SEMINAR

ARBORI DE DERIVARE ȘI AMBIGUITATE ÎN $\mathcal{L}_2$

Arborii sunt folosiți pt vizualizarea modului în care se produce un cuvânt prin derivare

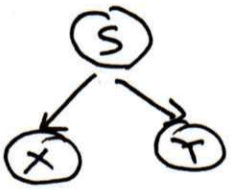
Ex: $G = \begin{cases} S \to XT \\ X \to aXb \mid ab \\ T \to bT \mid b \end{cases} \in \mathcal{G}_2$ $\quad L(G) = \{a^i b^j \mid 1 \le i < j\}$

Considerăm o derivare în G.

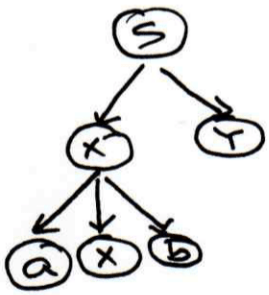
$$S \overset{1)}{\Longrightarrow} XT \overset{2)}{\Longrightarrow} aXbT \overset{3)}{\Longrightarrow} aaXbbT \overset{4)}{\Longrightarrow} aaXbbb \overset{5)}{\Longrightarrow} a^3b^4$$

Construcția arborelui asociat acestei derivări se face prin adăugarea la nodul aflat în stânga regulii aplicate la un pas a descendenților direcți cf. părții drepte.

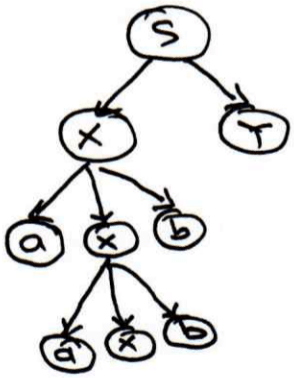
1)



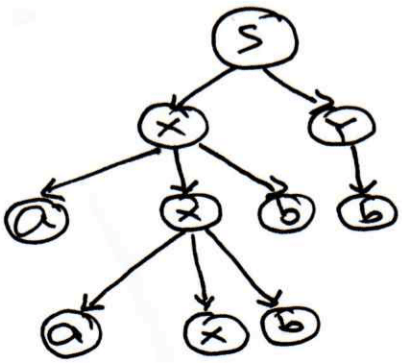
2)



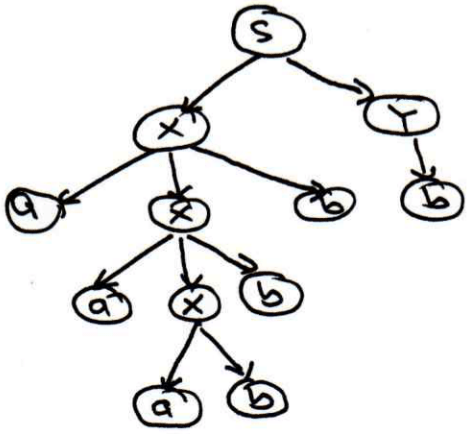
3)



4)



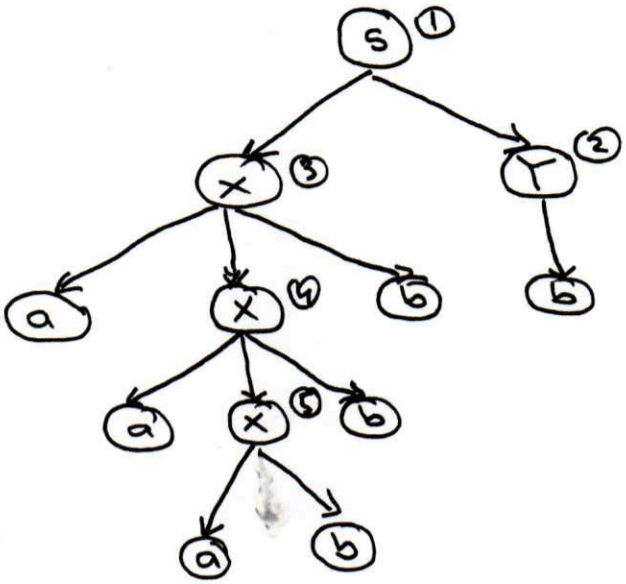
5)



Obs: La fiecare pas frontiera arborelui coincide cu forma propozițională din derivare!

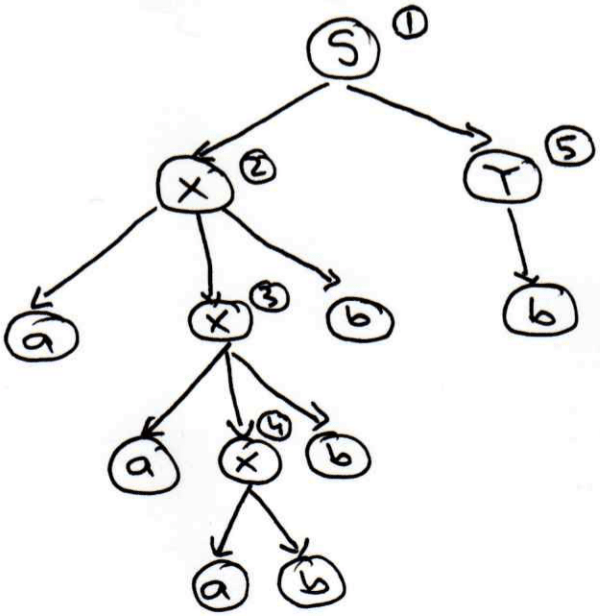
"Citirea" unei derivări de pe un arbore depinde de ordinea parcurgerii nodurilor asociate neterminalelor.

Ex:



$$S \underset{①}{\Rightarrow} XY \underset{②}{\Rightarrow} Xb \underset{③}{\Rightarrow} aXbb \underset{④}{\Rightarrow} aaXbbb \underset{⑤}{\Rightarrow} aaabbbb$$

Unicitatea asocierii arbore $\leftrightarrow$ derivare se face prin derivare extrem stângă (dreaptă)



$$S \underset{①}{\Rightarrow} XY \underset{②}{\Rightarrow} aXbY \underset{③}{\Rightarrow} aaXbbY \underset{④}{\Rightarrow} aaabbbY \underset{⑤}{\Rightarrow}$$

a^3b^4

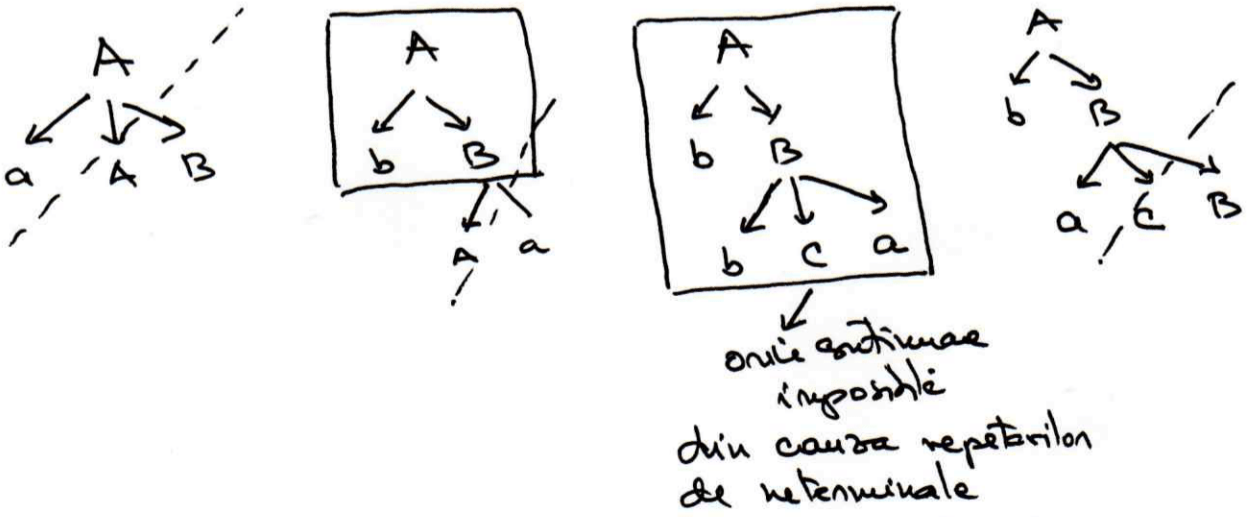
DECIDABILITATE Faptul că limbajul produs de o gramatică independentă de context este vid / nu este vid este DECIDABIL.

⊣

Algoritmul - Se construiesc toți arborii de derivare (ai G) cu proprietatea că neterminalele nu se repetă pe nici o ramură. Dacă există vreun arbore cu frontiera formată numai cu terminale atunci limbajul generat nu este vid, altfel este vid.

Exemplu: Fie $G = \begin{cases} A \to aAB \mid bB \mid aCB \\ B \to Aa \mid bCa \mid Bb \\ C \to AbB \mid cC \mid B \end{cases}$

Se încearcă sistematic regulile în ordinea în care sunt listate în gramatică. Obs: se pot folosi doar regulile ce nu adaugă același terminal pe o ramură!



Concluzie $L(G) =$ lb. vid. pt că singurul arbore construit conține și neterminale

AMBIGUITATE

G Gramatică ambiguă $\overset{def}{\iff}$ $\exists\, p \in L(G)$ ce admite două derivări stângi distincte

L limbaj ambiguu $\overset{def}{\iff}$ orice gramatică ce produce L este ambiguă

Exemplu $G = \begin{cases} X \to ABC \\ A \to aA \mid bA \mid \lambda \\ B \to aa \\ C \to aC \mid bC \mid b \end{cases}$

Evident $L(G) = \{ w \in \{a,b\}^* \mid w$ se termină cu b și conține subcuv. $aa \}$

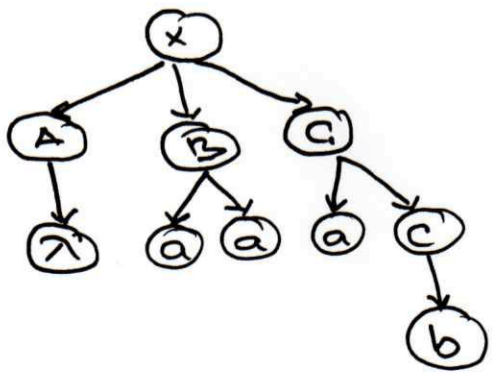
Considerăm $p = aaab \in L(G)$ și două derivări

(1) $X \Rightarrow ABC \Rightarrow BC \Rightarrow aaC \Rightarrow aaaC \Rightarrow aaab$

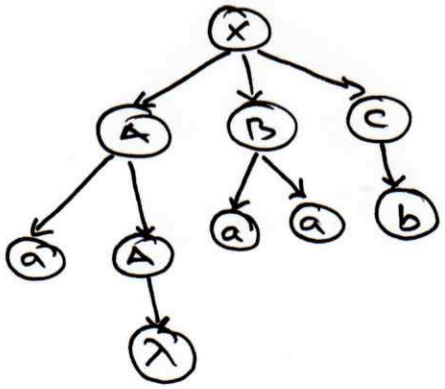
(2) $X \Rightarrow ABC \Rightarrow aABC \Rightarrow aBC \Rightarrow aaaC \Rightarrow aaab$

Arborii de derivare asociați sunt

(1)



(2)

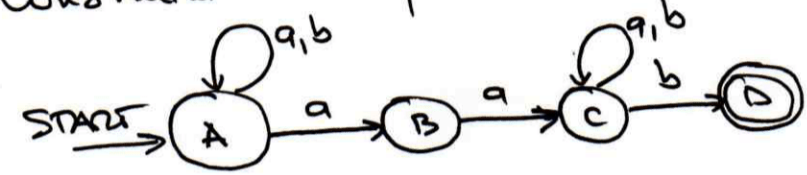


$L = \{ w \in \{a,b\}^* \mid w$ se termină cu b și $aa \in Sub(w) \}$

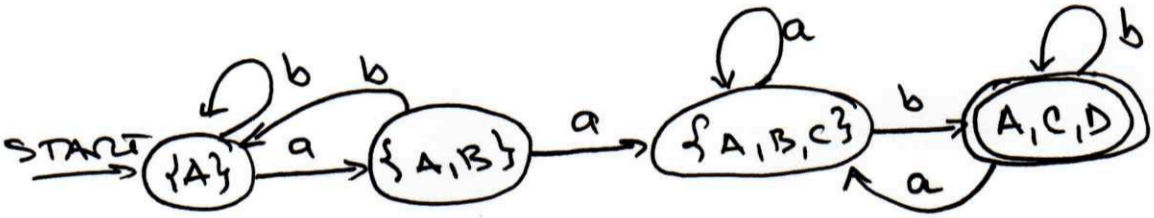
Problemă L ambiguu? Dacă L neambiguu găsiți o gramatică neambiguă G a.î. $L(G) = L$.

Greu de rezolvat fără "teorie". [1] Limbajele de tipul 3 sunt neambigue. - o gramatică neambiguă se obține pornind cu AFD ce recunoaște L!

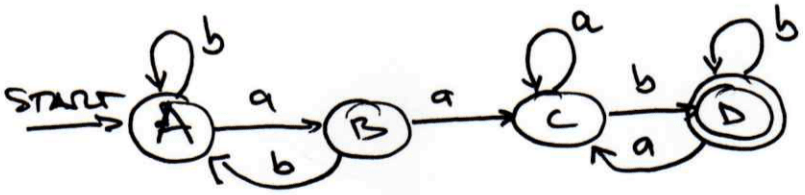
Construim AF pt L. (deci $L \in \mathcal{L}_3$!)



Găsim AFD echivalent



Redenumim stările

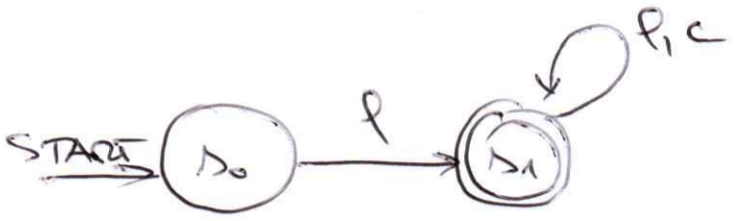


Citim regulile gramaticii neambigue

$$\begin{cases} A \to aB \mid bA \\ B \to aC \mid bA \\ C \to aC \mid bD \mid b \\ D \to aC \mid bD \mid b \end{cases} \in \mathcal{G}_3 \text{ neambiguă (cf. teoremei)}$$

Problema 3

Scrieți un program ce simulează funcționarea următorului ~~AFD~~



unde l - literă, c - cifră.

(Evident $L(AFD) = \{$ identificatori ai lui $C\}$.)

+

Un tabel pt funcția de evoluție

	l	c
S_0	S_1	$\emptyset$
S_1	S_1	S_1

Input: Un cuvânt de analizat

Output: Răspuns YES/NO - cuv. recunoscut / nerecunoscut

APLICAȚIE DIRECTĂ A AFD LA CĂUTARE.

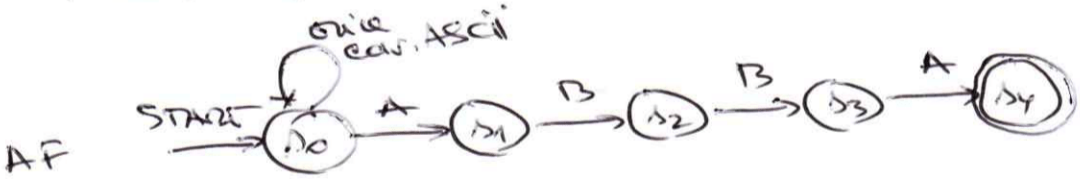
Se consideră un cuvânt fixat. (De ex: ABBA).
Scrieți un program pentru identificarea apariției cuvântului într-un text arbitrar (un fișier).
Se vor număra toate aparițiile.

Rezolvare

Metoda naivă - se mută un șablon (corespunzător cuvântului) cu text și se identifică aparițiile.

Metoda bazată pe AF.

Se consideră AFD ce recunoaște toate șirurile ce se termină cu ABBA. Căutarea se face cu programul ce simulează AFD. Găsirea unui cuvânt presupune apariția stării finale pe traiectorie



⇓

AFD

Calculul stării noi se face prin consultarea unui tablou cu puține coloane. Timp liniar de calcul pt. toate aparițiile. Nu se refac comparații inutile pt. găsirea unei apariții a cuvântului!

Timp de lucru - nr. caractere citite + transformarea AF → AFD

Restabil pt texte lungi (de exemplu căutarea în fișiere obj!)