Języki formalne i automaty 11-18.XII.2020

Gramatyki i języki bezkontekstowe $\mathcal{L}_2$

Produkcje bezkontekstowe: $X \to \omega\,, \qquad X \in V, \quad \omega \in (A \cup V)^*$

Produkcje kontekstowe: $\alpha X \beta \to \alpha \omega \beta, \qquad \omega \neq \lambda$

Przykłady

- $L = \{a^n b^n : \ n \geq 0\}$ $\qquad S \to aSb \mid \lambda$

- $L = \{a^m b^n : \ m \neq n\}$
  $L = L_1 \cup L_2$, gdzie $L_1 = \{a^m b^n : \ m > n\}$, $L_2 = \{a^m b^n : \ m < n\}$.

$$\begin{array}{rcl} S_0 & \to & AS \mid SB \\ S & \to & aSb \mid \lambda \\ A & \to & aA \mid a \\ B & \to & bB \mid b\,. \end{array}$$

- $L = \{\omega \in A^* : \ |\omega|_a = |\omega|_b\}$ $\qquad S \to SS \mid aSb \mid bSa \mid \lambda$

  $S \to SS \mid (\,S\,) \mid \lambda$

- Język prostych wyrażeń arytmetycznych

$$\begin{array}{rcl} S & \to & S + S \mid S * S \mid X \\ X & \to & x \mid y \mid z \mid (S) \end{array}$$

Przeanalizować wyprowadzenia

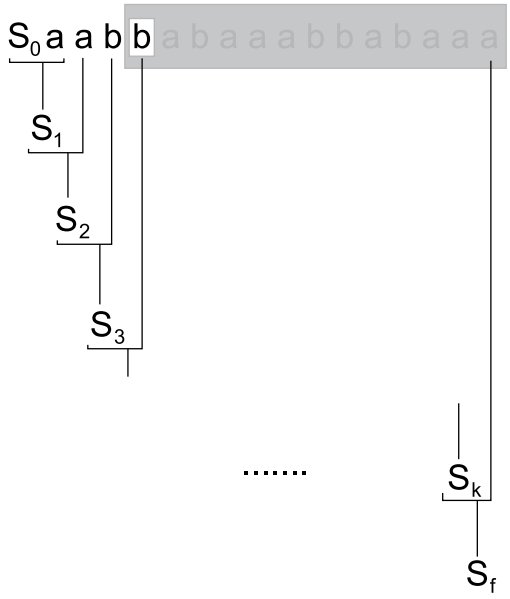
$$aaabbbbb, \quad aaaaabbb, \quad (()()), \quad (()())(), \quad x * (y + z), \quad x * y + z$$

Przykłady:

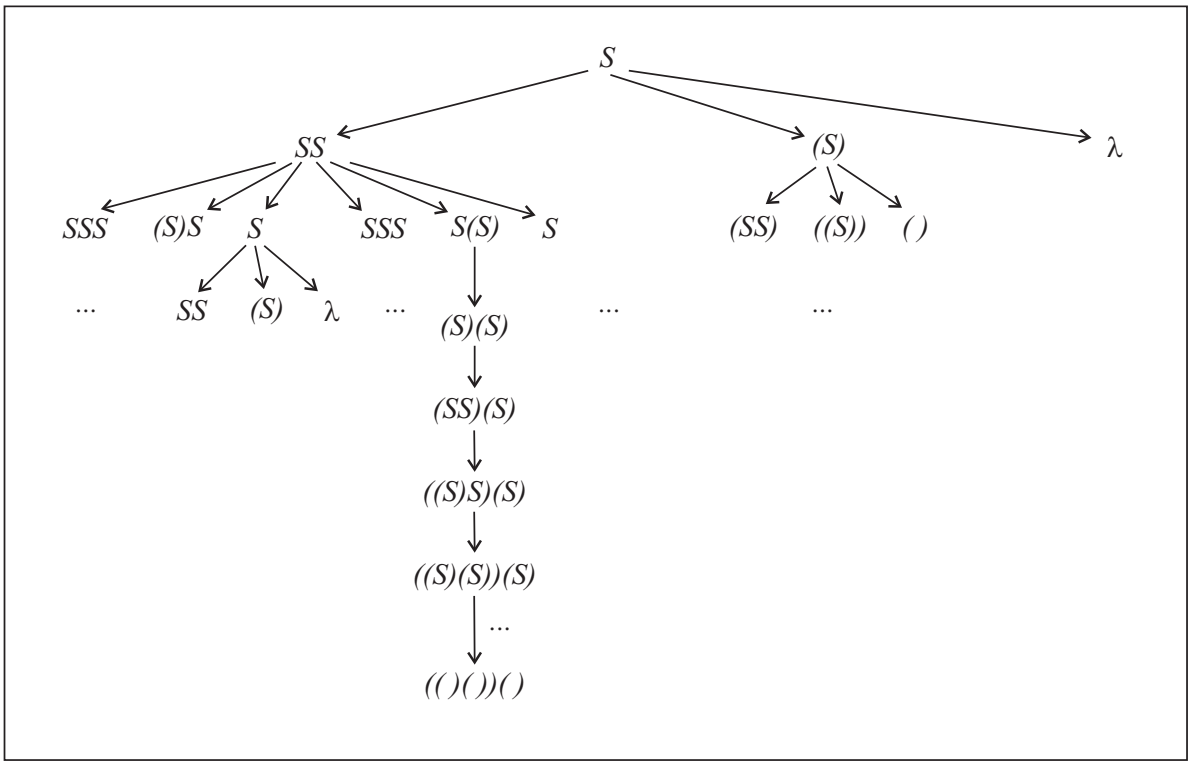
$L = \{a^n b^m : \ 2n \leq m \leq 3n\}, \qquad L = \{\omega : \ |\omega|_a \neq |\omega|_b\}, \qquad L = \{\omega \overleftarrow{\omega} : \ \omega \in A^*\}^c$

Jak sprawdzić mając daną gramatykę czy dane słowo $\alpha \in L(G)$?

Analiza bottom-up i top-down



$$S \to SS \mid (\,S\,) \mid \lambda$$



Szacowanie złożoności przeszukania drzewa wyprowazdeń

– gramatyka bez pustych i jednostkowych produkcji: $X \to \lambda, \quad X \to Y$

– $X \to \beta, \quad |\beta| > 1$ za wyjątkiem $X \to a$ (produkcje nieskracające)

– liczba wszystkich produkcji gramatyki p

Wówczas

– liczba węzłów na k-tym piętrze drzewa wyprowadzeń $\leq p^k$

– wszystkie $\alpha \in A^*$ długości $\leq n$ znajdują się w piętrach $\leq 2n$

– przeszukanie drzewa by sprawdzić czy $\alpha \in L(G)$ (o długości $\leq n$)

$$1 + p + p^2 + \cdots + p^{2n} \;=\; O(p^{2n+1})$$

By usprawnić wyprowadzanie napisów, szukamy dogodnej postaci gramatyki:

– bez pustych i jednostkowych produkcji
– bez wielu różnych wyprowadzeń tego samego napisu
– uporządkowanie procesu wyprowadzania: wyprowadzenia lewostronne

Jeszcze raz język prostych wyrażeń arytmetycznych

$$\begin{array}{lcl} S & \to & S + S \mid S * S \mid X \\ X & \to & x \mid y \mid z \mid (S) \end{array} \qquad \begin{array}{lcl} S & \to & T \mid T + S \\ T & \to & F \mid F * T \\ F & \to & x \mid y \mid z \mid (S) \end{array}$$

$$S \Rightarrow S + S \Rightarrow S + S * S \Rightarrow S + S * S + S \Rightarrow \ldots$$

$$S \Rightarrow S * S \Rightarrow S + S * S \Rightarrow S + S * S + S \Rightarrow \ldots$$

$$S \Rightarrow S + S \Rightarrow S + S + S \Rightarrow S + S * S + S \Rightarrow \ldots$$

W drugiej gramatyce

$$\begin{array}{rcl} S & \Rightarrow & T + S \Rightarrow F + S \Rightarrow x + S \Rightarrow x + T + S \Rightarrow x + F * T + S \\ & \Rightarrow & x + y * T + S \Rightarrow x + y * F + S \Rightarrow x + y * z + S \Rightarrow x + y * z + T \\ & \Rightarrow & x + y * z + F \Rightarrow x + y * z + z\,. \end{array}$$

Definicja *Mówimy że gramatyka jest jednoznaczna, jeśli dla dowolnego napisu $\alpha \in A^*$ istnieje co najwyżej jedno wyprowadzenie lewostronne z S.*

Istnieją języki bezkontekstowe *wewnętrznie niejednoznaczne*

Postać normalna gramatyki bezkontekstowej

Definicja *Mówimy, że bezkontekstowa gramatyka G jest dana w normalnej postaci Chomsky'ego, jeśli wszystkie jej produkcje mają formę*

$$X \to YZ \qquad lub \qquad X \to a\,,$$

gdzie $X, Y, Z \in V$ oraz $a \in A$.

Twierdzenie *Każdą gramatykę bezkontekstową $G = (A, V, S, \Pi)$ taką, że $\lambda \notin L(G)$, można przekształcić do równoważnej postaci normalnej Chomsky'ego $G' = (A, V', S, \Pi')$.*

Przekształcenie gramatyki do postaci normalnej Chomsky'ego

1) Usuwamy z gramatyki produkcje puste.
2) Eliminujemy produkcje jednostkowe.
3) Usuwamy symbole i produkcje bezużyteczne.
4) Jeśli $\lambda \in L(G)$, uzupełniamy G' o dodatkową produkcję $S \to \lambda$. Jest to jedyna pusta podukcja dopuszczalna w G'.
5) Przekształcenie produkcji gramatyki do postaci normalnej Chomsky'ego:

$$X \to YZ, \qquad X \to a\,.$$

Nagrania mp4 z przykładami redukcji gramatyk do postaci Chomsky'ego – p. strona przedmiotu, link z datą 18.XII.20.

Algorytm Cooke'a, Youngera i Kasamiego (CYK): dla zadanej gramatyki bezkontekstowej G w postaci normalnej Chomsky'ego i słowa α zbadać czy $\alpha \in L(G)$.

Niech $\alpha = a_1 a_2 \dots a_n$. Oznaczmy przez $\alpha_{ij} = a_i a_{i+1} \dots a_j$. Działanie algorytmu polega na stopniowym tworzeniu zbiorów symboli nieterminalnych

$$V_{ij} \;=\; \{X \in V : \;\; X \;\Rightarrow^* \; \alpha_{ij}\}\,,$$

kolejno od

$$V_{ii} \;=\; \{X \in V : \;\; X \to a_i \;\text{ jest produkcją w G}\}$$

aż do zbioru

$$V_{1n} \;=\; \{X \in V : \;\; X \;\Rightarrow^* \alpha_{1n} = \alpha\}\,.$$

Wystarczy wówczas sprawdzić czy $S \in V_{1n}$.

Niech $X \in V_{ij},\;\; i < j,\;\;$ czyli $\;X \Rightarrow^* \alpha_{ij} = a_i a_{i+1} \dots a_j$.
Wyprowadzenie to zaczyna się od zastosowania pewnej produkcji $X \to YZ$,

$$X \;\Rightarrow\; YZ \;\Rightarrow^* \;\alpha_{ij} \;=\; a_i a_{i+1} \dots a_k a_{k+1} \dots a_j\,,$$

czyli

$$Y \;\Rightarrow^* \; a_i a_{i+1} \dots a_k \;=\; \alpha_{ik} \qquad \text{oraz} \qquad Z \;\Rightarrow^* \; a_{k+1} \dots a_j \;=\; \alpha_{k+1\,j}$$

dla pewnego $i \le k < j$. Stąd $Y \in V_{ik}$ i $Z \in V_{k+1\,j}$. Odwrotnie

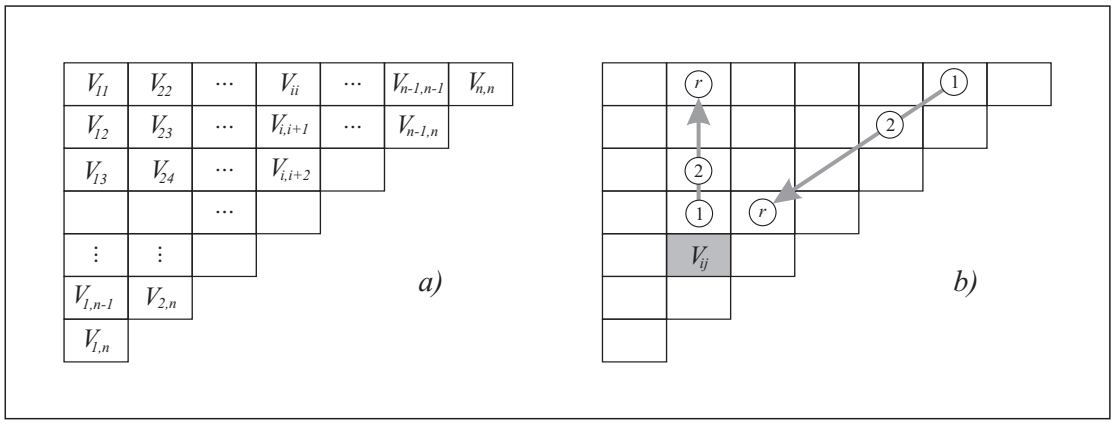
$$X \in V_{ij} \qquad \Leftrightarrow \qquad \exists\, i \le k < j \quad X \to YZ \in \Pi, \quad Y \in V_{ik} \quad \text{i} \quad Z \in V_{k+1\,j}\,.$$

Konstrukcja zbiorów V_{ij}

$$\begin{array}{ccc} Y \in V_{ik} & & V_{k+1\,j} \ni Z \\ \searrow & & \swarrow \\ & V_{ij} & \\ & X \to YZ & \end{array}$$

kolejno dla $k = i, i+1, \dots j-1$. A więc

$$V_{ij} \;=\; \bigcup_{i \le k < j} \{X : \;\; X \to YZ \in \Pi \quad \text{oraz} \quad Y \in V_{ik}, \;\; Z \in V_{k+1\,j}\}$$



Przykład. Weźmy jako przykład gramatykę $S \to () \mid (S) \mid SS$ po przekształceniu jej do postaci Chomsky'ego

$$\begin{array}{lcl} S & \to & LR \mid LQ \mid SS \\ Q & \to & SR \\ L & \to & (\\ R & \to &) \end{array}$$

Tworzenie pierwszego wiersza tabelki V_{ij} dla napisu $(()())()$

(	(	)	(	)	)	(	)
L	L	R	L	R	R	L	R

Tworzenie drugiego wiersza: $V_{i\,i+1}$ to zbiór tych symboli X, które są lewymi stronami produkcji $X \to YZ$, gdzie $X \in V_{ii}$, a $Y \in V_{i+1\,i+1}$.

(	(	)	(	)	)	(	)
L	L	R	L	R	R	L	R
$\emptyset$	S	$\emptyset$	S	$\emptyset$	$\emptyset$	S	

Pomocnicza tabela podglądu: $TP[Y,Z] = \{X \in V : \ X \to YZ \in \Pi\}$. Dla naszego przykładu

	S	L	R	Q
S	S	ø	Q	ø
L	ø	ø	S	S
R	ø	ø	ø	ø
Q	ø	ø	ø	ø

Końcowa postać tabeli

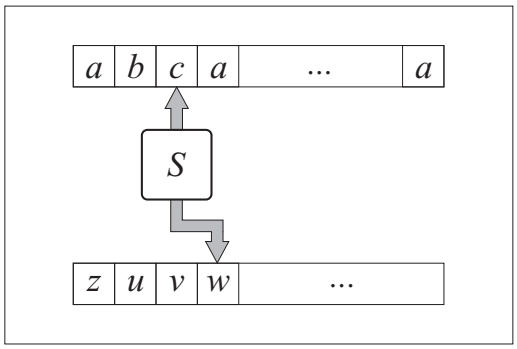
(	(	)	(	)	)	(	)
L	L	R	L	R	R	L	R
$\emptyset$	S	$\emptyset$	S	$\emptyset$	$\emptyset$	S	
$\emptyset$	$\emptyset$	$\emptyset$	Q	$\emptyset$	$\emptyset$		
$\emptyset$	S	$\emptyset$	$\emptyset$	$\emptyset$			
$\emptyset$	Q	$\emptyset$	$\emptyset$				
S	$\emptyset$	$\emptyset$					
$\emptyset$	$\emptyset$						
S							

Niech $\alpha = a_1 a_2 \dots a_n$ oraz niech tablica $TP[Y, Z]$ zawiera informację o lewych stronach produkcji $X \to YZ$. Prócz tego gramatyka zawiera produkcje terminalne $X_i \to a_i$. Pseudokod CYK:

```
function LP(U, W){
    V = empty;
    for (Y ∈ U)
        for (Z ∈ W)
            V = V ∪ TP[Y,Z];
    return V;
}

for (i=1,...,n) V_ii = {X_i};

for (l=1,...,n-1)
    for (i=1,...,n-l)
        j=i+l;
        V_ij = empty;
        for (k=i,...,j-1)
            V_ij = V_ij ∪ LP(V_ik, V_{k+1 j});

if (S ∈ V_1n) print("OK");
```

Złożoność wynosi $O(n^3)$ – potrójna pętla.

Automaty ze stosem



Definicja: *Niedeterministyczny automat ze stosem (NAS) opisany jest przez siedem obiektów*

$$M \;=\; (A, V, z, \Sigma, s_0, S_F, \pi)\,,$$

gdzie:

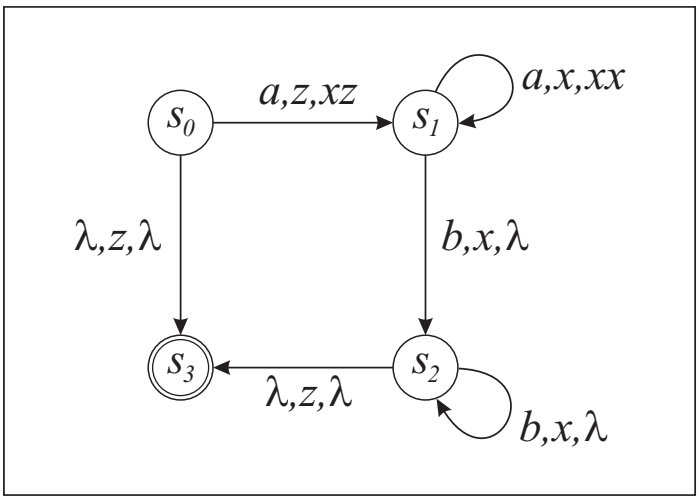
A *jest alfabetem wejściowym*
V *jest alfabetem stosu*
$z \in V$ *jest wyróżnionym symbolem początkowym stosu*
Σ *jest zbiorem stanów automatu,* $\sigma = \{s_0, s_1, \ldots, s_q\}$
$s_0 \in \Sigma$ *jest wyróżnionym stanem startowym*
$S_F \subseteq \Sigma$ *jest zbiorem stanów końcowych (akceptujących)*
π *jest relacją przejścia,* $\pi : \Sigma \times (A \cup \{\lambda\}) \times V \;\to\; \mathcal{F}(\Sigma \times V^*)\,.$

Relacja przejścia π(stan, litera, symbol stosu) = zbiór par (stan, napis do stosu)

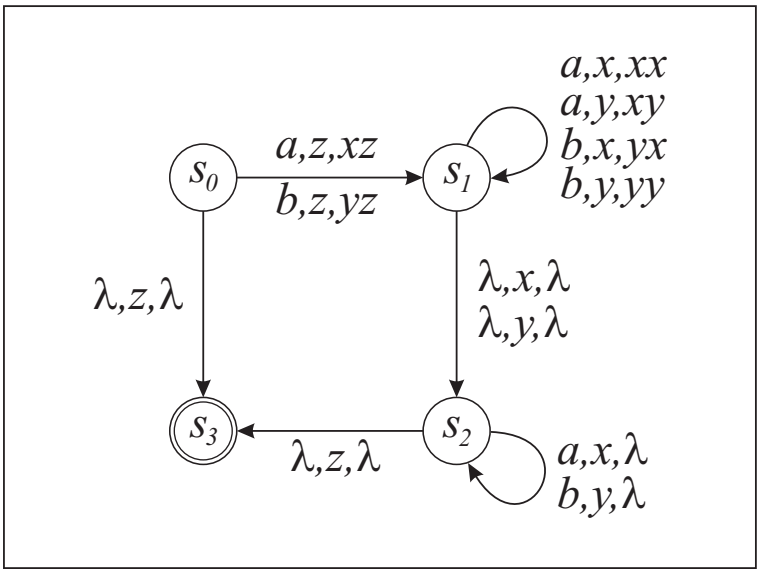
$$\pi(s_i, a, x) \;=\; \{\,(s_{j_1}, \gamma_1),\ (s_{j_2}, \gamma_2),\ \ldots,\ (s_{j_k}, \gamma_k)\,\}\,.$$

- Jeśli czytana litera to a, stan automatu s_i oraz znak na wierzchu stosu to x, automat przesuwa się w prawo do nastepnej litery, zmienia stan na jeden z możliwych stanów s_j, usuwa znak x ze stosu, a w jego miejsce zapisuje odpowiadający s_j ciąg symboli stosu γ_j, być może λ.
- Przejście puste $\pi(s_i, \lambda, x) = (s_j, \gamma_j)$ – bez odczytywania kolejnej litery i ruchu głowicy czytającej.
- Jeśli stos jest pusty (nie ma x do odczytania), automat zatrzymuje się nie mogąc wykonać kolejnego ruchu (stąd specjalny symbol z jako początkowa zawartość stosu).
- Napis wejściowy zostanie zaakceptowany, jeśli po przeczytaniu ostatniej jego litery automat znajdzie się w jednym ze stanów końcowych. Zawartość stosu w tym momencie jest nieistotna.

Przykład: Automat akceptujący język $L = \{a^n b^n : \quad n \geq 0\}$



Przykład: Automat akceptujący język $L = \{\alpha \overleftarrow{\alpha} : \quad \alpha \in A^*\}$



Definicja: *Automat ze stosem jest deterministyczny (DAS) jeśli relacja przejścia π jest częściową funkcją, a więc posiada następujące własności:*

(i) dla każdej trójki $(s, a, x) \in \Sigma \times A \times V$ istnieje co najwyżej jedna para $(s', \gamma) \in \Sigma \times V^*$ taka, że $\pi(s, a, x) = (s', \gamma)$;

(ii) jeśli puste przejście $\pi(s, \lambda, x)$ jest określone dla pewnych s i x, wtedy $\pi(s, a, x)$ nie może być określone dla jakiejkolwiek litery a.

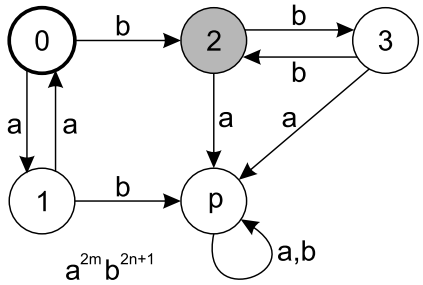
Twierdzenie: *Klasa $\mathcal{L}_2$ języków bezkontekstowych (tj. opisywanych przez gramatyki bezkontekstowe) jest identyczna z klasą języków akceptowanych przez niedeterminiestyczne automaty ze stosem $\mathcal{L}_{\mathrm{NAS}}$. Języki klasy $\mathcal{L}_{\mathrm{DAS}}$ stanowią właściwy podzbiór $\mathcal{L}_2$*

$$\mathcal{L}_{\mathrm{DAS}} \subsetneq \mathcal{L}_{\mathrm{NAS}} = \mathcal{L}_2 .$$

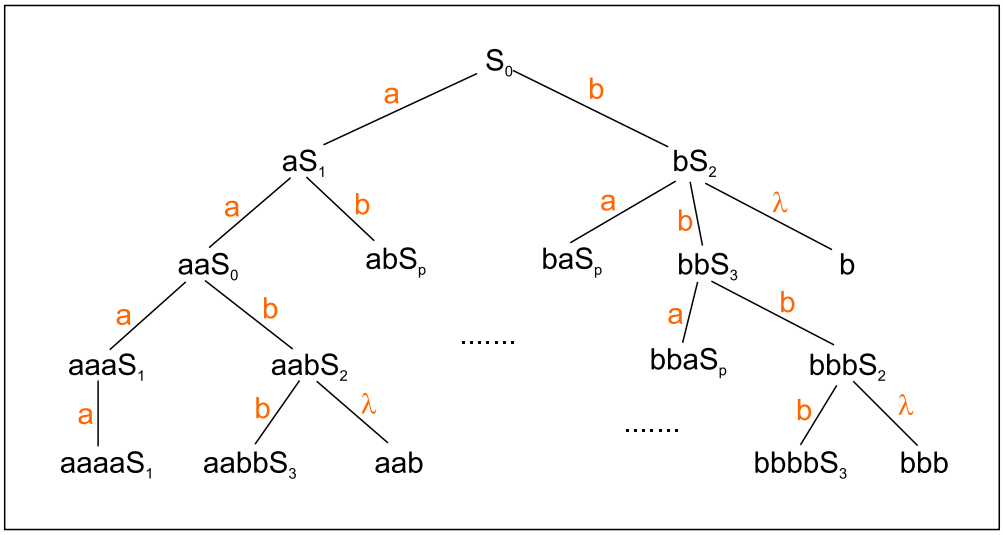
Przyklad: Język niedeterministyczny

$$L = L_1 \cup L_2 = \{a^n b^n : n \geq 0\} \cup \{a^n b^{2n} : n \geq 0\} .$$

Po przeczytaniu ostatniej litery a nadal nie wiadomo czy przetwarzane słowo może pochodzic z L_1 czy z L_2. Decyzja automatu co do wyboru dalszej ścieżki musi być podjęta niedeterministycznie.



$$
\begin{aligned}
S_0 &\rightarrow aS_1 \mid bS_2 \\
S_1 &\rightarrow aS_0 \mid bS_p \\
S_2 &\rightarrow aS_p \mid bS_3 \mid \lambda \\
S_3 &\rightarrow aS_p \mid bS_2 \\
S_p &\rightarrow aS_p \mid bS_p
\end{aligned}
$$



Języki formalne i automaty 4.I.2022

Gramatyki i języki bezkontekstowe $\mathcal{L}_2$

Twierdzenie: *Klasa $\mathcal{L}_2$ języków bezkontekstowych (tj. opisywanych przez gramatyki bezkontekstowe) jest identyczna z klasą języków akceptowanych przez niedeterminiestyczne automaty ze stosem $\mathcal{L}_{\text{NAS}}$. Języki klasy $\mathcal{L}_{\text{DAS}}$ stanowią właściwy podzbiór $\mathcal{L}_2$*

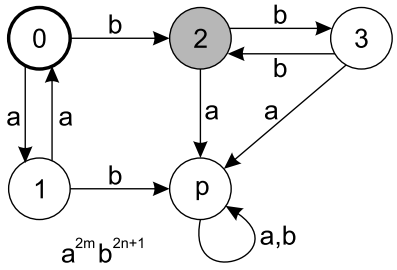
$$\mathcal{L}_{\text{DAS}} \subsetneq \mathcal{L}_{\text{NAS}} = \mathcal{L}_2 .$$

Przyklad: $L = L_1 \cup L_2 = \{a^n b^n : n \geq 0\} \cup \{a^n b^{2n} : n \geq 0\}$.

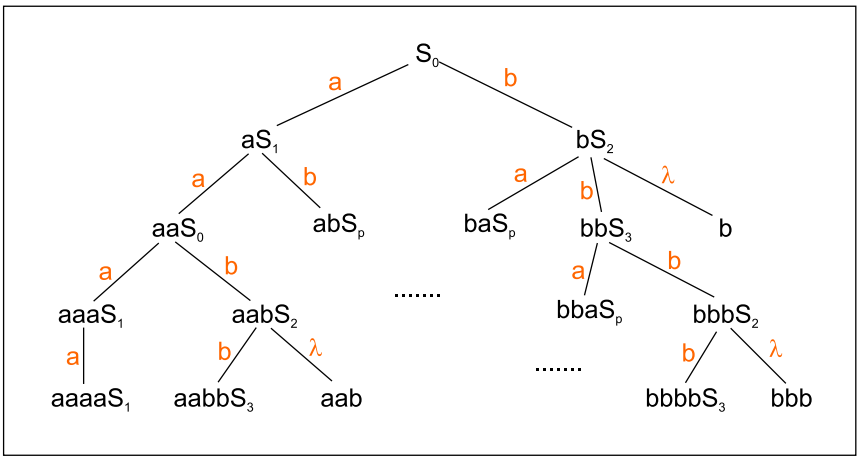
Po przeczytaniu ostatniej litery a nadal nie wiadomo czy przetwarzane słowo może pochodzic z L_1 czy z L_2. Decyzja automatu co do wyboru dalszej ścieżki musi być podjęta niedeterministycznie.

Niedeterminizm jest źródłem niższej efektywności algorytmów weryfikacji czy $\alpha \in L$. W praktycznych zastosowaniach (składnia języków programowania) wybieramy struktury deterministyczne.

Przykład: Gramatyka regularna na bazie automatu deterministycznego



$$\begin{aligned}
S_0 &\to aS_1 \mid bS_2 \\
S_1 &\to aS_0 \mid bS_p \\
S_2 &\to aS_p \mid bS_3 \mid \lambda \\
S_3 &\to aS_p \mid bS_2 \\
S_p &\to aS_p \mid bS_p
\end{aligned}$$



Specjalna klasa gramatyk **s-gramatyki** (simple)

Definicja: *G jest s-gramatyką, jeśli wszystkie jej produkcje mają postać*

$$X \to a\,Y_1 \ldots Y_m, \qquad \text{gdzie} \quad a \in A, \quad Y_i \in V, \quad m \geq 0\,,$$

przy czym dla każdej pary X i a istnieje co najwyżej jedna taka produkcja.

Produkcje s-gramatyki można zapisać w tablicy podglądu:

$$T[X, a] = Y_1 \ldots Y_m \qquad \text{lub} \qquad T[X, a] = \emptyset$$

Niech $\alpha = a_1 a_2 \ldots a_n$, tworzymy wyprowadzenie lewostronne zaczynając od symbolu S.

$$\begin{aligned} S &\underset{T[S,a_1] = Y_1 \ldots Y_m}{\Rightarrow} a_1 Y_1 \ldots Y_m \\ &\underset{T[Y_1,a_2] = Z_1 \ldots Z_k}{\Rightarrow} a_1 a_2 Z_1 \ldots Z_k Y_2 \ldots Y_m \\ &\underset{T[Z_1,a_3] = \ldots}{\Rightarrow} a_1 a_2 a_3 \ldots \end{aligned}$$

Podgląd jednego kolejnego symbolu wejściowego napisu pozwala na jednoznaczne ustalenie kolejnej produkcji w wyprowadzeniu (lub jej braku).

Definicja: *Gramatyka bezkontekstowa $G = (A, V, S, \Pi)$ jest typu $LL(k)$, jeśli dla każdej pary wyprowadzeń lewostronnych postaci*

$$\begin{aligned} S &\Rightarrow^* \alpha_1 X \omega_1 \Rightarrow \alpha_1 \beta_1 \omega_1 \Rightarrow^* \alpha_1 \alpha_2 \\ S &\Rightarrow^* \alpha_1 X \omega_2 \Rightarrow \alpha_1 \beta_2 \omega_2 \Rightarrow^* \alpha_1 \alpha_3\,, \end{aligned}$$

gdzie $\alpha_i \in A^*$, $\beta_i, \omega_i \in (A \cup V)^*$, równość co najwyżej k początkowych liter w α_2 i α_3 pociąga za sobą równość $\beta_1 = \beta_2$.

Przykłady

$G = \{S \to aSB|aB,\ B \to b\}$ nie jest s-gramatyką, bo obydwie produkcje dla S rozpoczynają się literą a. G jest gramatyką typu $LL(2)$:

	aa	ab	ba	bb
S	aSB	aB	$\emptyset$	$\emptyset$

Istnieje również s-gramatyka:

$$\begin{aligned} S &\to aA \\ A &\to aAB \,|\, b \\ B &\to b \end{aligned}$$

$G = \{S \to ()|(S)|SS\}$ nie jest typu $LL(k)$ dla żadnego k. Przy analizowaniu napisów o długości przekraczającej 2 mamy do wyboru dwie produkcje: $S \to (S)$ oraz $S \to SS$. Skanowany aktualnie symbol (ani k następnych) nie podpowiada, której z nich powinniśmy użyć. Weźmy bowiem słowa

$$((\ldots(\,)\ldots)) \qquad \text{albo} \qquad ((\ldots(\,)\ldots))(\,)\,.$$

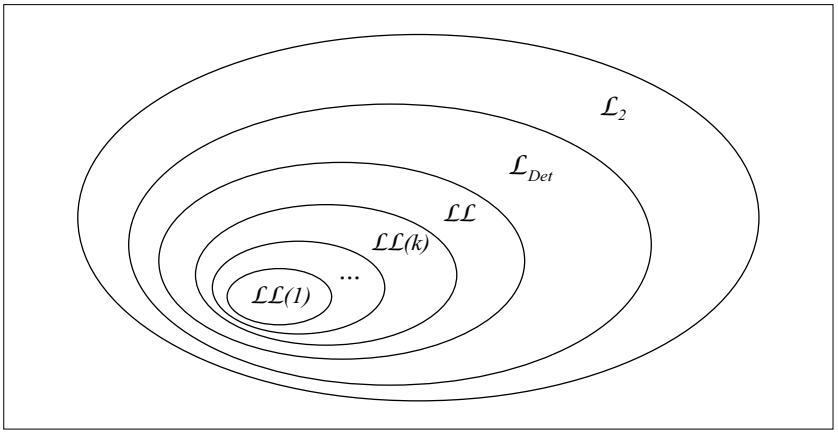
Wyprowadzenie pierwszego z nich polega na iteracyjnym użyciu produkcji $S \to (S)$, natomiast wyprowadzenie drugiego *musi* zaczynać się od $S \to SS$. Decyzję co do tego, której produkcji użyć w wyprowadzeniu jako pierwszej można podjąć na podstawie podglądu *nie mniej* niż $2n$ początkowych symboli wejściowych. Ponieważ jednak w naszej gramatyce możemy wyprowadzać takie napisy dla dowolnie dużych n, nie może być ona typu $LL(k)$ dla jakiegokolwiek ustalonego k.

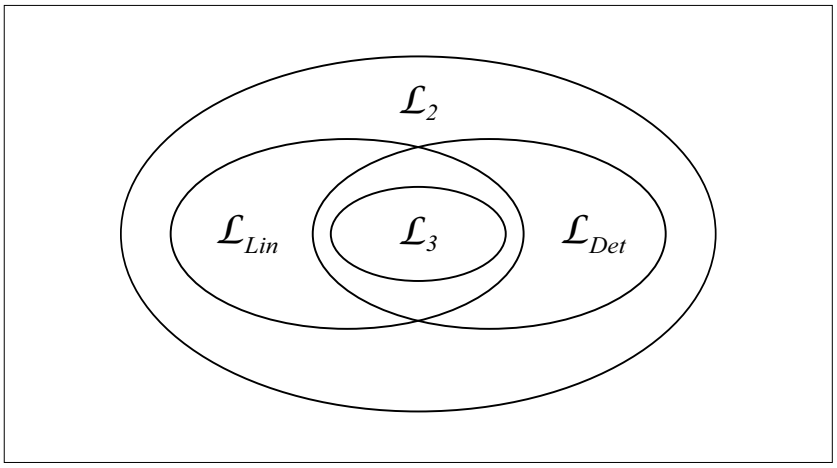
Ten sam język posiada gramatykę typu $LL(1)$: $G = \{S \to (S)S|\lambda\}$. Wyprowadzenie (())():

$$S \overset{(}{\Rightarrow} (S)S \overset{(}{\Rightarrow} ((S)S)S \overset{)}{\Rightarrow} (()S)S \overset{)}{\Rightarrow} (())S \overset{(}{\Rightarrow} (())(S)S \overset{)}{\Rightarrow} (())()S \overset{\lambda}{\Rightarrow} (())()\,.$$

Przeanalizować samodzielnie **Przykład 4.12** str. 105 z gramatyką $LL(1)$ dla wyrażeń arytmetycznych.

Klasy języków deterministyczych





Lematy o pompowaniu dla języków bezkontekstowych

Na początek języki liniowe:

$$X \to \alpha Y \beta, \quad X \to \alpha, \qquad \alpha, \beta \in A^*$$

Zakładamy przy tym, że produkcje są nieskracające i niejednostkowe (α, β nie mogą być jednocześnie puste). Niech N będzie liczbą symboli nieterminalnych w G, $V = \{S = X_1, X_2, \ldots, X_N\}$, a p maksimum długości prawych stron produkcji. Weźmy słowo $\omega \in L(G)$ o długości $m > pN$. Zatem jego wyprowadzenie musi zawierać co najmniej $N + 1$ kroków

$$\begin{aligned} S = X_1 \;&\Rightarrow\; \alpha_1 X_{i_2} \eta_1 \;\Rightarrow\; \alpha_1 \alpha_2 X_{i_3} \eta_2 \eta_1 \;\Rightarrow \ldots \Rightarrow\; \alpha_1 \ldots \alpha_N X_{i_{N+1}} \eta_N \ldots \eta_1 \\ &\Rightarrow\; \alpha_1 \ldots \alpha_N \gamma \eta_N \ldots \eta_1 \;=\; \omega\,. \end{aligned}$$

Stąd wniosek, że jeden z symboli X_i występuje w takim wyprowadzeniu co najmniej 2 razy

$$S \;\Rightarrow^*\; \alpha X_i \eta \;\Rightarrow^* \alpha\beta X_i \delta\eta \;\Rightarrow^*\; \alpha\beta\gamma\delta\eta\,,$$

a więc

$$X_i \;\Rightarrow^*\; \beta X_i \delta \qquad \text{oraz} \qquad X_i \;\Rightarrow^* \gamma\,.$$

Mamy zatem także

$$X_i \;\Rightarrow^*\; \beta^k \gamma \delta^k \qquad \text{czyli} \qquad \omega_k = \alpha\beta^k\gamma\delta^k\eta \in L(G) \quad \forall k \geq 0.$$

Lemat o pompowaniu dla jęz. liniowych

Niech L będzie nieskończonym językiem liniowym. Istnieje wówczas stała $m > 0$ (zależna jedynie od L) taka, że dowolne słowo $\omega \in L$ o długości co najmniej m można rozłożyć na części $\omega = \alpha\beta\gamma\delta\eta$ w taki sposób, że

(i) $|\alpha\beta\delta\eta| \;\leq\; m$,
(ii) $|\beta\delta| \;\geq\; 1$,

a przy tym wszystkie bez wyjątku słowa $\omega_k = \alpha\beta^k\gamma\delta^k\eta$ dla $k = 0, 1, 2, \ldots$ są także elementami L.

Przykład. $L = \{\omega \in \{a,b\}^* : \ |\omega|_a = |\omega|_b\}$ nie jest liniowy. Załóżmy nie wprost, że istnieje generująca go liniowa gramatyka. Zgodnie z Lematem dla L istnieje stała m wyznaczająca długość słów ω, dla których zawsze możliwe jest pompowanie $\omega_k \in L$.

Weźmy $\omega = a^m b^{2m} a^m$. Zgodnie z lematem $\omega = \alpha\beta\gamma\delta\eta$, gdzie $|\alpha\beta| + |\delta\eta| \leq m$, a więc muszą być zbudowane wyłącznie z liter a, natomiast (ii) zapewnia, że pompowane części β i δ zawierają co najmniej jedną literę a. Część centralna słowa zbudowana z liter b ukryta jest w podciągu γ i nie podlega pompowaniu. Tak więc liczba liter a w słowach ω_k wzrasta, podczas gdy liczba liter b pozostaje niezmieniona. Tym samym pompowanie tworzy słowa spoza L. Otrzymaliśmy sprzeczność z tezą lematu, a zatem założenie o liniowości języka L musiało być fałszywe.

Lemat o pompowaniu dla języków $\mathcal{L}_2$

Niech L będzie nieskończonym językiem bezkontekstowym. Istnieje wówczas stała $m > 0$ (zależna jedynie od L) taka, że dowolne słowo $\omega \in L$ o długości co najmniej m można rozłożyć na części $\omega = \alpha\beta\gamma\delta\eta$ w taki sposób, że

(i) $|\beta\gamma\delta| \leq m$,
(ii) $|\beta\delta| \geq 1$,

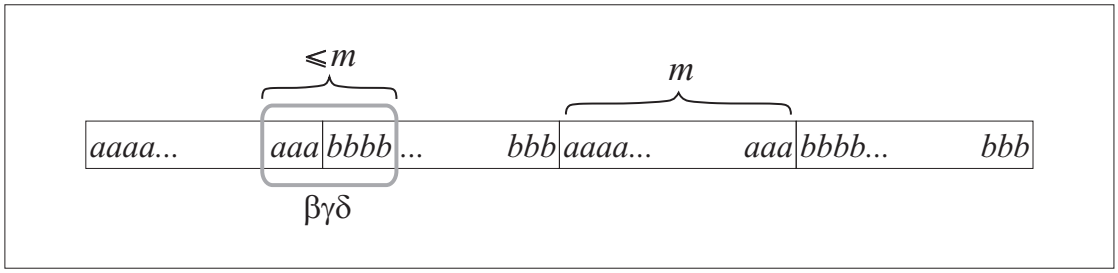
a przy tym wszystkie bez wyjątku słowa $\omega_k = \alpha\beta^k\gamma\delta^k\eta$ dla $k = 0, 1, 2, \ldots$ są także elementami L.

Przykład. $L = \{a^n b^n c^n : \ n \geq 0\}$. Załóżmy, że jest to język bezkontekstowy. Istnieje zatem stała m gwarantująca, że słowo $\omega = a^m b^m c^m$ dzięki swojej długości może być podzielone na segmenty $\omega = \alpha\beta\gamma\delta\eta$ w sposób zgodny z (i) oraz (ii) tak, że zastosowana do niego operacja pompowania $\omega_k = \alpha\beta^k\gamma\delta^k\eta$ generuje inne słowa z L. Jednak warunek (ii) $|\beta\gamma\delta| \leq m$ dla naszego słowa oznacza, że segment ten może mieć tylko jedną z postaci

$$a^r, \quad a^s b^t, \quad b^r, \quad b^s c^t \quad \text{lub ostatecznie} \quad c^r\,,$$

gdzie $r \leq m$ lub $s + t \leq m$. Wówczas jednak pompowanie powiela tylko jedną lub dwie spośród liter a, b, c, produkując słowa spoza języka L. Stąd wniosek, że język ten nie może być bezkontekstowy.

Przykład. $L = \{\mu\mu : \ \mu \in \{a,b\}^*\}$. Niech $\omega = a^m b^m a^m b^m$.



Segment $\beta\gamma\delta$ ze względu na ograniczoną długość nie większą niż m może mieścić się w obrębie co najwyżej dwóch sąsiednich bloków liter a i b. W każdym z możliwych położeń pompowanie niszczy symetrię słów języka L. Zatem nie może to być język bezkontekstowy.

Przykład. $L = \{a^{n^2} : \ n \geq 0\}$. Argument jest dokładnie *taki sam* jak w przypadku języków regularnych. Niech $L \in \mathcal{L}_2$ i $\omega = a^{m^2}$. Można więc je podzielić na 5 części

$$\omega \ = \ \alpha\beta\gamma\delta\eta \ = \ a^p\, a^q\, a^r\, a^s\, a^t\,, \qquad p+q+r+s+t = m^2\,,$$

gdzie (i) $q+r+s \leq m$, (ii) $q+s \geq 1$. Na podstawie tezy lematu wszystkie słowa

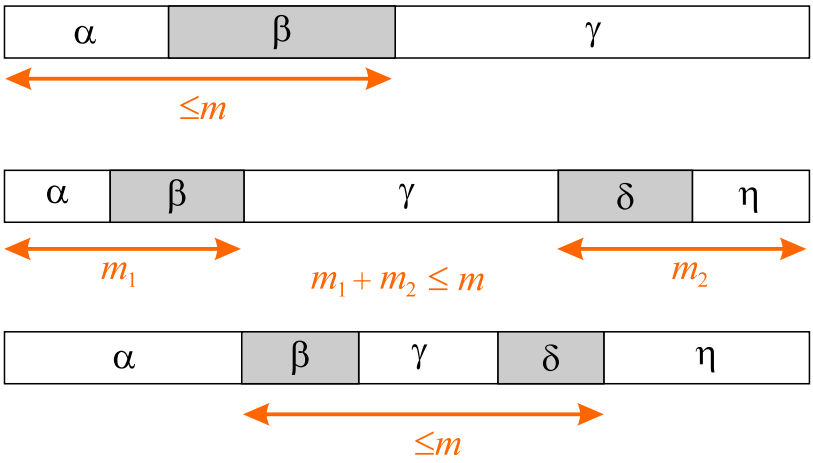
$$\omega_k = a^{p+(q+s)k+r+t}\,, \qquad k = 0,1,2,\ldots$$

są także elementami L. Jednak w L nie może być ani jednego słowa o pośredniej długości $m^2 < l < (m+1)^2$. Ale słowo $\omega_2 = a^{p+2q+r+2s+t} = a^{m^2+q+s}$ ma właśnie taką długość, bo $0 < q+s \leq m$, a więc $q+s < 2m+1$ skąd:

$$m^2 \ < \ m^2+q+s \ < \ m^2+2m+1 \ = \ (m+1)^2\,.$$

Widać zatem, że założenie o bezkontekstowości L musi być fałszywe.

Lematy o pompowaniu – porównanie

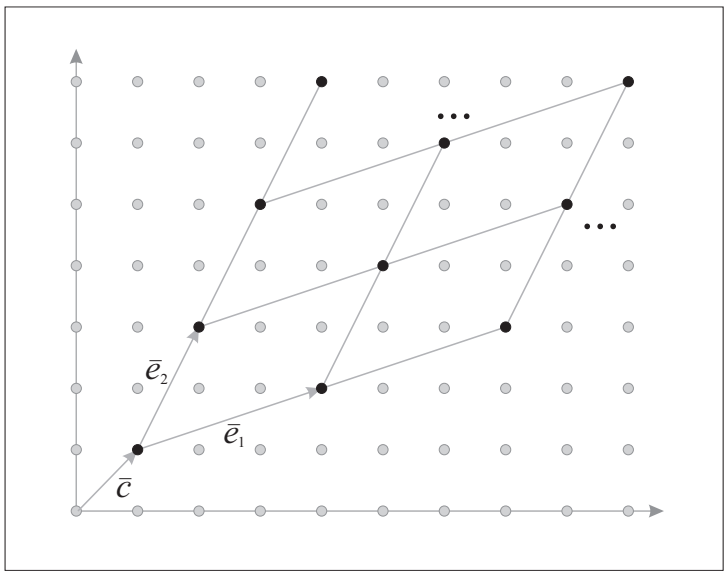


Twierdzenie Parikha

Podzbiór $K \subseteq \mathbb{N}^n$ nazywamy liniowym, jeśli istnieją wektory $\overline{c}$ i $\overline{e}_1, \ldots, \overline{e}_m \in \mathbb{N}^n$, $m \leq n$, dla których

$$K = \{\overline{c} + l_1\overline{e}_1 + l_2\overline{e}_2 + \cdots + l_m\overline{e}_m : l_1, l_2, \ldots l_m \in \mathbb{N}\}.$$

Zbór K nazywamy semiliniowym, jeśli jest on sumą mnogościową skończonej liczby zbiorów liniowych.



Zbiór liniowy można uważać za wielowymiarowe uogólnienie ciągu arytmetycznego, $a_n = a_0 + nr$, gdzie $\overline{c}$ pełni rolę wyrazu początkowego a_0, a wektory $\overline{e}_i$ są wielowymiarowymi przyrostami odpowiadającymi r.

Odwzorowaniem Parikha dla alfabetu $A = \{a_1, a_2, \ldots, a_n\}$ *nazywamy funkcję* $\Psi : A^* \to \mathbb{N}^n$ *określoną następująco:*

$$\Psi(\omega) \;=\; (|\omega|_{a_1}, |\omega|_{a_2}, \ldots, |\omega|_{a_n})\,.$$

Przykład: jeśli $A = \{a, b, c\}$, wówczas $\Psi(aacabb) = (3, 2, 1)$ itp.

Twierdzenie: *Jeśli język* L *jest bezkontekstowy, wówczas jego obrazem w odwzorowaniu Parikha* $\Psi(L)$ *jest zbiór semiliniowy.*

Zauważmy, że bardzo różne języki mogą mieć identyczną reprezentację w odwzorowaniu Parikha jako podzbiory $\mathbb{N}^n$. Na przykład $L = \{a^n b^n : \; n \geq 0\}$ i język poprawnie zagnieżdżonych nawiasów przekształcane są przez Ψ w zbiór $\{(n, n) : \; n \in \mathbb{N}\}$. Ma zatem sens następujące określenie: mówimy, że języki L i L' są *literowo (permutacyjnie) równoważne* jeśli $\Psi(L) = \Psi(L')$.

Lemat: *Każdy zbiór semiliniowy w* $\mathbb{N}^n$ *jest obrazem pewnego języka regularnego nad* n*-literowym alfabetem.*

Uzasadnienie dla $n = 2$. Weźmy K rozpięte przez $\overline{c} = (c_1, c_2)$ oraz $\overline{e}_1 = (p, q)$ i $\overline{e}_2 = (r, s)$. Tworzymy na tej podstawie wyrażenie regularne

$$\rho = a^{c_1} b^{c_2} (a^p b^q)^* (a^r b^s)^*\,.$$

Łatwo sprawdzić, że $\Psi(L(\rho)) = K$.
Jeśli K jest semiliniowy, $K = K_1 \cup \ldots \cup K_m$, utwórzmy podobnie wyrażenia regularne ρ_i dla K_i a następnie połóżmy $\rho = \rho_1 + \cdots + \rho_m$. $L(\rho)$ jest poszukiwanym językiem regularnym.

Wniosek: *Każdy język bezkontekstowy jest permutacyjnie równoważny z pewnym językiem regularnym.*

Stąd także: *Każdy język bezkontekstowy nad jednoliterowym alfabetem jest regularny.*

Uwaga: To że obraz Parikha jest zbiorem semiliniowym nie oznacza, że język jest bezkontekstowy, przykład: $L = \{a^n b^n c^n : \; n \geq 0\}$.

Wyniki dotyczące domkniętości i rozstrzygalności dla języków $\mathcal{L}_2$

Klasa $\mathcal{L}_2$ jest zamknięta ze względu na sumę mnogościową, konkatenację i $*$ Kleenego, nie jest natomiast zamknięta ze względu na przekrój i dopełnienie języków (a więc też nie ze względu na różnicę i różnicę symetryczną).

Istnieją przykłady języków bezkontekstowych, których przekroje i dopełnienia nadal są bezkontekstowe. Np. bezkontekstowe są języki

$$\{a^n b^n : \quad n \geq 0\}^c, \qquad \{\omega \overleftarrow{\omega} : \quad \omega \in \{a,b\}^*\}^c$$

dla przekroju np.

$$\{a^n b^m : \quad n \geq m\} \cap \{a^n b^m : \quad n \leq m\} \;=\; \{a^n b^n : \quad n \geq 0\}$$

Z kolei

$$\{a^n b^n c^m : \quad n, m \geq 0\} \cap \{a^m b^n c^n : \quad n, m \geq 0\} \;=\; \{a^n b^n c^n : \quad n \geq 0\}$$

to przykład, gdy przekrój wyprowadza poza klasę $\mathcal{L}_2$. Można też pokazać, że

$$L \;=\; \{\omega\omega : \quad \omega \in \{a,b\}^*\}^c$$

$$\begin{array}{lcl} S & \to & A|B|AB|BA \\ A & \to & a|aAa|aAb|bAa|bAb \\ B & \to & b|aBa|aBb|bBa|bBb \end{array}$$

jest bezkontekstowy (gramatyka po prawej stronie). Wiemy jednak już, że L^c nie jest klasy $\mathcal{L}_2$.

Oto jeden pozytywny rezultat:

Jeśli $L \in \mathcal{L}_2$ oraz $R \in \mathcal{L}_3$, wówczas $L \cap R$ jest bezkontekstowy.

Przykłady.

Pokażemy, że $L = \{a^n b^n : \quad n \geq 0, \quad n \neq 100\}$ jest bezkontekstowy. Weźmy jednoelementowy, a zatem regularny język $P = \{a^{100} b^{100}\}$. Dopełnienie $R = P^c$ też jest regularne. Ponieważ $L = \{a^n b^n : \; n \geq 0\} \cap R$, wnioskujemy, że jest on bezkontekstowy.

Pokażemy, że język $L = \{\omega : \quad |\omega|_a = |\omega|_b = |\omega|_c\}$ nie jest bezkontekstowy. Wynika to natychmiast z obserwacji, że

$$L \cap L(a^* b^* c^*) \;=\; \{a^n b^n c^n : \quad n \geq 0\}.$$

Zagadnienia rozstrzygalne w $\mathcal{L}_2$

1. Mając daną gramatykę G i słowo ω, rozstrzygnąć czy $\omega \in L(G)$.
   (Postać normalna Chomsky'ego i algorytm CYK $O(n^3)$.)
2. Mając daną gramatykę G, rozstrzygnąć czy $L(G)$ jest pusty, skończony bądź nieskończony.
3. Mając dane 2 gramatyki G i H typu $LL(k)$, rozstrzygnąć czy $L(G) = L(H)$.

Zagadnienia nierozstrzygalne w $\mathcal{L}_2$

1. Rozstrzygnąć czy $L(G) = L(H)$.
2. Rozstrzygnąć czy $L(G) \subseteq L(H)$.
3. Zbadać czy $L(G) \cap L(H) = \emptyset$.
4. Zbadać czy $L(G) = A^*$.
5. Rozstrzygnąć czy $L(G)$ jest językiem regularnym (to jest, czy dla danej gramatyki bezkontekstowej istnieje równoważna gramatyka regularna).