Gramatici

- Recapitulare, Exemple, Aplicatii
- Algoritmul CYK

Limbaje formale si tehnici de compilare

Limbaj formal

Limbaj regulare:

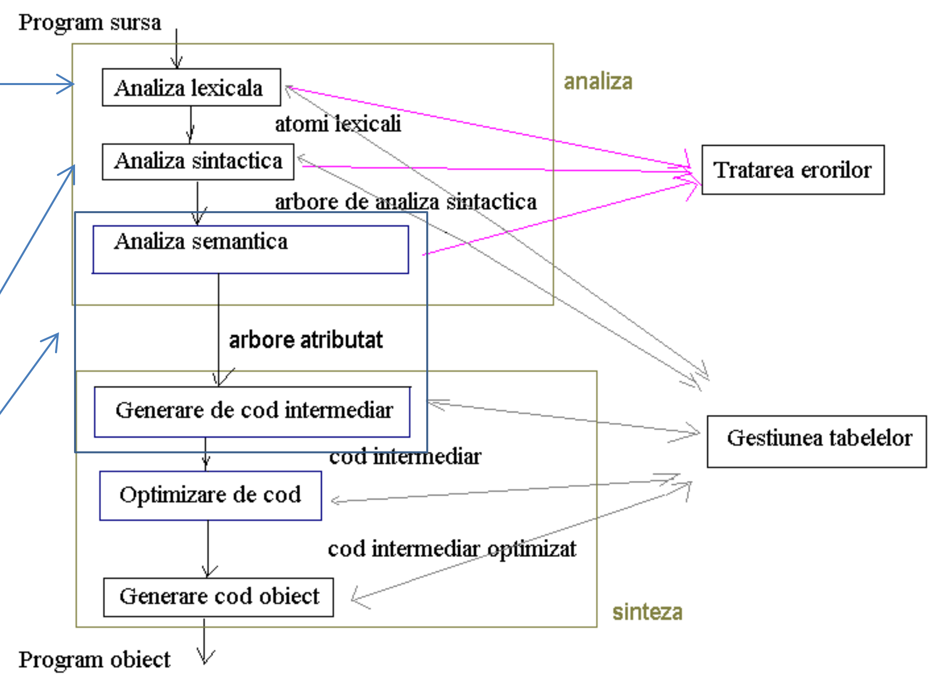
- gramatici regulare,
- automate finite,
- expresii regulare.

Limbaje independente de context:

- Gramatici independente de context, automate push-down
- Gramatici speciale: LL(k), LR(k)

Gramatici de atribute

Structura unui compilator



Gramatica

O gramatica este un cvadruplu **G = (N, Σ , P, S)**

- **N** este un alfabet de simboluri ***neterminale***
- Σ este un alfabet de simboluri ***terminale***
- $\mathbf{N \cap \Sigma = \phi}$
- $\mathbf{P \subseteq (N \cup \Sigma)^* N (N \cup \Sigma)^* \times (N \cup \Sigma)^*}$

  P multime finitã **(**multimea regulilor de productie**)**
- $\mathbf{S \in N}$ (simbolul de start - simbolul initial)

Notatie:

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

(α se înlocuieste cu β)

- Gramatici de tip **0**

  nici o restrictie *(suplimentara)* referitoare la forma regulilor de productie
- Gramaticile de tip **1**

  dependente de context ⬄ ***gramatici monotone***

  (monotonic, non-contracting)
- Gramaticile de tip **2**

  gramatici independente de context
- Gramaticile de tip **3**

  gramatici regulare

Clasificarea Chomsky

Ierarhia Chomsky

~ 1959-1963

Fie

- $\mathcal{L}0$ - multimea limbajelor generate de gram. de tip 0
- $\mathcal{L}1$ - multimea limbajelor generate de gram. de tip 1
- $\mathcal{L}2$ - multimea limbajelor generate de gram. de tip 2
- $\mathcal{L}3$ - multimea limbajelor generate de gram. de tip 3

Are loc:

$$\mathcal{L}0 \supset \mathcal{L}1 \supset \mathcal{L}2 \supset \mathcal{L}3$$

- Gramaticile monotona
  - $\forall\, \alpha \to \beta \in$ P: $|\alpha| <= |\beta|$ $\qquad \alpha, \beta \in (N \cup \Sigma)^*$
  - caz special: **S→ ε** poate $\in$. P In acest caz S nu apare în membrul drept al nici unei reguli de productie.

- Gramatica dependenta de context

  reguli de productie sunt de forma:

  $$\alpha\, A\, \beta \to \alpha\, \gamma\, \beta \qquad A \in N$$

  $$\alpha, \beta, \gamma \in (N \cup \Sigma)^*, \gamma \neq \boldsymbol{\varepsilon}$$

  - caz special: **S→ ε** poate $\in$. P In acest caz S nu apare în membrul drept al nici unei reguli de productie.

Tipuri de gramatici

- Gramatica regulara:

 reg. prod. sunt de forma

  - $A \rightarrow aB$
  - $A \rightarrow b$ unde $A,B \in N$ si $a,b \in \Sigma$

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

- Gramatica independenta de context:

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

Tipuri de gramatici

Gramaticile regulare si liniare in diverse surse.

Conventiile folosite in cadrul acestui curs:
vezi definitiile
de pe slide-urile anterioare.

Gramatica regulara la dreapta

- $A \rightarrow aB$
- $A \rightarrow b$

Discutie: unele surse accepta si: **A→ ε**

alte surse nu accepta deloc **ε** -productii

Gramatica regulara la stanga

- $A \rightarrow B\ a$
- $A \rightarrow b$ **...**

Gramatica liniara la dreapta (gr.regulara la dreapta extinsa)

- $A \rightarrow a1\ a2\ \ldots\ an\quad B$
- $A \rightarrow b1\ b2\ \ldots\ bm$ **...**

Gramatica liniara la stanga (gr.regulara la dreapta extinsa)

...

Gramatica liniara:

- are cel mult un neterminal in membrul drept al regulii de productie

<u>e.g.</u>: $S \rightarrow aSb$

$S \rightarrow ab$

Gramatici independente de context

Ne reamintim:

- gram. ε -independenta
- eliminarea regulilor de productie de redenumire
- forma normala Greibach
- forma normala Chomsky (FNC)

[...] si echivalente

Forma normala Greibach

Gramatica in forma normala Greibach

- reg. prod. sunt de forma
  - $A \to a\ \alpha$ unde $\alpha \in (N \cup \Sigma)^*$ si $a \in \Sigma$

- caz special:

  S→ ε poate $\in$ P

  In acest caz S nu apare în membrul drept al nici unei reguli de productie.

Forma normala Greibach

<u>Teorema</u>:

Pentru orice gramatica independenta de context
exista o gramatica in forma normala Greibach echivalenta.

<u>Constructie:</u>

Fie: gram. ε –independenta, fara redenumiri

1. (similar cu algoritmul pt. eliminarea recursivitatii stanga)
   - impunem o ordine asupra neterminalelor: $N = \{A_1, A_2, \ldots, A_n\}$
     si apoi modific r.p. a.i. sa nu existe $A_i \to A_j\alpha$ cu $i<j$
2. Pentru k = n-1 , … , 1

   avem neterminalul A_k

   pentru toate r.p. $A_k \to A_j\, \alpha$ $(k<j)$

   se inlocuiesc cu: $A_k \to \beta\alpha$,

   pentru toti β a.i. : $A_j \to \beta$ (in toate modurile posibile)

Forma normala Chomsky

Gramatica in forma normala Chomsky

Regulile de productie sunt de forma:

- $A \rightarrow BC$
- $A \rightarrow d$ unde $A,B,C \in N$ si $d \in \Sigma$

caz special:

S→ ε poate sa apara in multimea regulilor de productie,
dar in acest caz S nu apare în membrul drept al nici unei reguli de productie.

Forma normala Chomsky

Teorema:

Pentru orice gramatica independenta de context exista o gramatica in forma normala Chomsky echivalenta

Constructie:

Fie: gram. ε –independenta, fara redenumiri

Folosim transformari de forma:

$A \to X_1 X_2 \ldots X_n \quad : \quad A \to X_1 Z_1$

$Z_1 \to X_2 Z_2$

$Z_{n-2} \to X_{n-1} X_n$

Analizorul CYK (Coucke-Younger-Kasami)

- Se da: G – gramatica independenta de context in Forma Normala Chomsky (FNC)

  $w \in \Sigma^*$

- Se cere sa se verifice daca $w \in L(G)$

PP. $w = a_1 a_2 \ldots a_n$

Analizorul CYK (Coucke-Younger-Kasami)

$t_{ij}=\{A \in N \mid A \Rightarrow^* a_i \dots a_{i+j-1}\}$

Constructia tabelului $T = (t_{ij})$

- $t_{i1} = \{A \in N \mid A \rightarrow a_i \}$
- $t_{ij} = \bigcup_{k=1,j-1}\{A \in N \mid A \rightarrow BC,\ B \in t_{ik},\ C \in t_{i+k,j-k}\}$

$S \in t_{1n}$?

Analizorul CYK (Coucke-Younger-Kasami)

Fie gramatica cu r.p.:

$S \rightarrow AA \mid AS \mid b$

$A \rightarrow SA \mid AS \mid a$

$w = abaab$

? $w \in L(G)$

Analizorul CYK (Coucke-Younger-Kasami)

Avem T

S $\in T_{1,n}$

? sirul regulilor de productie aplicate ?

Subalg. recursiv;

- folosește G,T, $w=a_1 \ldots a_n$ (date) si obtine r.p. (rezultate)

Subalg. genProd(i,j,A)

- genereaza r.p. ce genereaza substringul $a_i..a_{i+j-1}$ din w
- pornind de la neterminalul A din $T_{i,j}$

Analizorul CYK (Coucke-Younger-Kasami)

Subalg. genprod(i,j,A)

Daca j=1 atunci

*cautam $A \rightarrow a_j \in P$

*rez.partial: $A \rightarrow a_j$

altfel

*cautam cel mai mic k a.i. $A \rightarrow BC \in P$

$B \in t_{ik}$, $C \in t_{i+k,j-k}$

*rez.partial : $A \rightarrow BC$

genprod(i,k,B)

genprod(i+k,j-k,C)

sf.daca

Sf.subalg.genprod