AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Właściwości języków bezkontekstowych

Teoria automatów i języków formalnych

Dr inż. Janusz Majewski
Katedra Informatyki

Postać normalna Chomsky'ego (1)

Twierdzenie:

Dowolny język bezkontekstowy nie zawierający słowa pustego ε jest generowany przez gramatykę, której wszystkie produkcje są postaci

$$A \to BC \qquad \text{lub} \qquad A \to a$$

gdzie: A, B, C $\in$ V oraz a $\in \Sigma$.

Algorytm przekształcania gramatyki bezkontekstowej do postaci normalnej Chomsky'ego:

wejście: $G = \langle V, \Sigma, P, S \rangle \in G_{BK}$, gramatyka bez ε–produkcji i bez produkcji łańcuchowych, nie zawierająca produkcji $S \to \varepsilon$

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

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

(ii) wszystkie produkcje gramatyki G' mają postać $A \to BC$ lub $A \to a$

Postać normalna Chomsky'ego (2)

Metoda:

$V' := V$; $\Sigma' := \Sigma$; $S' := S$;

$P' := \emptyset$;

$P' := P' \cup \{ A \to a \mid (A \to a) \in P, A \in V, a \in \Sigma \}$; /* te produkcje są już w dobrej postaci */

for każda produkcja $(A \to X_1X_2...X_n) \in P \wedge n \geq 2$ do

begin

for $i := 1$ to n do

if $X_i = a \wedge a \in \Sigma$ then begin

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

$P' := P' \cup \{ C_a \to a \}$;

$B_i := C_a$; end

else $B_i := X_i$;

$P' := P' \cup \{ A \to B_1B_2...B_n \}$;

end; /* każda prawa strona produkcji z P' nie krótsza niż 2 składa się z samych nieterminali */

for każda produkcja $(A \to B_1B_2...B_n) \in P' \wedge n > 2$ do

begin

$V' := V' \cup \{ D_1, D_2, ..., D_{n-2} \}$;

$P' := P' - \{ A \to B_1B_2...B_n \} \cup \{ A \to B_1D_1, D_1 \to B_2D_2, ..., D_{n-3} \to B_{n-2}D_{n-2}, D_{n-2} \to B_{n-1}B_n \}$;

end; /* wszystkie produkcje w P' mają poprawną postać */

Przykład (1)

Dana jest gramatyka G:

$S \rightarrow bA \mid aB$

$A \rightarrow bAA \mid aS \mid a$

$B \rightarrow aBB \mid bS \mid b$

Skonstruować równoważną jej gramatykę w postaci normalnej Chomsky'ego.

Dana gramatyka nie zawiera ε–produkcji ani produkcji łańcuchowych. Produkcje $A \rightarrow a$ oraz $B \rightarrow b$ są już w dobrej postaci. Wystąpienia terminali a i b w prawych stronach pozostałych produkcji zamieniamy nowododanymi nieterminalami C_a i C_b

$S \rightarrow C_bA \mid C_aB$

$A \rightarrow C_bAA \mid C_aS \mid a$

$B \rightarrow C_aBB \mid C_bS \mid b$

Przykład (2)

Dodajemy nowe produkcje

$C_a \rightarrow a$

$C_b \rightarrow b$

Wreszcie zbyt długie produkcje $A \rightarrow C_bAA$ oraz $B \rightarrow C_aBB$ zastępujemy krótszymi wprowadzając dwa dodatkowe nieterminale D_1 i D_2.

$A \rightarrow C_bD_1$ $\quad$ $D_1 \rightarrow AA$

$B \rightarrow C_aD_2$ $\quad$ $D_2 \rightarrow BB$

Ostateczny zbiór produkcji gramatyki w postaci normalnej Chomsky'ego jest następujący:

$S \rightarrow C_bA \mid C_aB$

$A \rightarrow C_bD_1 \mid C_aS \mid a$

$B \rightarrow C_aD_2 \mid C_bS \mid b$

$C_a \rightarrow a$

$C_b \rightarrow b$

$D_1 \rightarrow AA$

$D_2 \rightarrow BB$

Postać normalna Greibach (1)

Twierdzenie

Dowolny język bezkontekstowy nie zawierający słowa pustego ε jest generowany przez gramatykę, której wszystkie produkcje są postaci

$$A \to a\alpha$$

gdzie: $A \in V$, $a \in \Sigma$ zaś $\alpha \in V^*$.

Algorytm przekształcania gramatyki bezkontekstowej do postaci normalnej Greibach:

wejście: $G = \langle V, \Sigma, P, S \rangle \in G_{BK}$, gramatyka w postaci normalnej Chomsky'ego

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

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

(ii) wszystkie produkcje gramatyki G' mają postać $A \to a\alpha$ gdzie: $A \in V$, $a \in \Sigma$, $\alpha \in V^*$.

Postać normalna Greibach (2)

Metoda:

Tworzymy gramatykę $G_1 = <V_1, \Sigma, P_1, S>$, usuwając lewostronną rekursję w gramatyce G algorytmem według wariantu drugiego. Zachowujemy numerację symboli nieterminalnych z tamtego algorytmu, tzn. teraz zbiór nieterminali V_1 zawiera symbole $\{A_1, \ldots, A_n\}$ oraz niektóre (być może wszystkie) symbole ze zbioru $\{A_1', \ldots, A_n'\}$

$V' := V_1; \Sigma' := \Sigma;$ $S' := S;$ $P' := P_1;$

for i := n downto 2 do

  for j := n–1 downto 1 do

    for każda produkcja $(A_j \rightarrow A_i\alpha) \in P'$ do

      $P' := P' - \{ A_j \rightarrow A_i\alpha\} \cup \{ A_j \rightarrow \beta_k\alpha \mid k = 1, \ldots, m;$ β_k – prawa strona każdej produkcji $A_i \rightarrow \beta_k$, m – liczba wszystkich produkcji $A_i \rightarrow \beta_k \}$

for i := n downto 1 do

  for j := n downto 1 do

    for każda produkcja $(A_j' \rightarrow A_i\alpha) \in P'$ do

      $P' := P' - \{ A_j' \rightarrow A_i\alpha\} \cup \{ A_j' \rightarrow \beta k\alpha \mid k = 1, \ldots, m;$ β_k – prawa strona każdej produkcji $A_i \rightarrow \beta_k$, m – liczba wszystkich produkcji $A_i \rightarrow \beta_k \}$

Przykład (1)

Dla języka generowanego przez gramatykę G:

$S \rightarrow AB$

$A \rightarrow BS \mid b$

$B \rightarrow SA \mid a$

skonstruować gramatykę w postaci normalnej Greibach.

Ponieważ gramatyka jest już w postaci normalnej Chomsky'ego, więc usuwamy lewostronną rekursję według algorytmu drugiego.

Przyjmujemy następującą numerację symboli nieterminalnych:

$S - A_1$

$A - A_2$

$B - A_3$

Otrzymamy gramatykę G_1:

$S \rightarrow AB$

$A \rightarrow BS \mid b$

$B \rightarrow bBAB' \mid aB' \mid bBA \mid a$

$B' \rightarrow SBAB' \mid SBA$

Przykład (2)

$S \rightarrow AB$

$A \rightarrow BS \mid b$

$B \rightarrow bBAB' \mid aB' \mid bBA \mid a$

$B' \rightarrow SBAB' \mid SBA$

Wszystkie produkcje mające B (czyli nieterminal o najwyższym numerze) po lewej stronie mają prawe strony rozpoczynające się od terminala (czyli mają już poprawną postać). Zastępujemy teraz wystąpienie symbolu B na początku prawej strony produkcji $A \rightarrow BS$ wszystkimi prawymi stronami produkcji mających B po lewej stronie. Produkcje z A po lewej stronie mają teraz postać:

$A \rightarrow bBAB'S \mid aB'S \mid bBAS \mid aS \mid b$

Podobnie postępujemy z produkcją $S \rightarrow AB$ eliminując wystąpienie nieterminala A na początku jej prawej strony.

$S \rightarrow bBAB'SB \mid aB'SB \mid bBASB \mid aSB \mid bB$

Przykład (3)

$S \rightarrow$ bBAB'SB | aB'SB | bBASB | aSB | bB

$A \rightarrow$ bBAB'S | aB'S | bBAS | aS | b

$B \rightarrow$ bBAB' | aB' | bBA | a

$B' \rightarrow$ SBAB' | SBA

Wreszcie eliminujemy wystąpienia S na początku prawych stron obu produkcji mających B' po lewej stronie.

$B' \rightarrow$ bBAB'SBBAB' | aB'SBBAB' | bBASBBAB' | aSBBAB' | bBBAB' | bBAB'SBBA | aB'SBBA | bBASBBA | aSBBA | bBBA

Ostatecznie zbiór produkcji gramatyki w postaci normalnej Greibach jest następujący:

$S \rightarrow$ bBAB'SB | aB'SB | bBASB | aSB | bB

$A \rightarrow$ bBAB'S | aB'S | bBAS | aS | b

$B \rightarrow$ bBAB' | aB' | bBA | a

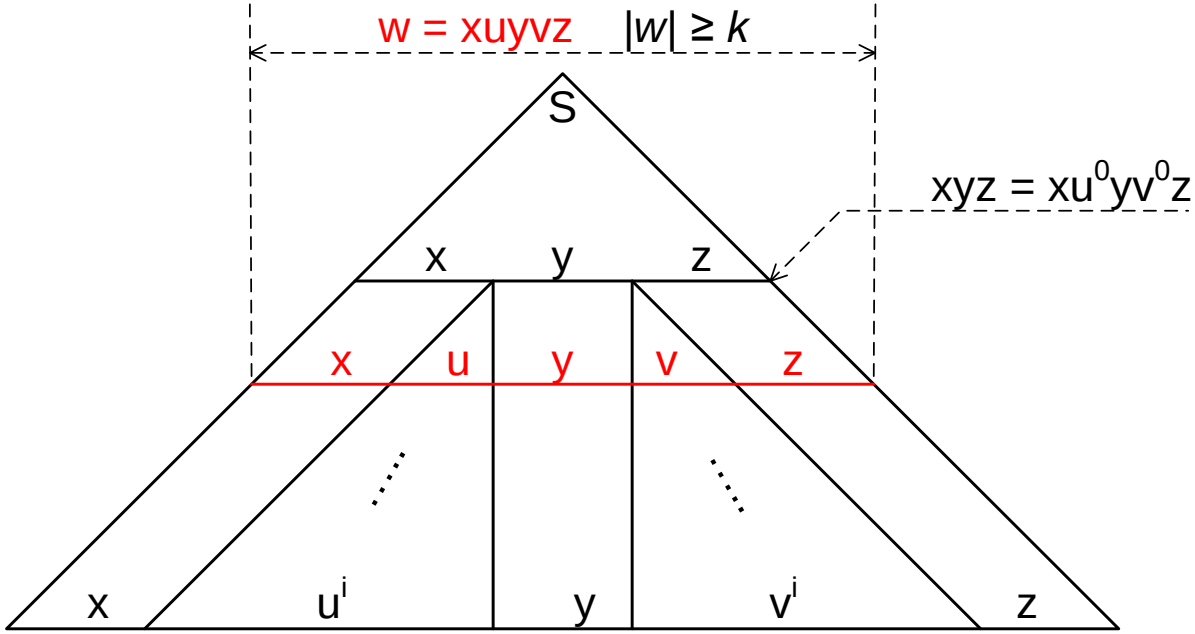
$B' \rightarrow$ bBAB'SBBAB' | aB'SBBAB' | bBASBBAB' | aSBBAB' | bBBAB' | bBAB'SBBA | aB'SBBA | bBASBBA | aSBBA | bBBA

Warunek konieczny bezkontekstowości języka (1)

Twierdzenie: Jeżeli $L \in \mathcal{L}_{BK}$

to $(\exists k)\ ((w \in L \ \wedge\ |w| \geq k) \Rightarrow (w = xuyvz \ \wedge\ uv \neq \varepsilon \ \wedge\ |uyv| \leq k \ \wedge\ (\forall i \geq 0)\ (xu^iyv^iz \in L)))$

Jest to tzw. lemat o rozrastaniu się języków bezkontekstowych. Mówi on o tym, że każde dostatecznie długie słowo języka bezkontekstowego da się przedstawić w postaci *xuyvz* oraz wszystkie słowa o postaci xu^iyv^iz *(i≥0)* też będą należały do tego samego języka.



Warunek konieczny bezkontekstowości języka (2)

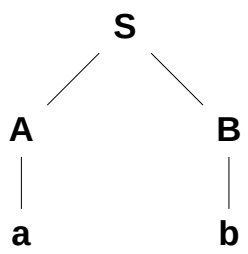
Szkic dowodu:

Niech G będzie gramatyką w postaci normalnej Chomsky'ego generującą $L - \{\varepsilon\}$. Zauważmy, że jeżeli $w \in L(G)$, i w jest długie, to dowolne drzewo rozkładu dla w musi zawierać długą drogę. Dokładniej, jeśli drzewo rozkładu słowa generowanego przez gramatykę w postaci normalnej Chomsky'ego nie zawiera drogi o długości większej od i, to długość danego słowa jest nie większa od 2^{i-1}.

(1)

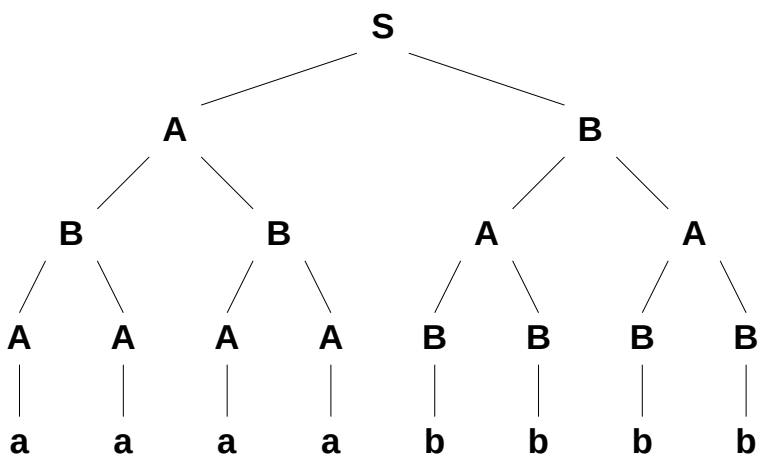
dlugosc_drogi = 1
dlugosc_slowa = 1

(2)



dlugosc_drogi = 2
dlugosc_slowa = 2

(4)



dlugosc_drogi = 4
dlugosc_slowa = 8

Warunek konieczny bezkontekstowości języka (3)

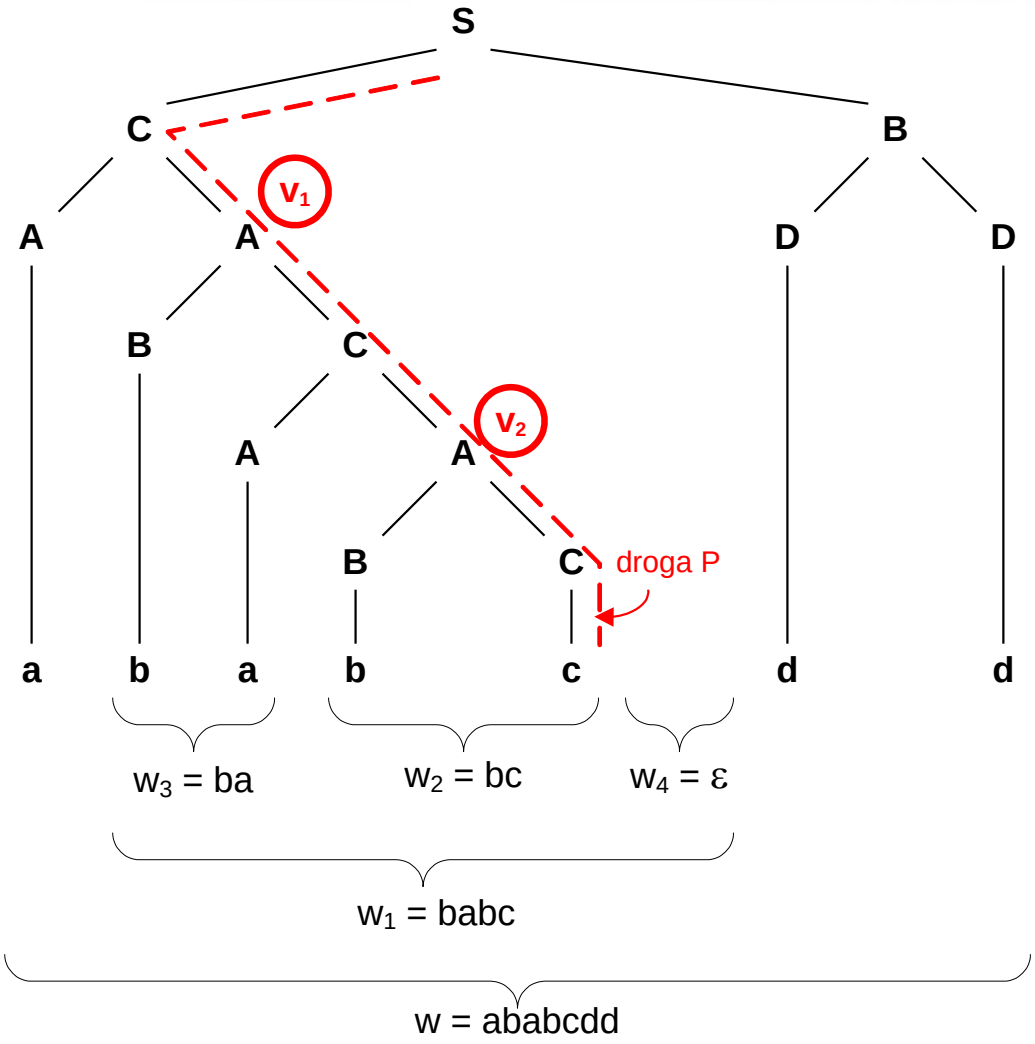
Niech G ma n symboli nieterminalnych oraz niech $k = 2^n$. Jeśli $w \in L(G)$ i $|w| \geq k$, to wobec faktu, że $|w| > 2^{n-1}$ oraz w świetle powyższego stwierdzenia o związku między długością słowa a długością drogi w drzewie rozbioru, każde drzewo rozbioru dla w musi zawierać drogę o długości co najmniej $n+1$. Ale taka droga zawiera co najmniej $n+2$ wierzchołków, z których wszystkie poza ostatnim są etykietowane nieterminalami gramatyki G. Tym samym musi istnieć jakiś nieterminal pojawiający się dwukrotnie na tej drodze. Niech P będzie drogą równie długą lub dłuższą niż jakakolwiek inna droga w rozważanym drzewie. Wtedy muszą istnieć dwa wierzchołki v_1 i v_2 leżące na tej drodze i spełniające warunki:

- Wierzchołki v_1 i v_2 mają tę samą etykietę, np. A.
- Wierzchołek v_1 jest bliższy korzeniowi niż v_2.
- Część drogi leżąca pomiędzy v_1 a liściem nie jest dłuższa niż $n+1$.

Warunek konieczny bezkontekstowości języka (4)

Aby się przekonać, że zawsze można znaleźć takie v_1 i v_2, wystarczy podążać drogą P od liścia ku górze, zapamiętując napotkane etykiety. Z pierwszych $n+2$ wierzchołków tylko liść ma etykietę będącą symbolem terminalnym. Pozostałe $n+1$ wierzchołków nie może być etykietowanych różnymi nieterminalami. Poddrzewo o korzeniu v_1 reprezentuje wyprowadzenie o długości co najwyżej 2^n. Jest tak, ponieważ w tym poddrzewie nie może istnieć droga o długości większej od $n+1$, gdyż P było drogą o największej długości w całym drzewie. Niech w_1 będzie koroną poddrzewa o wierzchołku v_1. Jeśli w_2 jest koroną poddrzewa o wierzchołku v_2, to w_1 możemy zapisać w postaci $w_1 = w_3 w_2 w_4$. Co więcej w_3 i w_4 nie mogą być równocześnie równe ε, gdyż pierwsza produkcja użyta w wyprowadzeniu w_1 musi mieć postać $A \rightarrow BC$ dla pewnych nieterminali B i C. Poddrzewo o wierzchołku v_2 musi być całkowicie zawarte w poddrzewie generowanym przez B lub w poddrzewie generowanym przez C. Wszystko to jest zilustrowane na rysunku na następnym slajdzie.

Warunek konieczny bezkontekstowości języka (5)



Warunek konieczny bezkontekstowości języka (6)

Wiemy już, że

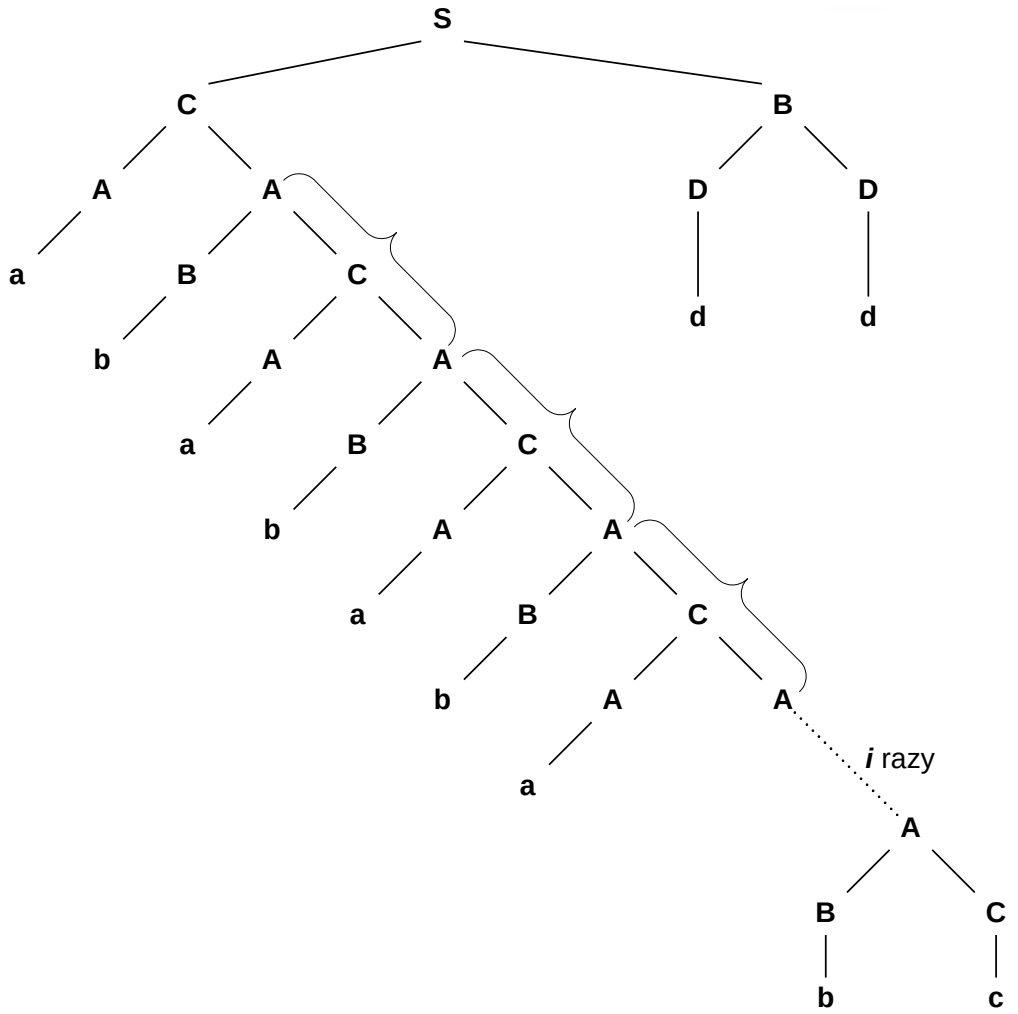
$$A \Rightarrow^* w_3 A w_4$$

oraz

$$A \Rightarrow^* w_2,$$

gdzie $|w_3 w_2 w_4| \leq 2^n = k$.

Ale stąd wynika, że $A \Rightarrow^* w_3^i w_2 w_4^i$ dla dowolnego $i \geq 0$. Łańcuch w można oczywiście zapisać w postaci $w = x w_3 w_2 w_4 z$ dla pewnych x i z. W celu zakończenia dowodu wystarczy przyjąć $w_3 = u$, $w_2 = y$ oraz $w_4 = v$, więc $w = xuyvz$.



Przykład

$L_1 = \{ a^i b^i c^i \mid i > 0 \}$ nie jest językiem bezkontekstowym. Przypuśćmy dla dowodu nie wprost, że L jest bezkontekstowy i niech k będzie stałą z lematu o rozrastaniu. Weźmy pod uwagę łańcuch $w=a^k b^k c^k$. Niech rozkład $w=xuyvz$ spełnia warunki lematu o rozrastaniu. Wtedy wobec $|uyv| \leq k$ łańcuch uv może zawierać co najwyżej dwa różne symbole. Co więcej, jeśli uv zawiera dwa różne symbole, to muszą one być symbolami kolejnymi, np. a i b. Jeśli uv zawiera wyłącznie symbole a, to xyz ma mniej symboli a niż symboli c oraz symboli b, czyli $xyz \notin L$ – sprzeczność. Postępujemy podobnie, jeśli uv składa się wyłącznie z symboli b lub wyłącznie z symboli c. Przypuśćmy teraz, że uv zawiera symbole a oraz symbole b. Jeżeli u lub v zawiera dwa różne symbole, to $xu^2yv^2z \notin L$. (Dla przykładu, jeśli u składa się z symboli a i b to xu^2yv^2z zawiera symbol b poprzedzający symbol a.) Jeśli zaś u zawiera tylko symbole a oraz v tylko symbole b, to wtedy xyz ma nadal mniej symboli a i symboli b niż symboli c, czyli znowu $xyz \notin L$. Podobna sprzeczność pojawia się w przypadku, gdy uv składa się z symboli b i symboli c. Ponieważ są to jedyne możliwości, to wnioskujemy, że L nie jest językiem bezkontekstowym.

Uwaga o wykorzystywaniu lematu o rozrastaniu

Przykład: $L_2 = \{ 0^i 1^i \mid i \geq 0 \}$ jest językiem bezkontekstowym (bo istnieje akceptujący go automat ze stosem $\rightarrow$ wcześniej był przykład).

Uwaga: Lemat o rozrastaniu języków bezkontekstowych można praktycznie wykorzystać jedynie wówczas, gdy zachodzi uzasadnione podejrzenie, że badany język nie należy do grupy języków bezkontekstowych. Twierdzenie odwrotne do lematu o rozrastaniu nie jest bowiem prawdziwe. W związku z tym, w przypadku chęci wykazania bezkontekstowości jakiegoś języka, pokazanie prawdziwości tezy lematu nic nam nie mówi o prawdziwości jego założenia, czyli nadal nie wiemy, czy badany język jest bezkontekstowy. Wówczas dla potwierdzenia przynależności języka do klasy języków bezkontekstowych należy skonstruować automat ze stosem akceptujący ten język lub zbudować generującą go bezkontekstową gramatykę.

Inny przykład

$L=\{a^i b^j c^i d^j \mid i\geq 1, j\geq 1\}$. Przypuśćmy, że L bezkontekstowy i niech k będzie stałą z lematu o rozrastaniu. Weźmy pod uwagę łańcuch $w=a^k b^k c^k d^k$. Niech rozkład $w=xuyvz$ spełnia warunki lematu o rozrastaniu. Wtedy wobec $|uyv|\leq k$ łańcuch uv może zawierać co najwyżej dwa różne symbole. Co więcej, jeśli uv zawiera dwa różne symbole, to muszą one być symbolami kolejnymi, np. a i b. Jeśli uv zawiera wyłącznie symbole a, to xyz ma mniej symboli a niż symboli c, czyli $xyz\notin L$ – sprzeczność. Postępujemy podobnie, jeśli uv składa się wyłącznie z symboli b, wyłącznie z symboli c lub wyłącznie z symboli d. Przypuśćmy teraz, że uv zawiera symbole a oraz symbole b. Jeżeli u lub v zawiera dwa różne symbole, to $xu^2yv^2z\notin L$. (Dla przykładu, jeśli u składa się z symboli a i b to xu^2yv^2z zawiera symbol b poprzedzający symbol a). Jeśli zaś u zawiera tylko symbole a oraz v tylko symbole b, to wtedy xyz ma nadal mniej symboli a niż symboli c, czyli znowu $xyz\notin L$. Podobna sprzeczność pojawia się w przypadku, gdy uv składa się z symboli b i symboli c lub z symboli c i symboli d. Ponieważ są to jedyne możliwości, to wnioskujemy, że L nie jest językiem bezkontekstowym.

Lemat Ogdena

Twierdzenie:

Jeżeli $L \in \mathcal{L}_{BK}$

to $(\exists k)$ (($w \in L \wedge$ w słowie w oznaczono dowolne k lub więcej pozycji jako wyróżnione) $\Rightarrow$ ($w = xuyvz \wedge$ łańcuch uv ma co najmniej jedną wyróżnioną pozycję $\wedge$ łańcuch uyv zawiera co najwyżej k wyróżnionych pozycji $\wedge$ $(\forall i \geq 0)$ $(xu^i yv^i z \in L)$)))

Lemat o rozrastaniu się języków bezkontekstowych jest szczególnym przypadkiem lematu Ogdena dla sytuacji, gdy wszystkie pozycje są pozycjami wyróżnionymi.

Przykład

$L=\{a^l b^m c^n \mid l\neq m,\ m\neq n,\ l\neq n\}$. Przypuśćmy, że L jest bezkontekstowy. Niech k będzie stałą z lematu Ogdena. Weźmy pod uwagę łańcuch $w=a^k b^{k+k!} c^{k+2k!}$. Załóżmy, że wyróżniamy pozycje symboli a, niech rozkład $w=xuyvz$ spełnia warunki lematu Ogdena. Jeżeli u lub v zawiera dwa różne symbole, to $xu^2yv^2z\notin L$. (Dla przykładu, jeśli u składa się z symboli a i b to xu^2yv^2z zawiera symbol b poprzedzający symbol a.) Jednak przynajmniej jedno spośród u i v musi zawierać symbole a, ponieważ tylko te symbole występują na wyróżnionych pozycjach. Zatem jeśli $v\in\{b\}^*$ lub $v\in\{c\}^*$, to u musi należeć do $\{a\}^+$. Jeżeli $v\in\{a\}^+$, to u musi należeć do $\{a\}^*$, gdyż inaczej jakiś symbol b lub c poprzedziłby symbol a. Rozważmy szczegółowo przypadek, gdy $v\in\{b\}^*$, a $u\in\{a\}^+$. (Pozostałe przypadki traktowane są w podobny sposób.) Niech $p=|u|$. Wtedy $1\leq p\leq k$, czyli p dzieli $k!$. Niech q będzie liczbą całkowitą, taką że $pq=k!$ Wtedy $w'=xu^{2q+1}yv^{2q+1}z\in L$. Ale $u^{2q+1}=a^{p(2q+1)}=a^{2pq+p}=a^{2k!+p}$. Ponieważ xyz zawiera dokładnie $k-p$ symboli a , to w' zawiera $2k!+p+(k-p)$ czyli $2k!+k$ symboli a, czyli tyle samo co symboli c, stąd $w'\notin L$ – sprzeczność. Podobna sprzeczność pojawia się w przypadku, gdy $v\in\{c\}^*$ lub $v\in\{a\}^+$. Zatem L nie jest językiem bezkontekstowym.

Własności zamkniętości języków bezkontekstowych (1)

Twierdzenie:

Języki bezkontekstowe są zamknięte ze względu na podstawienia, tzn. jeśli $L \in \mathcal{L}_{BK}$, $L \subseteq \Sigma^*$, dla każdego $a \in \Sigma$ język $L_a \in \mathcal{L}_{BK}$, $f(a)=L_a$, to $f(L) \in \mathcal{L}_{BK}$.

Szkic dowodu:

Niech $L \in \mathcal{L}_{BK}$, $L \subseteq \Sigma^*$, oraz niech dla każdego $a \in \Sigma$ język $L_a \in \mathcal{L}_{BK}$. Dalej niech będzie dana gramatyka G taka że $L=L(G)$ oraz dla każdego $a \in \Sigma$ gramatyki G_a, przy czym $L_a=L(G_a)$. Bez utraty ogólności można założyć, że zbiory nieterminali gramatyki G i gramatyk G_a są wzajemnie parami rozłączne. Zbudujmy nową gramatykę G' w następujący sposób: nieterminalami G' są wszystkie nieterminale gramatyki G i gramatyk G_a; symbole terminalne G' to symbole końcowe gramatyk G_a. Symbol początkowy G' pokrywa się z symbolem początkowym G. Produkcje G' to wszystkie produkcje gramatyk G_a, a także wszystkie produkcje utworzone z produkcji gramatyki G poprzez zastąpienie w nich każdego wystąpienia pewnego symbolu $a \in \Sigma$ symbolem początkowym S_a gramatyki G_a. Czytelnik zechce pokazać, że gramatyka G' generuje $f(L)$, przy czym dla każdego $a \in \Sigma$ jest $f(a)=L_a$.

Własności zamkniętości języków bezkontekstowych (2)

Przykład:

Niech L będzie zbiorem słów nad alfabetem {a,b} o jednakowej liczbie symboli a i b, $L_a = \{ 0^n1^n \mid n > 0 \}$, $L_b = \{ ww^R \mid w \in \{0,1\}^* \}$. Wybieramy gramatyki G, G_a i G_b.

G: $S \rightarrow aSbS \mid bSaS \mid \varepsilon$

G_a: $S_a \rightarrow 0S_a1 \mid 01$

G_b: $S_b \rightarrow 0S_b0 \mid 1S_b1 \mid \varepsilon$

Jeśli f jest podstawieniem takim, że $f(a) = L_a$ i $f(b) = L_b$, to f(L) jest generowany przez gramatykę G'

G': $S \rightarrow S_aSS_bS \mid S_bSS_aS \mid \varepsilon$

$S_a \rightarrow 0S_a1 \mid 01$

$S_b \rightarrow 0S_b0 \mid 1S_b1 \mid \varepsilon$

Własności zamkniętości języków bezkontekstowych (3)

Twierdzenie:

Języki bezkontekstowe są zamknięte ze względu na sumę teoriomnogościową, złożenie oraz domknięcie Kleene'ego, czyli jeśli $L_1 \in L_{BK}$, $L_2 \in L_{BK}$, to

$L_1 \cup L_2 \in L_{BK}$

$L_1 L_2 \in L_{BK}$

$L_1^* \in L_{BK}$

Szkic dowodu

Należy zauważyć, że ponieważ {a,b}, {ab} oraz {a}* są językami bezkontekstowymi, to zamkniętość klasy języków bezkontekstowych ze względu na podstawienia implikuje zamkniętość ze względu na sumę teoriomnogościową, złożenie i domknięcie Kleene'ego. Suma teoriomnogościowa L_a i L_b to po prostu wynik podstawienia L_a i L_b do {a,b}; podobnie $L_a L_b$ i L^* są wynikami podstawień odpowiednio do {ab} i {a}*. Zatem twierdzenie powyższe można traktować jako wniosek z twierdzenia wcześniejszego.

Własności zamkniętości języków bezkontekstowych (4)

Twierdzenie:

Języki bezkontekstowe są zamknięte ze względu na homomorfizmy, tzn. jeśli $L \in \mathcal{L}_{BK}$, $h:\Sigma \rightarrow \Delta^*$, $L \subseteq \Sigma^*$, h jest homomorfizmem, to $h(L) \in \mathcal{L}_{BK}$.

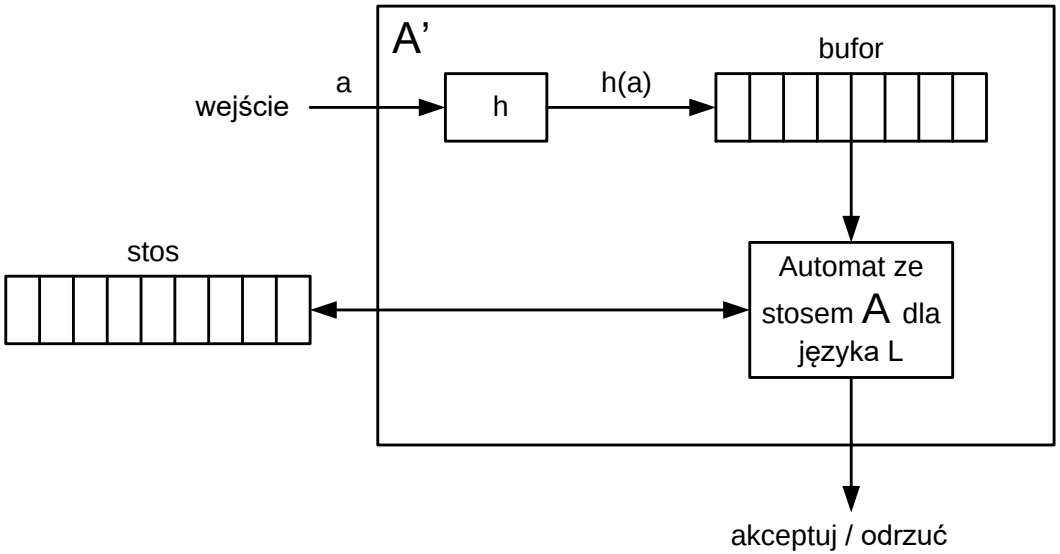
Ponieważ homomorfizm jest szczególnym rodzajem podstawienia, to twierdzenie powyższe także może być uważane za wniosek z twierdzenia o zamkniętości języków bezkontekstowych ze względu na podstawienie.

Twierdzenie:

Języki bezkontekstowe są zamknięte ze względu na przeciwobrazy homomorficzne, tzn. jeśli $L \in \mathcal{L}_{BK}$, $h:\Delta \rightarrow \Sigma^*$, h jest homomorfizmem, $L \subseteq \Sigma^*$, to $h^{-1}(L) \in \mathcal{L}_{BK}$.

Własności zamkniętości języków bezkontekstowych (5)

Szkic konstrukcji z dowodu twierdzenia o zamkniętości języków bezkontekstowych ze względu na przeciwobrazy homomorficzne



Konstruujemy nowy automat ze stosem A’ symulujący działanie automatu ze stosem A pracującego na wejściu, którym jest bufor zawierający obrazy homomorficzne kolejnych symboli wejściowych analizowanego słowa. Po odczytaniu symbolu *a* z wejścia, *h(a)* jest umieszczane w buforze, symbole z bufora są używane po jednym na raz i wprowadzane do symulowanego automatu A, dopiero gdy bufor jest pusty, automat A’ odczytuje następny symbol z wejścia i stosuje do niego homomorfizm.

Własności zamkniętości języków bezkontekstowych (6)

Twierdzenie:

Języki bezkontekstowe nie są zamknięte ze względu na przecięcie (iloczyn teoriomnogościowy), różnicę oraz dopełnienie, tzn. istnieją takie $L_1 \in \mathcal{L}_{BK}$, $L_2 \in \mathcal{L}_{BK}$, że

$L_1 \cap L_2 \notin \mathcal{L}_{BK}$, $L_1 - L_2 \notin \mathcal{L}_{BK}$ oraz $\Sigma^* - L_1 \notin \mathcal{L}_{BK}$

Twierdzenie:

Języki bezkontekstowe są zamknięte ze względu na przecięcie (iloczyn teoriomnogościowy), iloraz przez język regularny oraz różnicę z językiem regularnym, tzn. jeśli $L \in \mathcal{L}_{BK}$, $R \in \mathcal{L}_{RG}$, to

$L \cap R \in \mathcal{L}_{BK}$, $L/R \in \mathcal{L}_{BK}$ oraz $L - R \in \mathcal{L}_{BK}$

Własności zamkniętości języków bezkontekstowych (7)

Przykład:

Rozważymy dwa języki bezkontekstowe:

$L_1 = \{ a^i b^i c^j \mid i > 0; j > 0 \}$ generowany przez gramatykę bezkontekstową:

$S \rightarrow AB$

$A \rightarrow aAb \mid ab$

$B \rightarrow cB \mid c$

oraz $L_2 = \{ a^i b^j c^j \mid i > 0; j > 0 \}$ generowany przez gramatykę bezkontekstową:

$S \rightarrow AB$

$A \rightarrow aA \mid a$

$B \rightarrow bBc \mid bc$

Język $L = L_1 \cap L_2 = \{ a^i b^i c^i \mid i > 0 \}$ nie jest językiem bezkontekstowym, jak to pokazano w jednym z poprzednich przykładów.

Przykład – idealne przetasowanie (1)

Idealne przetasowanie języków L_A i L_B definiujemy następująco:

Niech $L_A, L_B \subseteq \Sigma^*$

$$Perfect\_Shuffle(L_A, L_B) = \{ w \mid w = a_1b_1a_2b_2\ldots a_kb_k,$$
$$a_1a_2\ldots a_k \in L_A,\ b_1b_2\ldots b_k \in L_B,\ \text{dla } a_i, b_i \in \Sigma \}$$

Czy klasa języków bezkontekstowych jest zamknięta ze względu na idealne przetasowanie?

Przykład – idealne przetasowanie (2)

Nie, weźmy bowiem:

$L_A = \{ 0^k 1^k \mid k \geq 1 \}$; $L_B = \{ 2^k 3^{3k} \mid k \geq 1 \}$

Języki L_A i L_B są bezkontekstowe, bowiem generujące je gramatyki są następujące:

$S_A \rightarrow 0\, S_A\, 1 \mid 01$ $S_B \rightarrow 2\, S_B\, 333 \mid 2333$

Mamy: $\Sigma = \{0, 1, 2, 3\}$

$L = \text{Perfect\_Shuffle}(L_A, L_B) = \{ (02)^k (03)^k (13)^{2k} \mid k \geq 1 \}$

Niech $\Gamma = \{a, b, c\}$, rozważymy homomorfizm $h: \Gamma \rightarrow \Sigma^*$

$h(a) = 02$

$h(b) = 03$

$h(c) = 1313$

$h^{-1}(L) = \{a^k b^k c^k \mid k \geq 1 \}$, co nie jest językiem bezkontekstowym.

Przykład – idealne przetasowanie (3)

Ponieważ języki L_A, L_B są bezkontekstowe, klasa języków bezkontekstowych jest zamknięta ze względu na przeciwobrazy homomorficzne, zaś
L = Perfect_Shuffle(L_A, L_B) nie jest językiem bezkontekstowym, więc klasa języków bezkontekstowych nie jest zamknięta ze względu na idealne przetasowanie.

Problem przynależności słowa do języka generowanego przez gramatykę bezkontekstową (1)

Jednym z najważniejszych, rozważanych już wcześniej problemów, jest zagadnienie: dla danej gramatyki bezkontekstowej $G = <V, \Sigma, P, S>$ i łańcucha $x \in \Sigma^*$, czy łańcuch ten należy do języka generowanego przez gramatykę G?

Prosty, ale nieefektywny algorytm rozwiązania tego problemu jest oparty na przekształceniu G do postaci normalnej Greibach $G' = <V', \Sigma, P', S>$ generującej $L(G') - \{\varepsilon\}$. Ponieważ algorytm usuwania ε–produkcji sprawdza, czy $\varepsilon \in L(G)$, więc możemy nie uwzględniać przypadku, gdy $x = \varepsilon$. Załóżmy więc, że $x \neq \varepsilon$, czyli $x \in L(G')$ wtedy i tylko wtedy, gdy $x \in L(G)$. Ponieważ każda produkcja gramatyki w postaci normalnej Greibach dodaje dokładnie jeden symbol terminalny do łańcucha generowanego w trakcie wyprowadzenia, to wiemy, że jeżeli x ma wyprowadzenie w G', to ma on wyprowadzenie o dokładnie $|x|$ krokach. Jeśli żaden z nieterminali gramatyki G' nie ma więcej niż k produkcji, to istnieje co najwyżej $k^{|x|}$ lewostronnych wyprowadzeń łańcuchów o długości $|x|$. Możemy je sprawdzić wszystkie w systematyczny sposób, ale może to wymagać czasu wykładniczego względem $|x|$.

Problem przynależności słowa do języka generowanego przez gramatykę bezkontekstową (2)

Przykład:

Rozważmy gramatykę *G*:

$S \rightarrow AA \mid a$
$A \rightarrow SS \mid b$

Po przekształceniu do postaci normalnej Greibach otrzymujemy gramatykę *G'*:

$S \rightarrow aSA'A \mid aSA \mid a \mid bA'A \mid bA$
$A \rightarrow aSA' \mid aS \mid bA' \mid b$
$A' \rightarrow aSA'SA' \mid aSSA' \mid bA'SA' \mid bSA' \mid aSA'S \mid aSS \mid bA'S \mid bS$

Analizujemy słowo *aab*. Obok w formie drzewa systematycznie wypisano wszystkie wyprowadzenia form zdaniowych rozpoczynających się od *aab* oraz od wszystkich przedrostków słowa *aab*. Jak widać $aab \in L(G')$, gdyż udało się znaleźć wyprowadzenie $S \Rightarrow^* aab$ w gramatyce *G'*.

- **S**
  - aSA'A
    - aaSA'AAa
      - aabA'AA'AA'A
      - aabAA'AA'A
    - aaSAA'A
      - aabA'AAA'A
      - aabAAA'A
    - aaA'A
      - aabA'SA'S
      - aabSA'A
      - aabA'SA
      - aabSA
  - **aSA**
    - aaSA'AA
      - aabA'AA'AA
      - aabAA'AA
    - aaSAA
      - aabA'AAA
      - aabAAA
    - **aaA**
      - aabA'
      - **aab**
  - a

Algorytm Cocke'a–Youngera–Kasamiego (1)

Podamy teraz jeden z algorytmów wymagających czasu sześciennego względem $|x|$, tzw. algorytm Cocke'a–Youngera–Kasamiego oparty na idei programowania dynamicznego. Algorytm ten bazuje na gramatyce w postaci normalnej Chomsky'ego i – podobnie jak poprzedni algorytm "sprawdzania systematycznego" oparty o gramatykę w postaci normalnej Greibach – nie nakłada żadnych wymagań na gramatykę, w szczególności nie żąda, aby była to gramatyka jednoznaczna.

Zakładamy, że dany jest łańcuch x o długości $n > 0$ i gramatyka G w postaci normalnej Chomsky'ego. Dla dowolnych i, j oraz dowolnego nieterminala A sprawdzamy, czy $A \Rightarrow^* x_{i,j}$, gdzie $x_{i,j}$ jest podłańcuchem o długości j, rozpoczynającym się od i–tej pozycji łańcucha x. Stosujemy indukcję po j. Dla $j = 1$, $A \Rightarrow^* x_{i,j}$ wtedy i tylko wtedy, gdy $A \rightarrow x_{i,j}$ jest produkcją, gdyż $x_{i,j}$ jest łańcuchem o długości 1.

Algorytm Cocke'a–Youngera–Kasamiego (2)

Przejdźmy teraz do wyższych wartości j. Jeśli $j > 1$, to $A \Rightarrow^* x_{i,j}$ wtedy i tylko wtedy, gdy istnieje pewna produkcja $A \rightarrow BC$ oraz pewne k, $1 \leq k < j$, takie że B wyprowadza pierwsze k symboli $x_{i,j}$, a C – ostatnie $j - k$ symboli $x_{i,j}$. Innymi słowy, $B \Rightarrow^* x_{i,k}$ oraz $C \Rightarrow^* x_{i+k,j-k}$. Ponieważ zarówno k, jak i $j - k$ jest mniejsze od j, to wiemy już, czy każde z tych dwóch ostatnich wyprowadzeń istnieje. Możemy więc stwierdzić, czy $A \Rightarrow^* x_{i,j}$. Na koniec, po osiągnięciu $j = n$ możemy rozstrzygnąć, czy $S \Rightarrow^* x_{1,n}$. Ale $x_{1,n} = x$, czyli x należy do $L(G)$ wtedy i tylko wtedy, gdy $S \Rightarrow^* x_{1,n}$. Aby sformułować algorytm Cocke'a–Youngera–Kasamiego w sposób bardziej precyzyjny, oznaczmy symbolem $V_{i,j}$ zbiór tych nieterminali A, dla których $A \Rightarrow^* x_{i,j}$. Zauważmy, że $1 \leq i \leq n - j + 1$, gdyż nie istnieje łańcuch o długości większej od $n - j + 1$, rozpoczynający się od i–tej pozycji łańcucha x.

Algorytm Cocke'a–Youngera–Kasamiego (3)

Metoda konstrukcji zbiorów $V_{i,j}$:

for i := 1 to n do

$V_{i,1} := \{ A \mid (A \to a) \in P,$ i–tym symbolem łańcucha x jest a $\}$;

for j := 2 to n do

for i := 1 to n – j + 1 do

begin

$V_{i,j} := \emptyset$;

for k := 1 to j – 1 do

$V_{i,j} := V_{i,j} \cup \{ A \mid (A \to BC) \in P,\ B \in V_{i,k}$ i $C \in V_{i+k,j-k} \}$;

end;

Algorytm Cocke'a–Youngera–Kasamiego (4)

Przykład:

Rozważamy gramatykę o produkcjach:

$S \rightarrow AB \mid BC$

$A \rightarrow BA \mid a$

$B \rightarrow CC \mid b$

$C \rightarrow AB \mid a$

oraz łańcuch wejściowy aabbab. Poniżej pokazano tabelę zbiorów $V_{i,j}$.

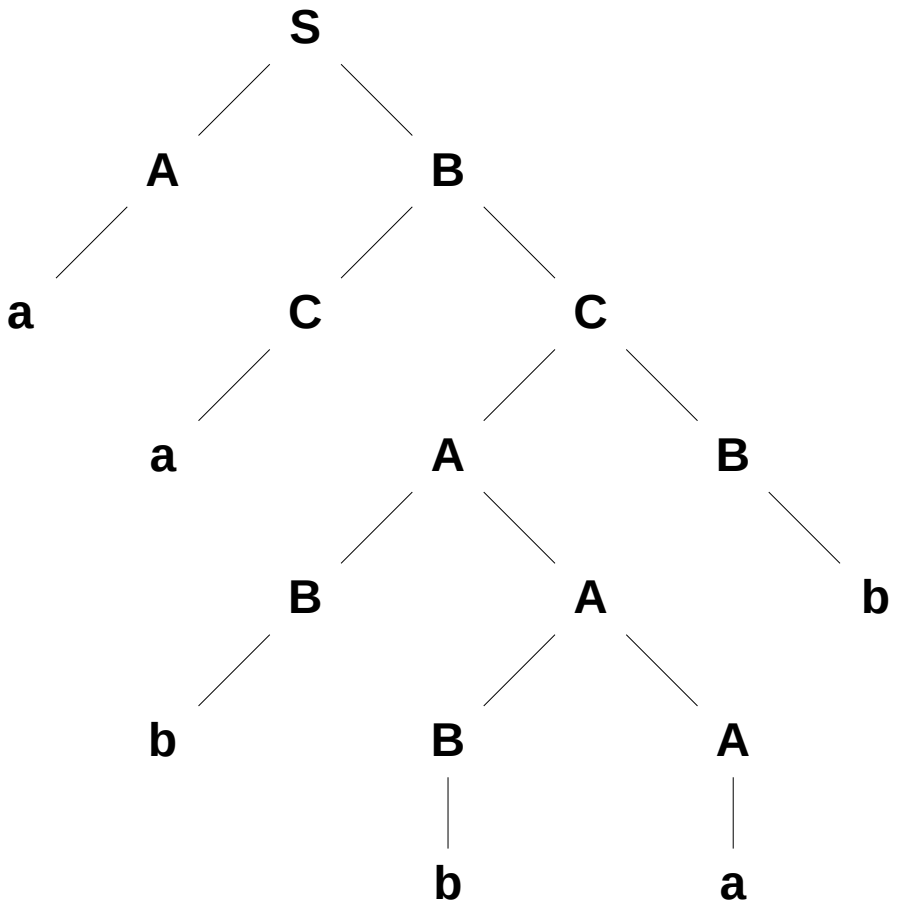
		a	a	b	b	a	b
				i	→		
		1	2	3	4	5	6
	1	A,C	A,C	B	B	A,C	B
	2	B	S,C	∅	A,S	S,C	
j	3	B	∅	A	S,C		
↓	4	∅	∅	S,C			
	5	A	B				
	6	S,C					

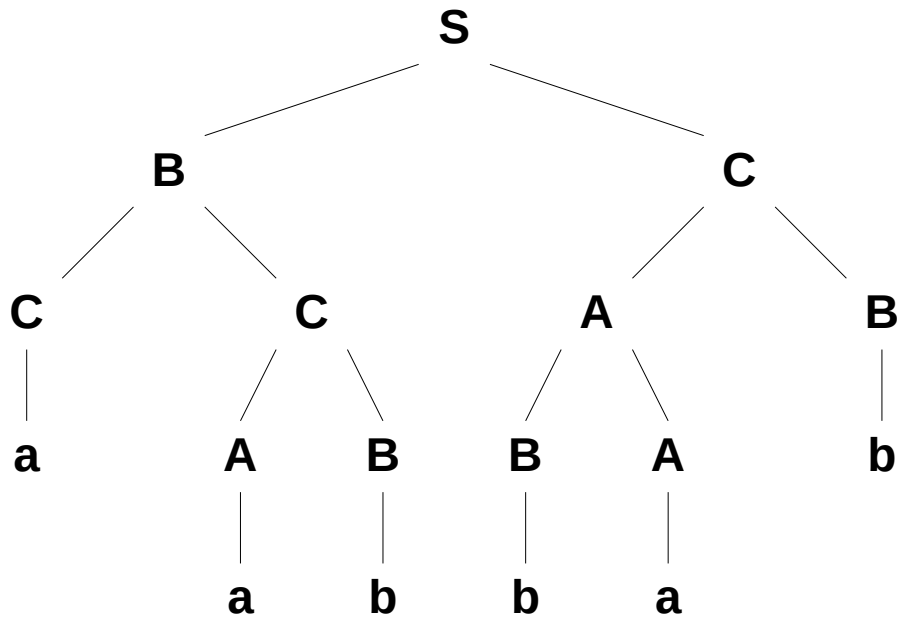
Algorytm Cocke'a–Youngera–Kasamiego (5)

		a	a	b	b	a	b
				i	→		
		1	2	3	4	5	6
	1	A,C	A,C	B	B	A,C	B
	2	B	S,C	∅	A,S	S,C	
j	3	B	∅	A	S,C		
↓	4	∅	∅	S,C			
	5	A	B				
	6	S,C					

Ponieważ S jest elementem $V_{1,6}$, to łańcuch *aabbab* należy do języka generowanego przez rozważaną gramatykę. Gramatyka z tego przykładu nie jest jednoznaczna. Na podstawie powyższej tabeli (analizując ją wstecz) można zbudować drzewa rozbioru dla analizowanego łańcucha. Na przykład, uwzględniając, że $A \in V_{1,1}$ i $B \in V_{2,5}$ (pierwsza produkcja $S \rightarrow AB$, A wyprowadza łańcuch o długości 1, B wyprowadza łańcuch o długości 5, itd.) otrzymujemy lewe drzewo z następnego rysunku, zaś biorąc pod uwagę, że $B \in V_{1,3}$ i $C \in V_{4,3}$ (pierwsza produkcja $S \rightarrow BC$, B wyprowadza łańcuch o długości 3, C wyprowadza łańcuch o długości 3, itd.) dostaniemy prawe drzewo.

Algorytm Cocke'a–Youngera–Kasamiego (6)





Ściśle wieloznaczne języki bezkontekstowe

- Język bezkontekstowy nazywamy ściśle wieloznacznym, jeśli każda gramatyka bezkontekstowa generująca ten język jest wieloznaczna.
- Przykładem bezkontekstowego języka ściśle wieloznacznego jest język:

$$L=\{a^n b^n c^m d^m \mid n,m \geq 1\} \cup \{a^n b^m c^m d^n \mid n,m \geq 1\}$$

Przykładowa gramatyka tego języka ma postać:

$S \rightarrow A \mid CD$

$A \rightarrow aAd \mid aBd$

$B \rightarrow bBc \mid bc$

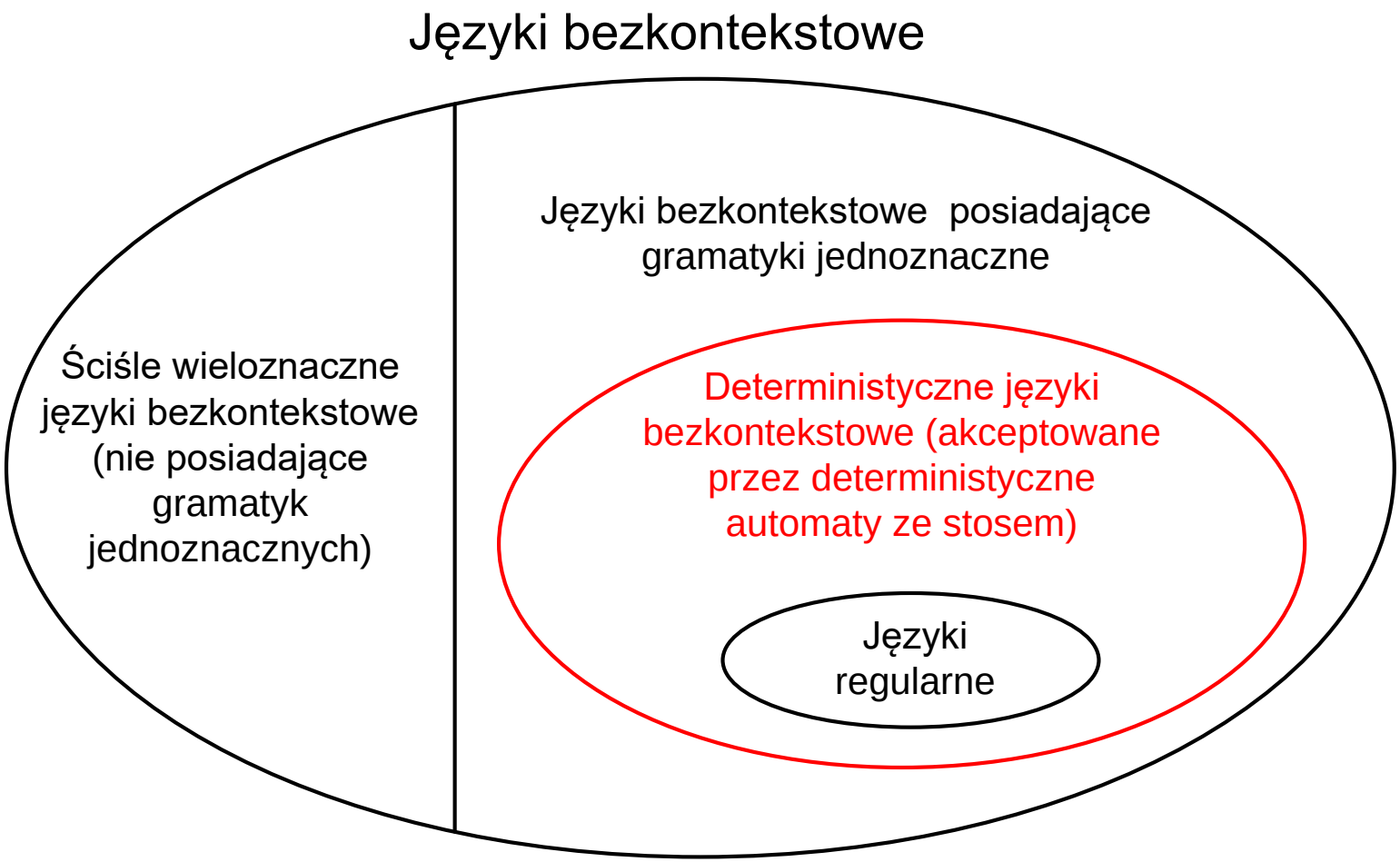
$C \rightarrow aCb \mid ab$

$D \rightarrow cDd \mid cd$

Deterministyczne języki bezkontekstowe (1)

- Deterministyczne języki bezkontekstowe to języki bezkontekstowe akceptowane przez deterministyczne automaty ze stosem.
- Każdy deterministyczny język bezkontekstowy jest generowany przez jednoznaczną gramatykę bezkontekstową.
- Istnieją języki generowane przez jednoznaczne gramatyki bezkontekstowe, które nie są deterministycznymi językami bezkontekstowymi. Przykładem może być język $\{xx^R \mid x \in \{a,b\}^*\}$ posiadający jednoznaczną gramatykę: $S \rightarrow aSa \mid bSb \mid \varepsilon$

Deterministyczne języki bezkontekstowe (2)



Deterministyczne języki bezkontekstowe (3)

Deterministyczne języki bezkontekstowe są zamknięte ze względu na:

- dopełnienie,
- iloraz przez język regularny,
- przeciwobraz homomorficzny,
- przecięcie z językiem regularnym.

Deterministyczne języki bezkontekstowe nie są zamknięte ze względu na:

- podstawienie,
- homomorfizm,
- przecięcie,
- sumę teoriomnogościową,
- złożenie,
- domknięcie Kleene'a.

Problemy decyzyjne dla języków bezkontekstowych

Rozważamy język *L* reprezentowany przez gramatykę *G* w postaci normalnej Chomsky'ego bez symboli nadmiarowych. Istnieją algorytmy pozwalające rozstrzygnąć:

- czy język L jest pusty,
- czy język L jest skończony,
- czy język L jest nieskończony.

Problemy decyzyjne dla języków bezkontekstowych

Rozważamy język *L* reprezentowany przez gramatykę *G* w postaci normalnej Chomsky'ego bez symboli nadmiarowych. Istnieją algorytmy pozwalające rozstrzygnąć:

- **czy język *L* jest pusty?**

Język *L* jest niepusty, gdy symbol początkowy *S* gramatyki *G* jest symbolem użytecznym (czyli symbol początkowy generuje jakikolwiek ciąg symboli terminalnych). Język jest niepusty wtedy i tylko wtedy, gdy *S* generuje jakiś ciąg symboli terminalnych.

Staranny algorytm wykrywania symboli użytecznych zajmuje $O(n)$ czasu, gdzie n jest rozmiarem gramatyki.

Problemy decyzyjne dla języków bezkontekstowych

Rozważamy język *L* reprezentowany przez gramatykę *G* w postaci normalnej Chomsky'ego bez symboli nadmiarowych. Istnieją algorytmy pozwalające rozstrzygnąć:

- **czy język *L* jest skończony?**

Budujemy graf skierowany gramatyki *G* w PNC, którego wierzchołkami są nieterminale gramatyki. Jeśli gramatyka zawiera produkcję $A \rightarrow BC$ lub $A \rightarrow CB$, to w grafie są krawędzie: z *A* do *B* i z *A* do *C*. Język *L* jest skończony, wtedy i tylko wtedy, gdy graf nie zawiera cykli.

Badanie acykliczności grafu skierowanego zajmuje $O(\max(\#V,\#E))$ czasu, gdzie $\#V$ i $\#E$ są odpowiednio liczbą wierzchołków i liczbą krawędzi w grafie.

Problemy decyzyjne dla języków bezkontekstowych

Rozważamy język *L* reprezentowany przez gramatykę *G* w postaci normalnej Chomsky'ego bez symboli nadmiarowych. Istnieją algorytmy pozwalające rozstrzygnąć:

- **czy język *L* jest nieskończony?**

Język *L* jest nieskończony, wtedy i tylko wtedy, gdy opisany wcześniej graf nie jest grafem acyklicznym.

Problemy decyzyjne dla języków bezkontekstowych

Rozważamy język *L* reprezentowany przez gramatykę *G* w postaci normalnej Chomsky'ego bez symboli nadmiarowych. Istnieje algorytm pozwalający podać

- **górne ograniczenie długości słowa w skończonym języku bezkontekstowym *L***

Język *L* jest skończony wtedy i tylko wtedy, gdy opisany wcześniej graf jest grafem acyklicznym. Znajdujemy długość najdłuższej ścieżki w tym grafie rozpoczynającej się w węźle odpowiadającym symbolowi początkowemu gramatyki. Jeśli ta długość wynosi *r*, to język generowany przez gramatykę nie zawiera słowa dłuższego niż 2^r.

Znajdowanie najdłuższej ścieżki w grafie skierowanym, rozpoczynającej się w danym źródle, zajmuje $O(\max(\#V,\#E))$ czasu, gdzie $\#V$ i $\#E$ są odpowiednio liczbą wierzchołków i liczbą krawędzi w grafie.

Problemy decyzyjne dla języków bezkontekstowych

Konwersje reprezentacji języków bezkontekstowych

- Przekształcenie gramatyki bezkontekstowej o rozmiarze n na równoważny jej (silnie nieterministyczny) automat ze stosem wymaga czasu $O(n)$ i daje automat o wielkości $O(n)$.
- Przekształcenie automatu ze stosem o rozmiarze n na równoważną mu gramatykę bezkontekstową wymaga czasu $O(n^3)$ i daje gramatykę o wielkości co najwyżej $O(n^3)$.
- Dla danej gramatyki bezkontekstowej o rozmiarze n można zbudować równoważną jej gramatykę w postaci normalnej Chomsky'ego w czasie $O(n^2)$, a wynikowa gramatyka ma rozmiar $O(n^2)$.

Problemy decyzyjne dla języków bezkontekstowych

Przekształcenia gramatyk języków bezkontekstowych

- Testowanie pustości języka bezkontekstowego drogą rozstrzygnięcia, czy symbol początkowy generującej go gramatyki o rozmiarze n jest symbolem użytecznym, staranny algorytm wykorzystujący odpowiednie struktury danych może być wykonany w czasie $O(n)$.
- Testowanie, czy język bezkontekstowy zawiera słowo puste przeprowadzone drogą rozstrzygnięcia, czy symbol początkowy gramatyki jest symbolem zerowalnym (wyprowadza słowo puste), także jest wykonalne w czasie $O(n)$.
- Problem znajdowania symboli osiągalnych gramatyki bezkontekstowej zrealizowany starannym algorytmem także wymaga czasu liniowego $O(n)$.
- Eliminacja ε–produkcji w danej gramatyce bezkontekstowej o wielkości n wymaga czasu $O(n^2)$, a wynikowa gramatyka na rozmiar $O(n^2)$.
- Eliminacja produkcji łańcuchowych (typu $A \rightarrow B$) w danej gramatyce bezkontekstowej o wielkości n wymaga czasu $O(n^2)$, a wynikowa gramatyka na rozmiar $O(n^2)$.

Problemy decyzyjne dla języków bezkontekstowych

Testowanie przynależności słowa do języka bezkontekstowego.

- Testowanie, czy dane słowo o długości *n* należy do języka generowanego przez daną gramatykę bezkontekstową w postaci normalnej Greibach można wykonać w czasie wykładniczym w stosunku do długości słowa (w technologii *top-down*).
- Testowanie, czy dane słowo o długości *n* należy do języka generowanego przez daną gramatykę bezkontekstową w postaci normalnej Chomsky'ego można wykonać w czasie $O(n^3)$ stosując oparty o programowanie dynamiczne algorytm Cocke'a–Youngera–Kasamiego (w technologii *bottom-up*).