algoritmul Cocke Younger Kasami

- Notăție:

$$\{A | A \rightarrow BC \in P, B \in V_{ik}, C \in V_{i+k \ j-k}\} = V_{ik} \circ V_{i+k \ j-k}$$

- Atunci:

pentru $2 \leq j \leq n, 1 \leq i \leq n+1-j$:

$$V_{ij} = \bigcup_{k=1}^{j-1} (V_{ik} \circ V_{i+k \ j-k})$$

Algoritmul Cocke Younger Kasami

- Intrare: $G = (N, T, S, P)$ în formă normală Chomsky, $w = a_1 a_2 \dots a_n$
- Ieșire: $w \in L(G)$?

```

for(i=1; i<=n; i++)
     $V_{i1} = \{A \mid \exists A \rightarrow a_i \in P\};$ 
for(j=2; j<=n; j++)
    for (i=1; i<=n+1-j; i++){
         $V_{ij} = \emptyset;$ 
        for(k=1; k<=j-1; k++)
             $V_{ij} = V_{ij} \cup (V_{ik} \circ V_{i+k \ j-k});$ 
    }
if( $S \in V_{1n}$ )  $w \in L(G)$  else  $w \notin L(G)$ 

```


Exemplu

$G = (\{S, X, Y, Z\}, \{a, b, c\}, S, P)$, unde P :

- $S \rightarrow XY$
- $X \rightarrow XY|a$
- $Y \rightarrow YZ|a|b$
- $Z \rightarrow c$

$w = abc$

Exemplu

$G = (\{S, X, Y, Z\}, \{a, b, c\}, S, P)$, unde P :

- $S \rightarrow XY$
- $X \rightarrow XY|a$
- $Y \rightarrow YZ|a|b$
- $Z \rightarrow c$

$w = abc$

$V_{11} = \{X, Y\}$	$V_{12} = \{S, X\}$	$V_{13} = \{S, X\}$
$V_{21} = \{Y\}$	$V_{22} = \{Y\}$	
$V_{31} = \{Z\}$		

$S \in V_{13} \Leftrightarrow abc \in L(G)$

Curs 5

- 1 Gramatici și limbaje independente de context
- 2 Forma redusă pentru gramatici independente de context
- 3 Eliminarea regulilor de ștergere și a redenumirilor
- 4 Forma normală Chomsky
- 5 Problema recunoașterii: algoritmul Cocke Younger Kasami
- 6 Eliminarea recursivității stângi în gramatici de tip 2

Recursivitate stângă

- Un neterminal A este **recursiv** dacă există măcar o derivare $A \Rightarrow^+ \alpha A \beta$.
- Dacă $\alpha = \epsilon$ atunci A se numește **stâng recursiv**,
dacă $\beta = \epsilon$ atunci A se numește **drept recursiv**.
- Dacă gramatica G conține cel puțin un neterminal stâng recursiv,
 G este **stâng recursivă**
- Un neterminal A este **stâng recursiv imediat** dacă există o regulă $A \rightarrow A\alpha \in P$.

Eliminarea recursivității stânga imediate

- Fie $G = (N, T, S, P)$ în formă redusă
- Fie $A \in N$, A stâng recursiv imediat.
- Fie $A \rightarrow A\alpha_1 | A\alpha_2 \dots | A\alpha_k | \beta_1 | \dots | \beta_n$ toate regulile care încep cu A (β_1, \dots, β_n nu încep cu A). Fie P_A mulțimea acestor reguli.
- Gramatica G' în care A nu este stâng recursiv imediat:
 - $G' = (N \cup \{A'\}, T, S, P')$
 - $P' = P \setminus P_A \cup \{A' \rightarrow \alpha_1 A' | \dots | \alpha_k A' | \epsilon, A \rightarrow \beta_1 A' | \dots | \beta_n A'\}$

Exemplu

$G = (\{S, A\}, \{a, b, c\}, S, P)$ unde P este:

- $S \rightarrow Ac|c$
- $A \rightarrow Aa|Ab|a|b|Sc$

$G' = (S, A, A', a, b, c, S, P')$ unde P' este:

- $S \rightarrow Ac|c$
- $A \rightarrow aA'|bA'|ScA'$
- $A' \rightarrow aA'|bA'|\epsilon$

Observație: A, S stâng recursive