Limbaje Formale, Automate şi Compilatoare

Curs 5

2017-18

Curs 5

1. Gramatici şi limbaje independente de context
2. Eliminarea regulilor de ştergere şi a redenumirilor
3. Forma normală Chomsky
4. Problema recunoaşterii: algoritmul Cocke Younger Kasami

Curs 5

Gramatici independente de context

- Gramatici de tip 2 (independente de context): $G = (N, T, S, P)$
  - N şi T sunt mulţimi nevide, finite, disjuncte de neterminali (variabile), respectiv terminali
  - $S \in N$ este simbolul de start
  - $P = \{x \to u | x \in N, u \in (N \cup T)^*\}$ este mulţimea regulilor (producţiilor).

- 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 $G = (N, T, S, P)$ si $w \in L(G)$

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

Exemplu

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

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

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

Definiţie 1

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

- *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 etichetaţi de la stânga la dreapta respectiv cu* X_1, X_2,..., X_n, *atunci* $A \to X_1X_2\dots X_n$ *este o regulă.*
  Cazul în care regula este $A \to \epsilon$ *reprezintă un caz special: nodul etichetat cu A are un singur descendent etichetat cu* ϵ.

Arbori sintactici

Definiţie 2

- *Frontiera unui arbore de derivare* este cuvântul $w = a_1 a_2 \ldots a_n$ 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.

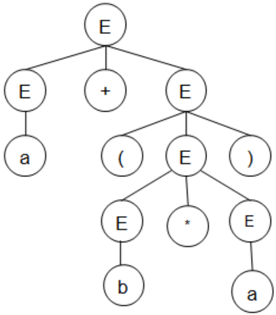
Exemplu

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

$a + (b * a)$

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

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



Ambiguitate

Definiţie 3

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

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

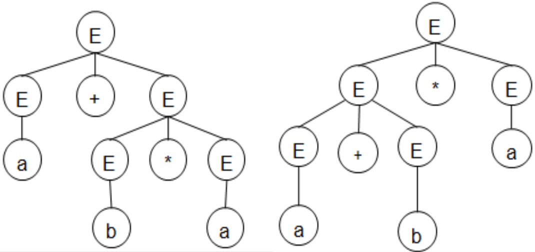
Ambiguitate

Definiţie 3

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

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

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



Ambiguitate

Definiţie 3

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

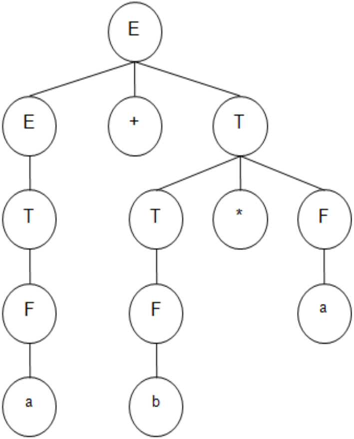
- Echivalent: w are 2 derivări extrem stângi(drepte) distincte.
- Problema ambiguităţii gramaticilor de tip 2 este nedecidabilă: nu există un algoritm care pentru o gramatică oarecare G să testeze dacă G este sau nu ambiguă

Exemplu: o gramatică echivalentă neambiguă

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

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

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



Curs 5

Eliminarea regulilor de ştergere

- Intrare: $G = (N, T, S, P)$
- Ieşire: $G' = (N, T, S, P')$, $L(G') = L(G)$, P' nu conţine reguli de ştergere (reguli de forma $A \to \epsilon$)

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

Eliminarea regulilor de ştergere

- Intrare: $G = (N, T, S, P)$
- Ieşire: $G' = (N, T, S, P')$, $L(G') = L(G)$, P' nu conţine reguli de ştergere (reguli de forma $A \to \epsilon$)

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

Are loc:

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

Eliminarea regulilor de ştergere

P' se obţine din P astfel:

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

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

- se înlocuieşte fiecare regulă de acest fel cu mulţimea de reguli de forma

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

  în toate modurile posibile (2^n)
- se elimină toate regulile de ştergere
- pentru a obţine cuvântul nul (dacă S este în N_ϵ) se adaugă S' simbol de start nou şi regulile $S' \to S$, $S' \to \epsilon$

Exemplu

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

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

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

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

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

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

```
for(A ∈ N){
    N_0 = {A};  i = 0;
    do{
        i = i + 1;
        N_i = N_{i−1} ∪ {C|C ∈ N,  ∃B → C ∈ P, B ∈ N_{i−1}};
    } while  N_i ≠ N_{i−1};
    N_A = N_i; //N_A = {X ∈ N|A ⇒* X}
}
P' = {X → α ∈ P|α ∉ N}
for(X → α_1|α_2|...|α_n ∈ P')
    for(A ∈ N  &&  X ∈ N_A, X ≠ A)
        P' = P' ∪ {A → α_1|α_2|...|α_n}
```

Exemplu

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

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

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

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

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

Curs 5

Forma normală Chomsky

Definiţie 4

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

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

Teorema 1

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

Demonstraţie

- Se elimină regulile de ştergere şi redenumirile

Demonstraţie

- Se elimină regulile de ştergere şi redenumirile
- Se elemină regulile care nu sunt în formă normală Chomsky: Dacă $A \rightarrow x_1 x_2 \ldots x_n$, $n > 1$ este o astfel de regulă atunci o înlocuim cu $A \rightarrow Y_1 Y_2 \ldots Y_n$ 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) şi se adaugă regula $x_a \rightarrow a$

Demonstrație

- Se elimină regulile de ştergere şi redenumirile
- Se elemină regulile care nu sunt în formă normală Chomsky: Dacă $A \to x_1 x_2 \ldots x_n$, $n > 1$ este o astfel de regulă atunci o înlocuim cu $A \to Y_1 Y_2 \ldots Y_n$ 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) şi se adaugă regula $x_a \to a$
- O regulă de forma $A \to Y_1 Y_2 \ldots Y_n$, dacă $n > 2$, o înlocuim cu:
  - $A \to Y_1 Z_1$
  - $Z_1 \to Y_2 Z_2$
  - $\ldots\ldots$
  - $Z_{n-3} \to Y_{n-2} Z_{n-2}$
  - $Z_{n-2} \to Y_{n-1} Y_n$,

  unde $Z_1, Z_2, \ldots, Z_{n-2}$ sunt neterminali noi.

Exemplu

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

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

Gramatica echivalentă în formă normală Chomsky

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

- $S \rightarrow x_a Z_1 | x_c Z_2$
- $Z_1 \rightarrow S x_b$
- $Z_2 \rightarrow A x_c$
- $A \rightarrow x_c A | c$
- $x_a \rightarrow a$
- $x_b \rightarrow b$
- $x_c \rightarrow c$

Curs 5

Algoritmul Cocke Younger Kasami (CYK)

- Problema recunoaşterii 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 constuiesc mulţimile

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

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

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

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

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

    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

- Notaţie:

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

- 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, $w = a_1 a_2 \dots a_n$
- Ieşire: $w \in L(G)$?

```
for(i=1; i<=n; i++)
    V_i1 = {A | ∃A → a_i ∈ P};
for(j=2; j<=n; j++)
    for (i=1; i<=n+1-j; i++){
        V_ij = ∅;
        for(k=1; k<=j-1; k++)
            V_ij = V_ij ∪ (V_ik ∘ V_{i+k j-k});
    }
if(S ∈ V_1n) w ∈ L(G) else w ∉ 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:

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

$w = abc$

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

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