Limbaje Formale, Automate şi Compilatoare

Curs 5

2020-21

Curs 5

1. Gramatici şi limbaje independente de context
2. Forma redusă pentru gramatici independente de context
3. Eliminarea regulilor de ştergere şi a redenumirilor

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
  - $P = \{x \rightarrow u | x \in N, u \in (N \cup T)^*\}$ 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:

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

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 $A \to X_1X_2 \ldots X_n$ este o regulă.*
  Dacă A are un succesor etichetat cu ϵ (pentru regula $A \to \epsilon$), nodul etichetat cu A nu mai are alţi succesori.

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*.*

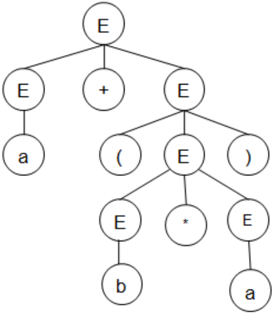
Exemplu

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

$a + (b * a)$

- 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.

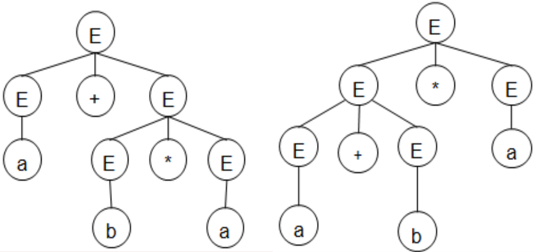
Ambiguitate

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.

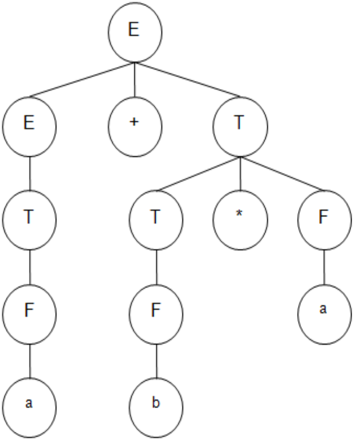
- Echivalent: w are 2 derivări extrem stângi(drepte) distincte.
- Problema ambiguităţii gramaticilor de tip 2 este nedecidabilă: nu există un algoritm care pentru o gramatică oarecare G să testeze dacă G este sau nu ambiguă

Exemplu: o gramatică echivalentă neambiguă

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

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

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



Curs 5

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

Definiţie 4

O gramatică este în formă redusă, dacă nu conţine 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 acelaşi tip în formă redusă echivalentă cu G.
- Pentru eliminarea simbolurilor inutile:
  - Se determină şi apoi se elimină simbolurile neproductive şi toate regulile ce conţin măcar unul dintre acestea.
  - Se determină apoi se elimină simbolurile inaccesibile şi toate regulile aferente.

Eliminarea simbolurilor neproductive - algoritm

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

$N_0 = \emptyset$; $i = 0$;
`do` {
    $i = i + 1$;
    $N_i = N_{i-1} \cup \{A | A \to \alpha \in P, \alpha \in (N_{i-1} \cup T)^*\}$;
} `while` $N_i \neq N_{i-1}$;
$N' = N_i$;
$P' = \{A \to \alpha \in P | A \in N', \alpha \in (N' \cup T)^*\}$;

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

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: $G = (N, T, S, P)$
- Ieşire: $G' = (N', T', S, P')$, $L(G') = L(G)$, N', T' conţin doar simboluri accesibile

$V_0 = \{S\};\ i = 0;$
do $\{$
$\quad i = i + 1;$
$\quad V_i = V_{i-1} \cup \{X | X \in N \cup T,\ \ \exists A \to \alpha X \beta \in P, A \in (V_{i-1} \cap N)\};$
$\}$ while $V_i \neq V_{i-1};$
$N' = V_i \cap N;$
$T' = V_i \cap T;$
$P' = \{A \to \alpha \in P | A \in N', \alpha \in (N' \cup T')^*\};$

- X accesibil ddacă $X \in V_i$

Exemplu

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

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

- Eliminarea simbolurilor neproductive duce la:

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

- Eliminarea simbolurilor inaccesibile duce la:

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

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

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` $\{$

$\quad i = i + 1;$

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

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

$N_\epsilon = N_i;$

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:

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

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

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

  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

$\texttt{for}(A \in N)\{$

$\quad N_0 = \{A\};\ i = 0;$

$\quad \texttt{do}\{$

$\qquad i = i + 1;$

$\qquad N_i = N_{i-1} \cup \{C | C \in N,\ \ \exists B \to C \in P, B \in N_{i-1}\};$

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

$\quad N_A = N_i; // N_A = \{X \in N | A \Rightarrow^* X\}$

$\}$

$P' = \{X \to \alpha \in P | \alpha \notin N\}$

$\texttt{for}(X \to \alpha_1 | \alpha_2 | \ldots | \alpha_n \in P')$

$\quad \texttt{for}(A \in N\ \ \&\&\ \ X \in N_A, X \neq A)$

$\qquad P' = P' \cup \{A \to \alpha_1 | \alpha_2 | \ldots | \alpha_n\}$

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':

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