AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Przekształcenia gramatyk bezkontekstowych

Teoria automatów i języków formalnych

Dr inż. Janusz Majewski
Katedra Informatyki

Symbole nadmiarowe, gramatyka bez ε-produkcji

Niech będzie dana gramatyka bezkontekstowa $G = <V, \Sigma, P, S> \in \mathcal{G}_{BK}$

Symbol $X \in (V \cup \Sigma)$ nazywamy nieużytecznym w $G \in \mathcal{G}_{BK}$ jeśli nie można w tej gramatyce przeprowadzić wyprowadzenia:

$$S \Rightarrow^*_G wXy \Rightarrow^*_G wxy$$

przy czym $w,x,y \in \Sigma^*$ (czyli z X nie można wyprowadzić żadnego słowa w symbolach terminalnych).

Symbol $X \in (V \cup \Sigma)$ nazywamy nieosiągalnym w $G \in \mathcal{G}_{BK}$, jeśli X nie pojawia się w żadnej formie zdaniowej tej gramatyki (czyli nie można go wyprowadzić z S).

Gramatyka $G \in \mathcal{G}_{BK}$ jest gramatyką bez ε-produkcji, gdy:

(i) P nie zawiera produkcji postaci $A \to \varepsilon$, $A \in V$, $A \neq S$

(ii) jeśli $(S \to \varepsilon) \in P$ to symbol S nie występuje w prawych stronach pozostałych produkcji

Możliwość przekształcania gramatyk

Twierdzenie

Każdą gramatykę bezkontekstową $G = <V, \Sigma, P, S> \in G_{BK}$ można przekształcić do postaci równoważnej $G' = <V', \Sigma', P', S> \in G_{BK}$ (to znaczy takiej, że $L(G') = L(G)$) nie zawierającej symboli nieosiągalnych, symboli nieużytecznych oraz będącej gramatyką bez ε-produkcji (w sensie powyższych definicji).

Sposoby wykonania odpowiednich przekształceń podają dalej podane algorytmy. Słuchacze zechcą udowodnić poprawność tych algorytmów.

Algorytm usuwania symboli nieosiągalnych (1)

wejście: $G = <V, \Sigma, P, S> \in \mathcal{G}_{BK}$

wyjście: $G' = <V', \Sigma', P', S> \in \mathcal{G}_{BK}$ taka, że:

(i) $L(G') = L(G)$

(ii) $(\forall X \in (V' \cup \Sigma')) (\exists \alpha,\beta \in (V' \cup \Sigma')^*) (S \Rightarrow^*_{G'} \alpha X \beta)$

Metoda:

$W_0 := \{S\}$;

$i := 0$;

repeat

$i := i + 1$;

$W_i := W_{i-1} \cup \{X \in (V \cup \Sigma) \mid (A \rightarrow \alpha X \beta) \in P \wedge A \in W_{i-1} \wedge \alpha,\beta \in (V \cup \Sigma)^* \}$;

until $W_i = W_{i-1}$;/* W_i zawiera te symbole terminalne i nieterminalne które:

- występują w produkcjach z P
- dają się wyprowadzić z S */

Algorytm usuwania symboli nieosiągalnych (2)

$V' := W_i \cap V;$

$\Sigma' := W_i \cap \Sigma;$

$P' := \{(u \rightarrow v) \in P \mid u,v \in W_i^* \}$

/* P' zawiera te produkcje z P, które zbudowane są z symboli z W_i */

$G' := \langle V', \Sigma', P', S \rangle$

Algorytm usuwania symboli nieużytecznych

wejście: $G = \langle V, \Sigma, P, S \rangle \in \mathcal{G}_{BK}$

wyjście: $G' = \langle V', \Sigma', P', S \rangle \in \mathcal{G}_{BK}$ taka, że:

(i) $L(G') = L(G)$

(ii) $(V' \cup \Sigma')$ nie zawiera symboli nieużytecznych

Metoda:

$V_0 := \varnothing;$ /* $\varnothing$ - zbiór pusty */

$i := 0;$

repeat

$i := i + 1;$

$V_i := V_{i-1} \cup \{ A \in V \mid (A \rightarrow \alpha) \in P \wedge \alpha \in (V_{i-1} \cup \Sigma)^* \}$

until $V_i = V_{i-1}$; /* V_i zawiera wszystkie nieterminale „użyteczne” */

$G_1 := \langle V \cap V_i, \Sigma, P_1, S \rangle$ gdzie

$P_1 := \{(u \rightarrow v) \in P \mid u,v \in (V_i \cup \Sigma)^* \};$

zastosować do G_1 algorytm usuwania symboli nieosiągalnych uzyskując w wyniku tego gramatykę $G' = \langle V', \Sigma', P', S \rangle$

Przykład usuwania symboli nadmiarowych

Przykład:

- G = <{S, A, B}, {a, b}, { S→a
  S→A
  A→AB
  B→b }, S>
- G_1 = <{S, B}, {a, b}, { S→a
  B→b }, S>
- G' = <{S}, {a}, { S→a }, S>

Algorytm usuwania ε-produkcji (1)

wejście: $G = \langle V, \Sigma, P, S \rangle \in \mathcal{G}_{BK}$

wyjście: $G' = \langle V', \Sigma, P', S' \rangle \in \mathcal{G}_{BK}$ taka, że:

(i) L(G) = L(G');

(ii) G' jest gramatyką bez ε-produkcji

Metoda:

$V_0 := \varnothing$;

i := 0;

repeat

i:= i + 1;

$V_i := V_{i-1} \cup \{ A \in V \mid (A \to x) \in P \wedge x \in V_{i-1}^* \}$;

until $V_i = V_{i-1}$;

/* $V_i = \{A \mid A \in V \wedge A \Rightarrow^+_G \varepsilon\}$ */

Algorytm usuwania ε-produkcji (2)

P′ := P ;

for $(A \to x) \in P$ do

begin

przedstawiamy „x” w postaci $\alpha_0 B_1 \alpha_1 B_2 \alpha_2 \ldots B_k \alpha_k$, $k \geq 0$, gdzie $B_j \in V_i$ $(1 \leq j \leq k)$, $\alpha_j \in ((V \cup \Sigma) - V_i)^*$ $(0 \leq j \leq k)$;

$P' := P' \cup \{(A \to \alpha_0 X_1 \alpha_1 X_2 \alpha_2 \ldots X_k \alpha_k) \mid (X_j = B_j \vee X_j = \varepsilon)\}$;

if $(A \to \varepsilon) \in P'$ then $P' := P' - \{(A \to \varepsilon)\}$;

end;

Algorytm usuwania ε-produkcji (3)

if $S \in V_i$ then

begin

$V' := V \cup \{S'\}$;

$P' := P' \cup \{(S' \to S), (S' \to \varepsilon)\}$;

$S' := S'$;

end

else

begin

$V' := V$;

$S' := S$;

end;

Przykład usuwania ε-produkcji

Przykład:

$S \rightarrow aSbS$

$S \rightarrow bSaS$

$S \rightarrow \varepsilon$

Po usunięciu $S \rightarrow \varepsilon$ otrzymujemy

$S' \rightarrow S$

$S' \rightarrow \varepsilon$

$S \rightarrow aSbS$, $S \rightarrow aSb$, $S \rightarrow abS$, $S \rightarrow ab$

$S \rightarrow bSaS$, $S \rightarrow bSa$, $S \rightarrow baS$, $S \rightarrow ba$

Produkcje łańcuchowe

Produkcje łańcuchowe, to produkcje postaci: $A \rightarrow B$; gdzie $A, B \in V$

Gramatyka G zawiera cykle, wówczas gdy istnieją w gramatyce wyprowadzenia postaci: $A \Rightarrow^{+}_{G} A$; $A \in V$

Cykle typu $A \Rightarrow^{+} A$ powstają poprzez istnienie w gramatyce produkcji łańcuchowych, np. $A \rightarrow B, B \rightarrow C, C \rightarrow A$, więc usuwając produkcje łańcuchowe – usuwamy również automatycznie wszystkie cykle.

Twierdzenie

Każdą gramatykę bezkontekstową $G = <V, \Sigma, P, S> \in \mathcal{G}_{BK}$ można przekształcić do postaci równoważnej $G' = <V', \Sigma', P', S> \in \mathcal{G}_{BK}$ (to znaczy takiej, że $L(G') = L(G)$) nie zawierającej produkcji łańcuchowych i cykli (w sensie powyższych definicji).

Algorytm usuwania produkcji łańcuchowych i cykli (1)

wejście: G = <V, Σ, P, S> $\in \mathcal{G}_{BK}$; G – bez ε-produkcji!
wyjście: G' = <V, Σ, P', S> $\in \mathcal{G}_{BK}$ taka, że
(i) L(G) = L(G')
(ii) G' – nie zawiera produkcji łańcuchowych i cykli
Metoda:

```
for A ∈ V do
  begin
    V_0 := {A};
    i := 0;
    repeat
      i := i + 1;
      V_i := V_{i-1} ∪ { C ∈ V | (B → C) ∈ P ∧ B ∈ V_{i-1} }
    until V_{i-1} = V_i ;
    V_A := V_i ;              /* V_A = { B | A ⇒*_G B } */
  end;
```

Algorytm usuwania produkcji łańcuchowych i cykli (2)

```
P' := ∅;   /* ∅ - zbiór pusty */
for p ∈ P do
  begin
    if p = (B → α) and α ∉ V then
                                  /* p nie jest łańcuchowa */
      begin
        M_B := {A ∈ V | B ∈ V_A};
                                  /* M_B = { A | A ⇒*_G B} */
        for A ∈ M_B do
            P' := P' ∪ { A → α };
      end;
  end;
```

Przykład usuwania produkcji łańcuchowych

Przykład: Rozpatrujemy gramatykę:

(1) $E \to E + T$
(2) $E \to T$
(3) $T \to T * F$
(4) $T \to F$
(5) $F \to (E)$
(6) $F \to id$

Wyznaczamy odpowiednie zbiory nieterminali:

$V_E = \{ E, T, F \}$ $V_T = \{ T, F \}$ $V_F = \{ F \}$
$M_E = \{ E \}$ $M_T = \{ E, T \}$ $M_F = \{ E, T, F \}$

Analizujemy kolejno wszystkie produkcje gramatyki przekształcanej

(1) $E \to E + T$
(2) wypada, bo jest łańcuchowa
(3) $E \to T * F$ $T \to T * F$
(4) wypada, bo jest łańcuchowa
(5) $E \to (E)$ $T \to (E)$ $F \to (E)$
(6) $E \to id$ $T \to id$ $F \to id$

Gramatyka prawidłowa Lewostronna rekursja

Gramatyka $G = < V, \Sigma, P, S > \in \mathcal{G}_{BK}$ nazywa się <u>prawidłową</u>, jeśli:

(i) nie zawiera cykli,

(ii) jest gramatyką bez ε-produkcji,

(iii) nie zawiera nieużytecznych symboli.

$U \in V$ jest symbolem lewostronnie rekursywnym gramatyki G jeżeli $U \Rightarrow^{+}_{G} U\beta\,;\ \beta \in (V \cup \Sigma)^*$

Gramatyka G nazywa się <u>lewostronnie rekursywną</u> $\Leftrightarrow$
$(\exists\, U \in V)\ (\ U \Rightarrow^{+}_{G} U\beta\,;\ \beta \in (V \cup \Sigma)^*\ )$

Usuwanie lewostronnej rekursji (1)

Twierdzenie

Dana jest gramatyka $G = < V, \Sigma, P, S > \in \mathcal{G}_{BK}$ oraz symbol nieterminalny $A \in V$.

Zbiór $\{ A \to A\alpha_1 \mid \ldots \mid A\alpha_m \mid \beta_1 \mid \ldots \mid \beta_n \} \subseteq P$ jest zbiorem wszystkich produkcji mających po lewej stronie symbol A ; żadne z $\beta_i \in (V \cup \Sigma)^*$ (i = 1, ... , n) nie zaczyna się od symbolu A.

Niech $G' = < V \cup \{A'\}, \Sigma, P', S >$ będzie nową gramatyką, w której A' – nowy symbol nieterminalny, zaś zbiór produkcji P' jest określony jak niżej

$P' = P - \{A \to A\alpha_1 \mid \ldots \mid A\alpha_m \mid \beta_1 \mid \ldots \mid \beta_n \} \cup \{ A \to \beta_1 A' \mid \ldots \mid \beta_n A' \}$
$\cup \{ A' \to \alpha_1 A' \mid \ldots \mid \alpha_m A' \mid \varepsilon \}$

Wówczas zachodzi L(G) = L(G').

Twierdzenie powyższe podaje sposób usuwania lewej rekursji, ale tylko bezpośredniej.

Twierdzenie

Każdą gramatykę bezkontekstową $G = <V, \Sigma, P, S> \in \mathcal{G}_{BK}$ można przekształcić do postaci równoważnej $G' = <V', \Sigma', P', S> \in \mathcal{G}_{BK}$ (to znaczy takiej, że L(G') = L(G)) nie zawierającej (bezpośredniej i pośredniej) lewostronnej rekursji.

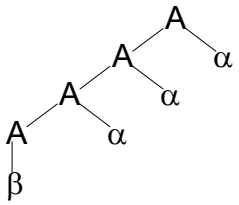
Przykład usuwania lewostronnej rekursji

Przykład:

$A \to A\alpha$

$A \to \beta$

Analizowane słowo : $\beta\alpha\alpha\alpha$



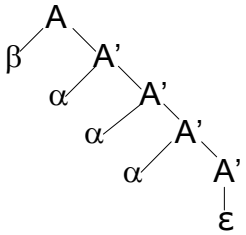
Po modyfikacji:

$A \to \beta A'$

$A' \to \varepsilon$

$A' \to \alpha A'$

Analizowane słowo : $\beta\alpha\alpha\alpha$



Algorytm usuwania lewostronnej rekursji (1)

wejście: $G = \langle V, \Sigma, P, S \rangle \in \mathcal{G}_{BK}$,
G –gramatyka prawidłowa!

wyjście: $G' = \langle V', \Sigma, P', S \rangle \in \mathcal{G}_{BK}$ taka, że

(i) G' – nie jest gramatyką lewostronnie rekursywną

(ii) L(G') = L(G)

Metoda:

numerujemy symbole nieterminalne, czyli
$V := \{A_1, \ldots, A_n\}$; $n := \#V$

V' := V ;

P' := P ;

i := 1 ;

Algorytm usuwania lewostronnej rekursji (2)

repeat

if $\exists(A_i \to A_i\delta) \in P'$ and $\delta \in (V \cup \Sigma)^*$ then begin

$V' := V' \cup \{A_i'\}$

$P' := P' - \{A_i \to A_i\alpha_1 \mid \ldots \mid A_i\alpha_q \mid \beta_1 \mid \ldots \mid \beta_p\}$
$\cup \{A_i \to \beta_1 A_i' \mid \ldots \mid \beta_p A_i'\} \cup \{A_i' \to \alpha_1 A_i' \mid \ldots \mid \alpha_q A_i' \mid \varepsilon\}$ end;

/* q – liczba wszystkich produkcji $A_i \to A_i \ldots$ (lewostr. rekurs.)
żadne z $\beta_1, \ldots, \beta_p$ nie zaczyna się od A_i
p – liczba wszystkich produkcji $A_i \to \ldots$ bez lewej rekursji */

if i = n then STOP else i := i + 1 ;

for j := 1 to i –1 do

$P' := P' - \{A_i \to A_j\alpha_1 \mid \ldots \mid A_j\alpha_x\} \cup \{ (A_i \to \gamma_k\alpha_l) \mid l = 1, \ldots, x ; k = 1, \ldots, m$
γ_k – prawe strony produkcji: $A_j \to \gamma_1 \mid \ldots \mid \gamma_k \mid \ldots \mid \gamma_m\}$;

/* x – liczba wszystkich produkcji typu $A_i \to A_j\alpha$
m – liczba wszystkich produkcji typu $A_j \to \gamma_1 \mid \ldots \mid \gamma_m$;
w ten sposób prawa strona każdej produkcji $A_i \to \ldots$ zaczyna się albo od symbolu terminalnego, albo od nieterminala A_j o numerze $j > i$ */

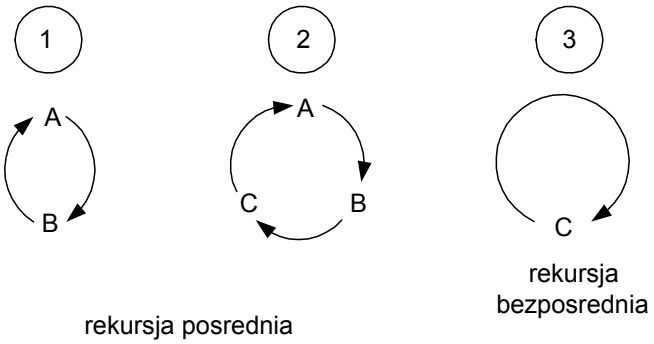
until FALSE ;

Przykład usuwania lewostronnej rekursji (1)

Przykład:

$A \rightarrow BC \mid a$
$B \rightarrow CA \mid Ab$
$C \rightarrow AB \mid CC \mid a$

Rodzaje rekursji lewostronnej:



Przykład usuwania lewostronnej rekursji (2)

Przykład:

$A \rightarrow BC \mid a$
$B \rightarrow CA \mid Ab$
$C \rightarrow AB \mid CC \mid a$

I	i = 1	G bez zmian
II	i = 2 ; j = 1	$B \rightarrow CA \mid BCb \mid ab$
I	i = 2	$B \rightarrow CAB' \mid abB'$
		$B' \rightarrow CbB' \mid \varepsilon$
II	i = 3 ; j = 1	$C \rightarrow BCB \mid aB \mid CC \mid a$
	i = 3 ; j = 2	$C \rightarrow CAB'CB \mid abB'CB \mid aB \mid CC \mid a$
I	i = 3	$C \rightarrow abB'CBC' \mid aBC' \mid aC'$
		$C' \rightarrow AB'CBC' \mid CC' \mid \varepsilon$

Ostatecznie:

$A \rightarrow BC \mid a$
$B \rightarrow CAB' \mid abB'$
$B' \rightarrow CbB' \mid \varepsilon$
$C \rightarrow abB'CBC' \mid aBC' \mid aC'$
$C' \rightarrow AB'CBC' \mid CC' \mid \varepsilon$

Usuwanie lewostronnej rekursji (2)

Wariant drugi usuwania lewostronnej rekursji – dla potrzeb przekształcania gramatyki bezkontekstowej do postaci normalnej Greibach

Twierdzenie (wariant drugi)

Dana jest gramatyka $G = < V, \Sigma, P, S > \in \mathcal{G}_{BK}$ oraz symbol nieterminalny $A \in V$.

Zbiór $\{ A \to A\alpha_1 \mid ... \mid A\alpha_m \mid \beta_1 \mid ... \mid \beta_n \} \subseteq P$ jest zbiorem wszystkich produkcji mających po lewej stronie symbol A ; żadne z $\beta_i \in (V \cup \Sigma)^*$ (i = 1, ... , n) nie zaczyna się od symbolu A.

Niech $G' = < V \cup \{A'\}, \Sigma, P', S >$ będzie nową gramatyką, w której A' – nowy symbol nieterminalny, zaś zbiór produkcji P' jest określony jak niżej

$P' = P - \{A \to A\alpha_1 \mid ... \mid A\alpha_m \} \cup \{ A \to \beta_1 A' \mid ... \mid \beta_n A' \}$
$\cup \{ A' \to \alpha_1 A' \mid ... \mid \alpha_m A' \mid \alpha_1 \mid ... \mid \alpha_m \}$

Wówczas zachodzi L(G) = L(G').

Twierdzenie podaje drugi sposób usuwania bezpośredniej lewej rekursji (nie powodujący dodawania ε–produkcji, ale zwiększający liczbę nowych produkcji wprowadzanych do gramatyki). Cała reszta algorytmu pozostaje niezmieniona – por. http://kompilatory.agh.edu.pl