Hierarchia Chomsky'ego

7 lipca 2006

1 Lematy iteracyjne

1.1 Lematy dla języków regularnych

Niech L będzie językiem regularnym. Istnieje stała $N \geq 0$ taka, że dla każdego słowa $w \in L$

(1) jeśli $|w| \geq N$ to $\exists u_1, u_2, u_3 \in \Sigma^*$ takie, że $w = u_1u_2u_3$, $u_2 \neq \epsilon$ oraz $\forall i \geq 0\ u_1u_2^iu_3 \in L$

(2) jeśli $w = w_1w_2w_3$ oraz $|w_2| \geq N$ to $\exists u_1, u_2, u_3 \in \Sigma^*$ takie, że $w_2 = u_1u_2u_3$, $u_2 \neq \epsilon$ oraz $\forall i \geq 0\ w_1u_1u_2^iu_3w_3 \in L$

(3) jeśli $0 \leq i_0 \leq i_1 \leq \ldots \leq i_N \leq |w|$ (pozycje zaznaczone w słowie w) to istnieją $0 \leq j < k \leq N$ takie, że $w = u_1u_2u_3$, $|u_1| = i_j$, $|u_1u_2| = i_k$ oraz $\forall i \geq 0\ u_1u_2^iu_3 \in L$

Mamy tutaj ciąg trzech twierdzeń, o wzrastających możliwościach wnioskowania na temat regularności badanego języka. Tezę (3) można uważać za odpowiednik lematu Ogdena dla języków regularnych (patrz [1]).
Język $L_1 = \{\, b^na^p \mid n > 0,\ p \text{ jest liczbą pierwszą} \,\} \cup \{a\}^*$ spełnia (1) ale nie spełnia (2).
Język $L_2 = \{\, (ab)^n(cd)^n \mid n \geq 0 \,\} \cup \Sigma^*\{aa, bb, cc, dd, ac\}\Sigma^*$ spełnia (2) ale nie spełnia (3).
Język $L_3 = \{\, udv \mid u, v \in \{a, b, c\}^*,\ u \neq v \text{ albo } u \text{ lub } v \text{ zawiera podsłowo będące kwadratem} \,\}$ spełnia (3) ale nie jest językiem regularnym.
Oznaczając w twierdzeniu (3) $i_j = j$ otrzymujemy najbardziej znany lemat o pompowaniu dla języków regularnych:

Niech L będzie językiem regularnym. Istnieje stała $N \geq 0$ taka, że dla każdego słowa $w \in L$ zachodzi również:
jeśli $|w| \geq N$ to $\exists u_1, u_2, u_3 \in \Sigma^*$ takie, że $w = u_1u_2u_3$, $|u_1u_2| \leq N$ oraz $\forall i \geq 0\ u_1u_2^iu_3 \in L$

1.2 Lematy dla języków bezkontekstowych

1.2.1 Lemat o rozrastaniu (Y.Bar-Hillel, M.Perles, E.Shamir)

Niech L będzie językiem bezkontekstowym. Wówczas istnieje stała N zależna tylko od L taka, że dla każdego słowa $z \in L$ takiego, że $|z| > N$ zachodzi również:
istnieje faktoryzacja $z = uvwxy$, $|vx| > 0$ i $|vwx| \leq N$ taka, że dla każdego $i \geq 0$ uv^iwx^iy należy do L.

1.2.2 Lemat Ogdena

Niech L będzie językiem bezkontekstowym. Wówczas istnieje stała N zależna tylko od L taka, że dla każdego słowa $z \in L$ takiego, że $d(z) > N$, gdzie $d(z)$ oznacza liczbę wyróżnionych pozycji w słowie z, zachodzi również:
istnieje faktoryzacja $z = uvwxy$, $d(u)d(v)d(w) + d(w)d(x)d(y) \geq 1$ i $d(vwx) \leq N$ taka, że dla każdego $i \geq 0$ uv^iwx^iy należy do L.

1.2.3 Lemat Bader-Moury

Niech L będzie językiem bezkontekstowym. Wówczas istnieje stała N zależna tylko od L taka, że dla każdego słowa $z \in L$ takiego, że $d(z) > N^{e(z)+1}$, gdzie $d(z)$ oznacza liczbę wyróżnionych pozycji oraz $e(z)$ oznacza liczbę pozycji wykluczonych w słowie z, zachodzi również:
istnieje faktoryzacja $z = uvwxy$, $d(vx) \geq 1$, $e(vx) = 0$ i $d(vwx) \leq N^{e(vwx)+1}$ taka, że dla każdego $i \geq 0$ uv^iwx^iy należy do L.

1.3 Przykłady zastosowania lematów dla języków bezkontekstowych

W podrozdziale przyjęto następujące oznaczenia: w faktoryzacjach l oznacza część słowa położoną na lewo od znaków wyszczególnionych, analogicznie r oznacza część prawą. Jeśli pewna faktoryzacja nie zawiera symbolu l, oznacza to, że wyszczególnione pozycje są pierwszymi w danym słowie. Analogicznie jeśli nie ma symbolu r, wyszczególnione pozycje są pozycjami ostatnimi. ? oznacza dowolny terminal ze zbioru $\{a, b, c, d\}$. $d(symbol)$ oznacza liczbę wyróżnień wśród danego rodzaju symboli. Analogicznie $e(symbol)$ to liczba wykluczeń. W faktoryzacjach symbole wyróżniane są za pomocą daszka ($\hat{a}$) . z_i to i-ty symbol w słowie z. Początkowy symbol ma indeks 0. Wszystkie liczby występujące w podrozdziale są liczbami naturalnymi, przy czym $0 \in \boldsymbol{N}$.

Przykład 1.1 Zastosowanie lematu klasycznego
Przyjmijmy na potrzeby przykładu następujące oznaczenia:

$$
\begin{aligned}
A &= \{\, a^p b^p c^r d^r \mid 1 \leq p \leq r \,\} \\
B &= \{\, a^p b^q c^r d^s \mid 1 \leq q < p \ \wedge\ p - q \leq max(r, s) \,\} \\
C &= \{\, a^p b^q c^r d^s \mid 1 \leq r < s \ \wedge\ s - r \leq max(p, q) \,\} \\
D &= \{\, a^p b^q c^{s+1} d^s \mid p, q, s \geq 1 \,\}
\end{aligned}
$$

Język L jest zdefiniowany jest następująco: $L = A \cup B \cup C \cup D$.

Przy pomocy klasycznego lematu o pompowaniu wykażemy, że język L nie jest bezkontekstowy.

Dowód. Niech $z = a^n b^n c^n d^n$, gdzie n-stała z klasycznego lematu o pompowaniu. Załóżmy, że język L jest bezkontekstowy. Zatem dla tego słowa powinna istnieć faktoryzacja spełniająca tezę lematu. Rozważmy wszystkie możliwości:

1. vx zawiera wyłącznie k symboli a ($k > 0$). Po napompowaniu słowo przyjmuje postać $z' = a^{n+k(i-1)} b^n c^n d^n$. Gdy $i = 2n + 1$, $z' = a^{n(2k+1)} b^n c^n d^n$. $z' \notin A$, ponieważ symboli a jest zbyt wiele. Gdyby $z' \in B$, wówczas musiałoby zachodzić $p - q \leq max(r, s)$. Tymczasem $p - q = 2nk > n = max(r, s)$, występuje sprzeczność. W dalszej części pracy będziemy stosowali skrócony zapis tego typu dowodów. $z' \notin C \cup D$, ponieważ symboli c jest tyle samo, co d.

2. vx zawiera wyłącznie k symboli b ($k > 0$). Po napompowaniu słowo przyjmuje postać $z' = a^n b^{n+k(i-1)} c^n d^n$. Gdy $i = 2$, $z' = a^n b^{n+k} c^n d^n$. $z' \notin A$, ponieważ symboli b jest zbyt wiele. $z' \notin B$, ponieważ $p < q$. $z' \notin C \cup D$, ponieważ symboli c jest tyle samo, co d.

3. vx zawiera wyłącznie k symboli c ($k > 0$). Po napompowaniu słowo przyjmuje postać $z' = a^n b^n c^{n+k(i-1)} d^n$. Gdy $i = 3$, $z' = a^n b^n c^{n+2k} d^n$. $z' \notin A$, ponieważ symboli c jest zbyt

wiele. $z' \notin B$, ponieważ $p = q$. $z' \notin C$, ponieważ $r > s$. $z' \notin D$, ponieważ symboli c jest więcej niż d o co najmniej 2.

4. vx zawiera wyłącznie k symboli d ($k > 0$). Po napompowaniu słowo przyjmuje postać $z' = a^n b^n c^n d^{n+k(i-1)}$. Gdy $i = 2n + 1$, $z' = a^n b^n c^n d^{n(2k+1)}$. $z' \notin A$, ponieważ symboli d jest zbyt wiele. $z' \notin B$, ponieważ $p = q$. $z' \notin C$, ponieważ $s - r = 2nk > n = max(p, q)$. $z' \notin D$, ponieważ symboli d jest więcej, niż c.

5. v zawiera wyłącznie k symboli a ($k > 0$). x zawiera wyłącznie m symboli b ($m > 0$). Po napompowaniu słowo przyjmuje postać $z' = a^{n+k(i-1)} b^{n+m(i-1)} c^n d^n$. Gdy $i = 2n + 1$, $z' = a^{n(2k+1)} b^{n(2m+1)} c^n d^n$. $z' \notin A$, ponieważ symboli a jest więcej niż c. $z' \notin B$, ponieważ dla $k > m$: $p - q = 2n(k - m) > n = max(r, s)$, natomiast dla $k \leq m$: $p \leq q$. $z' \notin C \cup D$, ponieważ symboli c jest tyle samo, co d.

6. v zawiera wyłącznie k symboli b ($k > 0$). x zawiera wyłącznie m symboli c ($m > 0$). Po napompowaniu słowo przyjmuje postać $z' = a^n b^{n+k(i-1)} c^{n+m(i-1)} d^n$. Gdy $i = 3$, $z' = a^n b^{n+2k} c^{n+2k} d^n$. $z' \notin A$, ponieważ symboli b jest więcej niż a. $z' \notin B$, ponieważ $p < q$. $z' \notin C$, ponieważ $r > s$. $z' \notin D$, ponieważ symboli c jest więcej niż d o co najmniej 2.

7. v zawiera wyłącznie k symboli c ($k > 0$). x zawiera wyłącznie k symboli d. Po napompowaniu słowo przyjmuje postać $z' = a^n b^n c^{n+k(i-1)} d^{n+k(i-1)}$. Gdy $i = 0$, $z' = a^n b^n c^{n-k} d^{n-k}$. $z' \notin A$, ponieważ symboli a jest więcej niż c. $z' \notin B$, ponieważ $p = q$. $z' \notin C \cup D$, ponieważ symboli c jest tyle samo, co d.

8. v zawiera wyłącznie k symboli c ($k > 0$). x zawiera wyłącznie m symboli d ($0 < m \neq k$). Po napompowaniu słowo przyjmuje postać $z' = a^n b^n c^{n+k(i-1)} d^{n+m(i-1)}$. Gdy $i = 2n + 1$, $z' = a^n b^n c^{n(2k+1)} d^{n(2m+1)}$. $z' \notin A$, ponieważ symboli c nie jest tyle samo, co d. $z' \notin B$, ponieważ $p = q$. $z' \notin C$, ponieważ dla $k < m$: $s - r = 2n(m - k) > n = max(p, q)$, natomiast dla $k \geq m$ $s \leq r$. $z' \notin D$, ponieważ liczby symboli c i d różnią się o $2n(k - m) \neq 1$.

Dla każdego z rozważanych przypadków $z' \notin L$. Pozostaje udowodnić, że nie ma innej możliwości, niż osiem już rozważonych. Gdy do v bądź x należy więcej niż 1 rodzaj symboli, po napompowaniu wystąpi niedozwolony przeplot symboli. Poza rozważonymi przypadkami pozostają już tylko takie, w których pomiędzy v i x (czyli w w) znajdują się wszystkie symbole co najmniej jednego rodzaju. Taka faktoryzacja nie jest jednak możliwa, ponieważ wówczas nie jest zachowany warunek $|vwx| < n$.
A zatem rozważone zostały wszystkie możliwe faktoryzacje i żadna z nich nie spełnia tezy lematu. Początkowe założenie nie jest prawdziwe. Otrzymaliśmy sprzeczność. Zatem język L nie jest bezkontekstowy.

Ograniczenia lematu klasycznego i lematu Ogdena oraz lemat Bader-Moury

Na potrzeby kolejnych trzech przykładów przyjmijmy oznaczenia:

$$A = \{ a^p b^p c^r d^r f^n g^n \mid 1 \leq p \leq r \ \wedge \ n \geq 1 \}$$
$$B = \{ a^p b^q c^r d^s f^n g^n \mid 1 \leq q < p \ \wedge \ p - q \leq max(r, s) \ \wedge \ n \geq 1 \}$$
$$C = \{ a^p b^q c^r d^s f^n g^n \mid 1 \leq r < s \ \wedge \ s - r \leq max(p, q) \ \wedge \ n \geq 1 \}$$
$$D = \{ a^p b^q c^{s+1} d^s f^n g^n \mid p, q, s \geq 1 \ \wedge \ n \geq 1 \}$$
$$E = \{a, b, c, d\}^* \{ f^m g^n \mid m \neq n \} = \{ ?^p f^m g^n \mid m \neq n \}$$

Oznaczmy: $E = E_1 \cup E_2$, gdzie $E_1 = \{\, ?^p f^m g^n \mid m > n \,\}$ oraz $E_2 = \{\, ?^p f^m g^n \mid m < n \,\}$. Język L jest zdefiniowany następująco: $L = A \cup B \cup C \cup D \cup E$. Język ten nie jest bezkontekstowy. Okazuje się jednak, że przy pomocy klasycznego lematu o pompowaniu i lematu Ogdena nie da się tego wykazać. Oto dowody:

Przykład 1.2 Ograniczenia lematu klasycznego
Język L spełnia warunki klasycznego lematu o pompowaniu.

Dowód. Wykażemy, że istnieje taka stała N, że dla każdego słowa $z : z \in L \wedge |z| \geq N$ istnieje faktoryzacja spełniająca założenia lematu. Na bieżąco będziemy sprawdzali tylko warunek dotyczący należenia słów po napompowaniu do języka. Pozostałe warunki sprawdzimy na końcu.

Niech $z \in A \cup B \cup C \cup D$. Rozważmy faktoryzację:

u	**v**	**w**	**x**	**y**
l	ϵ	ϵ	f	r

Po napompowaniu dla $i = 0$ otrzymamy słowo należące do E_2, a dla $i > 0$ - do E_1. Założenie jest spełnione.

Niech teraz $z \in E$. Gdy $p > 0$, stosujemy faktoryzację:

u	**v**	**w**	**x**	**y**
ϵ	$?$	ϵ	ϵ	r

Po napompowaniu słowo dalej należy do E, ponieważ ilość symboli ? może być dowolna.

Jeśli $p = 0 \wedge m > 0 \wedge n > 0$, stosujemy faktoryzację:

u	**v**	**w**	**x**	**y**
l	f	ϵ	g	r

Po napompowaniu słowo dalej należy do E, ponieważ niezerowa różnica między liczbami symboli f i g zostaje zachowana.

Jeśli $p = 0 \wedge m > 0 \wedge n = 0$, $z \in E_1$ i stosujemy faktoryzację:

u	**v**	**w**	**x**	**y**
ϵ	ϵ	ϵ	f	r

Po napompowaniu niezerowa liczba symboli f zostaje zachowana, jeżeli część y zawiera co najmniej 1 symbol f, czyli jeśli $N \geq 2$. Wówczas słowo dalej należy do E_1.

Jeśli natomiast $p = 0 \wedge m = 0$, wówczas $n > 0$ i $z \in E_2$, ponieważ $\epsilon \notin L$. Stosujemy faktoryzację:

u	**v**	**w**	**x**	**y**
ϵ	ϵ	ϵ	g	r

Sytuacja jest analogiczna, jak poprzednio. Po napompowaniu niezerowa liczba symboli g zostaje zachowana, jeżeli część y zawiera co najmniej 1 symbol g, czyli jeśli $N \geq 2$. Wówczas słowo pozostaje w E_2.

Dotychczas wszystkie niezbędne warunki są spełnione dla $N \geq 2$. Sprawdźmy, czy przedstawione faktoryzacje spełniają pozostałe założenia lematu. Dla wszystkich faktoryzacji $|vx| = |vwx| \in \{1, 2\}$. A zatem $|vx| \geq 1$, natomiast $|vwx| < N$ dla $N \geq 3$. Podsumowując, dla dowolnego $N : N \geq 3$ spełnione są wszystkie warunki lematu. A zatem szukana stała N istnieje. Przy pomocy lematu o rozrastaniu nie jesteśmy w stanie rozstrzygnąć, czy L jest bezkontekstowy, czy też nie.

Przykład 1.3 Ograniczenia lematu Ogdena
Język L spełnia warunki lematu Ogdena.

Dowód. Wykażemy, że istnieje taka stała N, że dla każdego słowa $z : z \in L \wedge d(z) > N$ istnieje faktoryzacja spełniająca założenia lematu.

Niech $z \in A \cup B \cup C \cup D$. Jeśli $d(a)+d(b)+d(c)+d(d) \geq 3$, oznaczamy 3 ostatnie spośród wyróżnionych pozycji symboli a, b, c, d przez z_i, z_j, z_k. Ostatni symbol ze zbioru $\{a, b, c, d\}$ oznaczmy przez z_d. Stosujemy faktoryzację:

u	**v**	**w**	**x**	**y**
$z_0 z_1 ... z_{j-1}$	z_j	$z_{j+1} z_{j+2} ... z_d$	f	r

Po napompowaniu dla $i = 0$ otrzymamy słowo należące do E_2, a dla $i > 1$ - do E_1. $d(u)d(v)d(w) > 0$, ponieważ u, v, w zawierają odpowiednio z_i, z_j, z_k. $d(vwx) \in \{2, 3\}$, a zatem warunek $d(vwx) \leq N$ jest spełniony dla $N \geq 3$.

Jeżeli $d(a) + d(b) + d(c) + d(d) < 3 \wedge d(f) \geq 3$, to stosujemy następującą faktoryzację:

u	**v**	**w**	**x**	**y**
$lf...\hat{f}...f$	$\hat{f}$	$f...\hat{f}$	ϵ	r

przy czym u, v, w zawierają kolejno 3 pierwsze pozycje wyróżnione wśród symboli f. Po napompowaniu dla $i = 0$ otrzymamy słowo należące do E_2, a dla $i > 0$ - do E_1. $d(u)d(v)d(w) > 0$, ponieważ u, v, w zawierają co najmniej po jednym symbolu wyróżnionym. $d(vwx) = 2 \leq N$ dla $N \geq 2$.

Jeżeli $d(a) + d(b) + d(c) + d(d) < 3 \wedge d(f) < 3 \wedge d(g) \geq 3$, sytuacja jest analogiczna, jak w ostatnim przypadku; faktoryzacja przyjmuje postać:

u	**v**	**w**	**x**	**y**
$lg...\hat{g}...g$	$\hat{g}$	$g...\hat{g}$	ϵ	r

z podobnymi jak poprzednio warunkami. Jedyna zmiana jest taka, że dla $i = 0$ otrzymamy słowo należące do E_1, a dla $i > 0$ - do E_2.
Aby zaszedł jeden z powyższych przypadków, musimy mieć pewność, że dla $d(a)+d(b)+d(c)+d(d) < 3 \wedge d(f) < 3$ zachodzi $d(g) \geq 3$. Powołując się na warunek $d(z) > N$ stwierdzamy, że N musi wynosić co najmniej 6.

Rozważmy teraz przypadek, gdy $z \in E$. Jeżeli $d(?) \geq 3$, stosujemy faktoryzację:

u	**v**	**w**	**x**	**y**
$?...\hat{?}...?$	$\hat{?}$	$?...\hat{?}$	ϵ	r

przy czym u, v, w zawierają kolejno 3 pierwsze pozycje wyróżnione wśród symboli ?. Po napompowaniu otrzymamy słowo należące do E, ponieważ symboli ? może być dowolna ilość. $d(u)d(v)d(w) > 0$, ponieważ u, v, w zawierają co najmniej po jednym symbolu wyróżnionym. $d(vwx) = 2 \leq N$ dla $N \geq 2$.

Jeżeli $d(?) < 3 \wedge d(f) \geq 3 \wedge n > 0$, stosujemy faktoryzację:

u	**v**	**w**	**x**	**y**
$lf...\hat{f}...f$	$\hat{f}$	$f...\hat{f}...f$	g	r

przy czym u, v, w zawierają kolejno 3 ostatnie pozycje wyróżnione wśród symboli f. Po napompowaniu otrzymamy słowo należące dalej do E, ponieważ różnica między ilościami symboli f i g zostanie zachowana. $d(u)d(v)d(w) > 0$, ponieważ u, v, w zawierają co najmniej po jednym symbolu wyróżnionym. $d(vwx) \in \{2, 3\}$, a zatem założenia są spełnione dla $N \geq 3$.

Jeżeli $d(?) < 3 \wedge d(f) \geq 3 \wedge n = 0$, wówczas $z \in E_1$ i stosujemy faktoryzację:

u	**v**	**w**	**x**	**y**
$lf...\hat{f}...f$	$\hat{f}$	$f...\hat{f}$	ϵ	r

przy czym u, v, w zawierają kolejno 3 ostatnie pozycje wyróżnione wśród symboli f. Po napompowaniu otrzymamy słowo należące dalej do E_1, ponieważ dodatnia ilość symboli f zostaje zachowana. $d(u)d(v)d(w) > 0$, $d(vwx) = 2 < N$ dla $N \geq 2$.

Gdy $d(?) < 3 \wedge d(f) < 3 \wedge d(g) \geq 3$, występują znowu 2 przypadki. Gdy $m > 0$, stosujemy faktoryzację:

u	**v**	**w**	**x**	**y**
l	f	$g...\hat{g}...g$	$\hat{g}$	$g...\hat{g}...g$

przy czym w, x, y zawierają kolejno 3 pierwsze pozycje wyróżnione wśród symboli g. Po napompowaniu otrzymamy słowo należące dalej do E, ponieważ różnica między ilościami symboli f i g zostanie zachowana. $d(w)d(x)d(y) > 0$, $d(vwx) \in \{2, 3\}$, a zatem założenia są spełnione dla $N \geq 3$.

Gdy $m = 0$, wówczas $z \in E_2$ i stosujemy faktoryzację:

u	**v**	**w**	**x**	**y**
l	ϵ	$\hat{g}...g$	$\hat{g}$	$g...\hat{g}...g$

przy czym w, x, y zawierają kolejno 3 pierwsze pozycje wyróżnione wśród symboli g. Po napompowaniu otrzymamy słowo należące dalej do E_2, ponieważ dodatnia ilość symboli g zostanie zachowana. $d(w)d(x)d(y) > 0$, $d(vwx) = 2$, a zatem założenia są spełnione dla $N \geq 2$.

Dla $z \in E$, podobnie jak dla $z \in A \cup B \cup C \cup D$, aby jeden z powyższych przypadków na pewno zaszedł, konieczne jest, by $N \geq 6$. Jednocześnie zauważamy, że dla tej wartości N spełnione są wszystkie pozostałe wymagania, które pojawiły się podczas dowodu. Podsumowując, dla dowolnego $N : N \geq 6$ spełnione są wszystkie tezy lematu. A zatem szukana stała N istnieje. Przy

pomocy lematu Ogdena nie jesteśmy w stanie udowodnić, że L nie jest bezkontekstowy.

Przykład 1.4 Zastosowanie lematu Bader-Moury
Powyższe dowody były dość skomplikowane - trzeba było przeanalizować stosunkowo dużo przypadków. Ponadto nadal nie wykazaliśmy, czy język L jest bezkontekstowy, czy też nie. Dla tego konkretnego języka pomocny okazuje się lemat Bader-Moury.

Przy pomocy lematu Bader-Moury wykażemy, że język L nie jest bezkontekstowy.

Dowód. Pokażemy, że następujący fakt zachodzi dla dowolnego dodatniego N. Jeżeli wyróżnimy i wykluczymy pewne pozycje w słowie $z : z \in L$, przy czym $d(z) > N^{e(z)+1}$, dojdzie do sytuacji, w której nie będzie możliwe znalezienie faktoryzacji spełniającej warunki zadania.
Niech $z \in A \cup B \cup C \cup D$, z zawiera po jednym symbolu f i g. Wykluczmy dwie końcowe pozycje fg. Wyróżnijmy wszystkie pozycje symboli a, b, c, d. Oczywiście musi zachodzić nierówność $d(z) > N^3$, czyli $|z| > N^3 + 2$.
Oznaczmy język z przykładu 1.1 przez L'. Jeżeli w słowie z pominiemy należącą zawsze do części y faktoryzacji wykluczoną końcówkę fg, to powstałe słowo należy do L'. A zatem tezę lematu Bader-Moury można zapisać w postaci:

Istnieje taka stała N, że jeśli $z \in L' \wedge |z| > N^3$, to istnieje faktoryzacja $z = uvwxy$, dla której

- $|vx| \geq 1$,
- $|vwx| \leq N$,
- $\forall i \geq 0 : uv^i wx^i y \in L'$.

Jest to zaostrzenie klasycznego lematu o pompowaniu, zatem skoro L' nie spełnia lematu o rozrastaniu (patrz dowód w przkładzie 1.1), tym bardziej nie będzie spełniona zaostrzona wersja twierdzenia. Dlatego też istnieje takie słowo, dla którego szukana faktoryzacja nie istnieje, wykazanie czego bylo celem ninejszego dowodu. Warunki lematu Bader-Moury nie są spełnione. Możemy stąd wywnioskować, że L nie jest językiem bezkontekstowym.

1.4 Lemat dla języków deterministycznych bezkontekstowych

Lemat o rozrastaniu się języków deterministycznych bezkontekstowych (Sheng Yu, [9]).
Oznaczmy przez $FIRST_k(w)$ dla $w \in \Sigma^*$ pierwsze k symboli w:

$$FIRST_k(w) = \begin{cases} x & \text{jeżeli } |w| > k,\ w = xy,\ |x| = k \\ w & \text{jeżeli } |w| \leq k \end{cases}$$

Niech L będzie językiem DCFL. Wówczas istnieje stała C zależna tylko od L taka, że dla każdej pary słów w, $w' \in L$ jeżeli spełnione są założenia: $w = xy$ oraz $w' = xz$, $|x| > C$ oraz $FIRST_1(y) = FIRST_1(z)$, to zachodzi również:

(1) istnieje faktoryzacja $x = x_1x_2x_3x_4x_5$, $|x_2x_4| \geq 1$ i $|x_2x_3x_4| \leq C$ taka, że dla każdego $i \geq 0$, $x_1x_2^ix_3x_4^ix_5y$ oraz $x_1x_2^ix_3x_4^ix_5z$ należą do L,

(2) istnieją faktoryzacje $x = x_1x_2x_3$, $y = y_1y_2y_3$ i $z = z_1z_2z_3$, $|x_2| \geq 1$ oraz $|x_2x_3| \leq C$ takie, że dla każdego $i \geq 0$, $x_1x_2^ix_3y_1y_2^iy_3$ oraz $x_1x_2^ix_3z_1z_2^iz_3$ należą do L,

Przykład 1.5 Język $L = \{a^i b^i \mid i \geq 0\} \cup \{a^i b^{2i} \mid i \geq 0\}$ nie jest DCFL.
Dowód. Przyjmijmy, że L jest DCFL i niech C będzie stałą dla L z lematu o pompowaniu. Wybierzmy $w = a^n b^n$ i $w' = a^n b^{2n}$ dla liczb całkowitych $n > C$, oraz $x = a^n b^{n-1}$, $y = b$, $z = b^{2n-1}$. Wybór $w = xy$ i $w' = xz$ spełnia założenia lematu. Zgodnie z lematem punkty (1) i (2) również powinny być spełnione. Rozważmy punkt (1). Jedyny możliwy podział $x = x_1x_2x_3x_4x_5$ taki, że $|x_2x_4| > 0$ oraz dla każdego i $x_1x_2^ix_3x_4^ix_5y \in L$, musi spełnić warunek $x_2 = a^k$ i $x_4 = b^k$ dla pewnego $k > 0$. Ale wówczas $x_1x_2^0x_3x_4^0x_5z = x_1x_2x_3z = a^{n-k}b^{2n-k} \notin L$. Dlatego punkt (1) nie jest spełniony. Teraz rozważmy punkt (2). Jakakolwiek faktoryzacja $x = x_1x_2x_3$ taka, że $|x_2| > 0$ i $|x_2x_3| \leq C < n$ spowoduje, że $x_2 \in b^+$ oraz $y_2 \in b^*$, więc $x_1x_3y_1y_3 = a^nb^{n-|x_2|-|y_2|} \notin L$. Zatem (2) również nie jest spełniony. Zaprzecza to lematowi o pompowaniu, toteż język L nie jest DCFL.

Przykład 1.6 Język $L = \{\, w \in \{a,b\}^* \mid w = uv,\ |u| = |v|,\ v \text{ zawiera } a \,\}$ nie jest DCFL.
Dowód. Przyjmijmy, że L jest DCFL. Niech C będzie stałą z lematu o pompowaniu. Rozważmy parę słów w i w' należących do języka L: $w = b^{n+1}ab^n$, $w' = b^{n+1}ab^nab^{2n+1}$, gdzie $n > C$. Niech $x = b^{n+1}ab^{n-1}$, $y = b$, $z = bab^{2n+1}$. Wówczas $w = xy$ i $w' = xz$ spełniają założenia z lematu o pompowaniu. Również (1) i (2) powinny być spełnione. Jednakże (1) nie jest spełniony, ponieważ każdy podział $x = x_1x_2x_3x_4x_5$ taki, że $|x_2x_4| \geq 1$ implikuje, że $x_1x_3x_5z$ nie zawiera żadnego a w drugiej połowie i nie należy do L. Punkt (2) również nie jest spełniony. Dla jakiejkolwiek faktoryzacji $x = x_1x_2x_3$ i $y = y_1y_2y_3$ takiej, że $|x_2x_3| \leq C < n$ i $|x_2| \geq 1$ otrzymujemy $x_2 \in b^+$, $y_2 = b$ lub $y_2 = \epsilon$. Dlatego też $x_1x_2^2x_3y_1y_2^2y_3 = b^{n+1}ab^{n+|x_2|+|y_2|} \notin L$. Przeczy to lematowi o pompowaniu. Zatem L nie jest DCFL.

2 Twierdzenie Parikh'a

Twierdzenie Parikh'a jest użyteczne przy wnioskowaniu o umiejscowieniu języka w hierarchii Chomsky'ego, gdyż pozwala uprościć problem przynależności języka do klasy języków bezkontekstowych poprzez zamianę na równoważny (często prostszy) problem przynależności do klasy języków regularnych.

Odwzorowanie Parikh'a

Niech będzie dany alfabet $\Sigma = \{\, a_1, a_2, \ldots, a_r \,\}$. Dla danego słowa $w \in \Sigma$ tradycyjnie oznaczamy przez $|w|_{a_i}$ liczbę wystąpień symbolu a_i w słowie w.
Dla danego słowa w odwzorowanie Parikh'a $\Phi : \Sigma^* \to \mathbf{N}^r$ definiuje wektor r-elementowy w następujący sposób:

$\Phi(w) = (\, |w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_r} \,)$

Dla języka L definiujemy natomiast: $\Phi(L) = \{\, \Phi(w) \mid w \in L \,\}$.
Niech wektory $V_0, V_1, \ldots, V_k \in \mathbf{N}^r$. Oznaczmy dodatkowo $\Gamma = \{V_1, \ldots, V_k\}$.
Zbiór $T \subseteq \mathbf{N}^r$ nazywamy liniowym jeśli $T = \{\, V_0 + \alpha_1V_1 + \cdots + \alpha_kV_k \mid \alpha_i \in \mathbf{N},\ i = 1, \ldots, k \,\}$, co zapisujemy w skrócie $T = V_0 + span(\Gamma)$.

Zbiór T nazywamy semiliniowym jeśli jest skończoną sumą zbiorów liniowych, tj. $T = \bigcup_{i=1}^{N} T_i$, gdzie każdy T_i jest liniowy.

Twierdzenie Parikh'a

Jeśli język L jest bezkontekstowy to zbiór $\Phi(L)$ jest semiliniowy.

Innymi słowy jeżeli język $L : L \subseteq \Sigma^* \wedge |\Sigma| = r$ jest bezkontekstowy, to $\Phi(L) = \bigcup_{i=1}^{N} T_i$, przy czym $T_i = v_i + span(\Gamma_i)$ dla pewnego r-elementowego wektora v_i i skończonego zbioru Γ_i wektorów r-elementowych, $i = 1, 2, ..., N$.

Przykład 2.1 Zastosowanie lematu Parikha.
Język $L = \{ a^n b^m c^k \mid k \neq nm \}$ nie jest językiem bezkontekstowym.

Załóżmy, że język L jest bezkontekstowy. Wówczas, na mocy lematu Parikha, $\Phi(L) = \bigcup_{i=1}^{N} T_i$, przy czym $T_i = v_i + span(\Gamma_i)$ dla $i = 1, 2, ..., N$, dla pewnego trójelementowego wektora v_i i skończonego zbioru Γ_i wektorów trójelementowych. Niech d będzie największą liczbą spośród współrzędnych wektorów należących do zbioru $\Gamma = \bigcup_{i=1}^{N} (\{v_i\} \cup \Gamma_i)$. Zauważmy, że $d \geq 1$.

Rozważmy wektor (n, n, m), gdzie $n = [(d+2)!]^2$ oraz $m = [(d+2)!]^3$. Oczywiście $(n, n, m) \in \Phi(L)$. A zatem $\exists\, 1 \leq i \leq N : (n, n, m) \in T_i$.

Ponieważ $(n, n, m) \in T_i$, istnieją liczby $\alpha_1, \ldots, \alpha_l$ i należące do Γ_i niezerowe wektory $(p_1, q_1, r_1), \ldots, (p_l, q_l, r_l)$ takie, że

$$(n, n, m) = v_i + \sum_{j=1}^{l} \alpha_j (p_j, q_j, r_j).$$

Udowodnimy teraz nie wprost, że istnieje należący do Γ_i wektor $(0, 0, s)$, gdzie $1 \leq s \leq d$. Załóżmy, że $\forall\, 1 \leq j \leq l : p_j + q_j \geq 1$. A zatem $\sum_{j=1}^{l} \alpha_j \leq \sum_{j=1}^{l} \alpha_j (p_j + q_j) = 2n$. Zauważmy, że

$$m \leq d + \sum_{j=1}^{l} \alpha_j r_j \leq d + d \sum_{j=1}^{l} \alpha_j \leq (2n+1)d.$$

Tymczasem

$$m = (d+2)![(d+2)!]^2 \geq 3d[(d+2)!]^2 = 3nd > (2n+1)d.$$

Wystąpiła sprzeczność. A zatem uwodniliśmy, że $(0, 0, s) \in \Gamma_i$. Stąd

$$\forall \beta \geq 0 : (n, n, m') = (n, n, m + \beta s) \in T_i \subseteq \Phi(L).$$

By doprowadzić do sprzeczności, należy znaleźć taką liczbę β, by powyższa zależność nie była spełniona. Ponieważ $1 \leq s \leq d$, $s | n^2 - m$ i możemy przyjąć $\beta = \frac{n^2 - m}{s}$. Wówczas $\beta s = n^2 - m$. Ponieważ $m' = n^2$ to $(n, n, m') \notin \Phi(L)$. Wystąpiła sprzeczność, a zatem założenie okazało się nieprawdziwe. Język L nie jest bezkontekstowy, co należało udowodnić.

Wniosek. Z lematu Parikh'a wynika następujące, użyteczne twierdzenie:
Jeśli język L zdefiniowany nad alfabetem jednoliterowym jest bezkontekstowy to jest regularny.

Przykład 2.2 Dany jest język L nad alfabetem $\Sigma = \{a, b\}$ składający się ze słów Fibonacci'ego zdefiniowanych następująco: $w_0 = a$, $w_1 = b$, $w_{n+2} = w_n w_{n+1}$.

Należy rozstrzygnąć czy język L jest bezkontekstowy.
Zdefiniujmy homomorfizm h poprzez odwzorowanie $h(a) = h(b) = 0$.
$L' = h(L) = \{ 0^{F_n} \mid n \geq 1, F_1 = 1, F_2 = 2, F_{n+2} = F_{n+1} + F_n \}$ jest językiem zdefiniowanym nad alfabetem jednoliterowym. Zatem z twierdzenia Parikh'a wynika, że jeśli L jest bezkontekstowy to L' jest regularny. Teraz wystarczy zastosować zwykły lemat o pompowaniu dla języków regularnych. Weźmy słowo $0^m \in L'$ gdzie m jest większe lub równe od stałej z lematu o pompowaniu. W takim razie również słowo $0^{m+ki} \in L'$, gdzie i jest dowolną liczbą naturalną a k pewną stałą ($k \leq m$). Zatem $m + ki$ oraz $m + k(i + 1)$ są elementami ciągu Fibonacci'ego czyli $m + ki = F_{p_i}$ oraz $m + k(i + 1) = F_{p_{i+1}}$. Stąd $F_{p_{i+1}} - F_{p_i} = k$. Ponieważ $F_{p_{i+1}} \geq F_{p_i+1}$ to $k = F_{p_{i+1}} - F_{p_i} \geq F_{p_i+1} - F_{p_i} = F_{p_{i-1}}$ czyli ciąg F_{p_i} jest ograniczony stałą k. Z drugiej strony wiemy, że ciągi F_n oraz p_i są rosnące. Sprzeczność. Zatem L' nie jest językiem regularnym i na mocy twierdzenia Parikh'a L nie jest językiem bezkontekstowym.
W podobny sposób można udowodnić, że język L_∞ składający się ze słów zdefiniowanych następująco: $w_0 = 1$, $w_1 = 10$, $w_n = w_{n-1}w_{n-2}^2$, $n \geq 2$, związany z symboliczną dynamiką chaotyczną ([5]), nie jest bezkontekstowy.

3 Złożoność Kołmogorowa

3.1 Wprowadzenie

W jakim sensie słowo 011010110111001 jest bardziej skomplikowane niż 0101010101010101? Ponieważ złożoność obliczeniowa zajmuje się nieskończonymi językami, a nie skończonymi słowami, więc na powyższe pytanie nie da odpowiedzi. Istnieje jednak teoria opisowa, której podstawy stworzył A. Kołmogorow, zajmująca się takimi pytaniami. Teoria opisowa może zostać zastosowana do badania języków formalnych. Przykładowo w trakcie parsingu przeprowadzanego przez deterministyczny automat ze stosem na jego stosie nie mogą pojawiać się dowolne słowa, lecz tylko takie, które spełniają pewne zależności odnoszące się do ich złożoności.

3.1.1 Podstawowe definicje

Oznaczmy przez x zarówno liczbę naturalną, jak i *n–ty* binarny element ciągu o następującej sekwencji: 0,1,00, 01,10,11,000,001,010,011,100,101,110,111,0000, ... Tym samym reprezentacja '3' pokrywa się z naturalną liczbą 3 i z binarnym zapisem 00. Możemy zatem zdefiniować bijekcję pomiędzy nieujemnymi liczbami naturalnymi, a skończonymi binarnymi napisami $\{0, 1\}^*$.

Niech $|x|$ oznacza długość napisu x (ilość bitów) w binarnym zapisie napisu x. Oznaczmy przez Σ alfabet złożony z dwóch symboli 0 i 1. Dla każdego słowa $x = x_1x_2 \ldots x_n \in \Sigma^*$ w ułożeniu leksykalnym, prawdziwe jest następujące stwierdzenie:
Kolejny numer każdego napisu w ciągu możemy obliczyć z następującego wzoru:

$$G(x) = 2^n - 1 + \Sigma_{i=0}^{n-1} x_i 2^i. \tag{3.1}$$

Ponadto możemy zdefiniować

$$x^{\$} := 1^{|bin(n)|} 0 bin(n) x \tag{3.2}$$

samoograniczony kod dla x, gdzie $bin(n)$ jest binarną reprezentacją numeru n (jest to n–te słowo w ułożeniu leksykograficznym).

Przykład 3.1 Spróbujmy znaleźć samoograniczony kod dla słowa $x = 100010110$. Łatwo zauważyć, iż $n = |x| = 9$, tym samym $|bin(n)| = |bin('010')| = 3$, gdyż dziewiątym słowem w

ułożeniu słownikowym jest napis '010'. Tym samym $1^{|bin(9)|} = 111$. Korzystając ze wzoru (3.2) możemy już zatem napisać:

$$x^{\$} = 1^3 \;\; 0 \;\; 010 \;\; 100010110 = 1110010100010110.$$

Pokazaliśmy, iż 1110010100010110 jest samoograniczonym kodem dla słowa 100010110.

Nietrudno zauważyć, iż $|bin(n)| = [log_2(|n| + 1)]$. Analogicznie da się pokazać, iż

$$|x^{\$}| = 2[log_2(|n| + 1)] + |n| + 1 \tag{3.3}$$

gdzie $[x]$ oznacza największą liczbę całkowitą nie większą od x.
Stosując wzór (3.3), do przytoczonego wcześniej przykładu, mamy:

$$|x^{\$}| = 2[log_2(9 + 1)] + 9 + 1 = 2[log_2 10] + 10 = 2 \cdot 3 + 10 = 16$$

Otrzymany wynik rzeczywiście zgadza się z długością samoograniczonego kodu dla słowa 100010110.

3.1.2 Złożoność Kołmogorowa

Oznaczmy przez U uniwersalną maszynę, która oczekuje na wejściu formuł postaci $p^{\$}q$, gdzie $p, q \in \Sigma^*$. Teraz możemy dla $p, q \in \Sigma^*$ zdefiniować *warunkową złożoność Kołmogorowa* dla x jako:

$$C(x|q) := \min\{\, |p| : U(p^{\$}q) = x,\ p \in \Sigma^* \,\}$$

gdzie q jest informacją dodatkową. Analogicznie definiujemy *złożoność Kołmogorowa* dla x (bez informacji dodatkowej):

$$C(x) := \min\{\, |p| : U(p^{\$}) = x,\ p \in \Sigma^* \,\}$$

$C(x)$ jest długością najmniejszego "programu" zakodowanego za pomocą słów nad alfabetem $\{0, 1\}$ w pewnym kodowaniu, który zwraca słowo x. Program zostaje oczywiście uruchomiony na maszynie Turinga.
Z powyższych definicji wynika natychmiast, że

$$C(x|q) \leq C(x) \leq |x| + O(1) \tag{3.4}$$

tzn. złożoność warunkowa nie przekracza złożoności danego słowa gdyż posiadamy dodatkową informację (podpowiedź). Z drugiej strony słowo nie może być bardziej złożone niż program wypisujący po kolei wszystkie elementy słowa - długość takiego programu to $|x|$ plus pewna stała przeznaczona na kod instrukcji wypisujących wynik na wyjściu.

3.1.3 Liczby nieściśliwe

Dla wszystkich liczb naturalnych $n = 0, 1, 2, 3, \ldots$, oznaczmy przez $\underline{n}$, n-te słowo w Σ^* w porządku leksykalnym. Wprowadźmy, także $l(n) := |\underline{n}|$, czyli długość zapisu liczby n. Mówimy, że pewne słowo $x \in \Sigma^*$ jest ściśliwe jeżeli $C(x) < |x|$ (porównaj z ogólną zależnością (3.4)); inaczej dane słowo jest nieściśliwe. Analogicznie, liczba $n \in \mathbf{N}$ jest ściśliwa jeżeli jej n-te słowo w ułożeniu leksykalnym wyliczeniowym $\underline{n}$ jest ściśliwe.
Ponieważ liczb których zapis w alfabecie składającym się z dwóch symboli wynosi n jest dokładnie 2^n a liczb o krótszym zapisie jest tylko $2^n - 1$ dlatego z zasady szufladkowej Dirichleta

wynika, że dla każdego n muszą istnieć liczby o długości n które są nieściśliwe. Przykładowo zapisem liczby n może być n-te w porządku alfabetycznym słowo z Σ^*. Ponieważ pusty program nic nie wypisuje na wyjście więc liczby 1 oraz 2 (których zapis to odpowiednio '0' oraz '1') są nieściśliwe. Niewielkie liczby naturalne, których zapis składa się z zaledwie kilku znaków również są nieściśliwe, gdyż do ich wypisania potrzebne jest conajmniej tyle instrukcji ile ich zapis posiada znaków. Najlepszą metodą ich wypisania jest wypisanie ich znak po znaku. Program, który doszukiwałby się jakichś zależności w ich zapisie zająłby zdecydowanie więcej miejsca. Z drugiej jednak strony liczbę $n = 2^{30} - 1$ (zapis '00000000000000000000000000000') można ścisnąć. Wyobraźmy sobie program, w którym zapamiętana jest liczba 30 (ilość zer w zapisie liczby n). Zapis tej liczby zajmuje cztery znaki ('1111'). Oprócz tego program posiada instrukcje wypisujące na wyjście znak '0', instrukcje zmniejszające zapamiętaną liczbę oraz instrukcje wykonujące skok warunkowy. Całość programu powinna zająć mniej niż 30 znaków. Jest to więc liczba ściśliwa. Z powyższej uwagi możemy jednak wysnuć wniosek:

Dla każdego $n \in \mathbf{N}$ istnieje co najmniej jedno nieściśliwe słowo o długości n. Stąd też istnieje nieskończenie wiele liczb nieściśliwych.

Przykład 3.2 Weźmy liczbę naturalną $n = 466$, wówczas 466-tym słowem w ułożeniu leksykograficznym jest napis $\underline{n} = 11010011$. Nie trudno sprawdzić, korzystając ze wzoru (3.1), iż rzeczywiście zachodzi

$$G(11010011) = 2^8 - 1 + \Sigma_{i=0}^{7} x_i 2^i = 256 - 1 + 1 + 2 + 16 + 64 + 128 = 466$$

Stosując powyższe definicje zawarte w rozdziale, dla $n = 466$ mamy: $l(n) = |11010011| = 8$. Zgodnie z wcześniejszymi definicjami zawartymi w rozdziale 3.1.2, złożonością Kołmogorowa dla danego słowa x nazywamy długość najkrótszego programu, po uruchomieniu którego na maszynie deterministycznej Turinga, na wyjściu otrzymamy ten napis x. W naszym przypadku najkrótszy program, wypisujący tylko na wyjściu kolejne symbole słowa $\underline{n}$, potrzebuje co najmniej $|\underline{n}|$ instrukcji. Tym samym $C(n) \geq |\underline{n}|$, a zatem słowo $\underline{n}$ jest nieściśliwe. Zatem liczba $n = 466$ jest przykładem naturalnej liczby nieściśliwej.

3.2 Twierdzenie dla języków deterministycznych bezkontekstowych

Dla każdego języka $L \subseteq \Sigma^*$ i słowa $x \in \Sigma^*$, oznaczamy przez $x^{-1}L := \{v \in \Sigma^* \mid xv \in L\}$ zbiór słów v które rozszerzają napis x do słowa w L.

Oryginalne twierdzenie KC–DCF jest trudne do stosowania, dlatego przytaczamy tylko wniosek z twierdzenia, prostszy w zastosowaniach.

Jeżeli język $L \subseteq \Sigma^*$ jest deterministycznym językiem bezkontekstowym, $x, y \in \Sigma^*$, a c jest stałą, wówczas istnieje stała c' taka, że jeżeli $u, v, w \in \Sigma^*$, gdzie u jest sufiksem lewostronnie nieskończonego słowa $y^\omega x := ...yyx$, z następujących warunków:

- $C(v''|p_{uv'}) \leq c$ - tzn. dla każdego podziału $v = v'v''$, v'' może być odtworzone przy pomocy programu mającego co najwyżej c bitów jeżeli do dyspozycji mamy $(uv')^{-1}L$ w porządku leksykograficznym
- $C(w|p_{uv}) \leq c$ - tzn. w może być odtworzone przy pomocy programu mającego co najwyżej c bitów jeżeli do dyspozycji mamy $(uv)^{-1}L$ w porządku leksykograficznym

- $C(v) \geq 2 \cdot l(l(|u|))$

wynika, że $C(w) \leq c'$.

3.3 Przykłady zastosowań twierdzenia KC–DCFL

Przykład 3.3 Zbiór palindromów
Niech $L = \{ x \mid x = x^R, x \in \{0,1\}^* \}$. Załóżmy, że język ten jest deterministyczny, niech $y = 0, x = 1, u = 0^n 1$ oraz $v = 0^n$. v składa się więc z n zer, więc jego złożoność Kołmogorowa sprowadza się do zapisania liczby n (plus stały współczynnik). Mamy więc:

$C(v) = C(\underline{n}) + O(1)$

Załóżmy, że n jest nieściśliwe, wówczas z definicji nieściśliwości mamy

$C(\underline{n}) \geq l(n)$

Niech $s = b_{k-1}b_{k-2}\ldots b_1 b_0$ będzie zapisem liczby n oraz niech $k = l(n)$ oznacza długość. Wówczas

$n = \sum_{i=0}^{k-1}(1 + b_i)2^i = 2^k - 1 + \sum_{i=0}^{k-1} b_i 2^i$

Podstawiając za b_i same zera, a potem same jedynki otrzymujemy:

$2^k - 1 \leq n \leq 2(2^k - 1)$

A stąd wynika, że $k \leq log(n+1) < k+1$, czyli $log(n+1) - 1 < l(n) \leq log(n+1)$.

Oznacza to, że funkcja długości zapisu liczby n tj. funkcja l ma charakter logarytmiczny. Zatem dla odpowiednio dużych n będzie zachodziła równość:

$n \geq l(n)$

Oznacza to również, że dla odpowiednio dużych n będzie zachodziło:

$n \geq l(n+1)^2$

Ostatecznie otrzymujemy więc

$C(v) \geq l(n) \geq l(l(n+1)^2) = 2 \cdot l(l(n+1)) = 2 \cdot l(l(|u| - 1 + 1)) = 2 \cdot l(l(|u|))$

co spełnia warunek 3. Dla każdego podziału $v = v'v''$, v'' jest pierwszym leksykograficznie słowem z $(uv')^{-1}L$. Możemy więc wziąć program, który zwróci pierwsze słowo z $(uv')^{-1}L$. Ponieważ składa się ono z samych zer złożoność programu wypisującego nie będzie rosła w nieskończoność wraz ze wzrostem długości v''.

Jeśli zer do wypisania jest $0 < x < n$ mogłoby się wydawać, że aby to uczynić potrzebujemy programu takiego jak :

```
for(int i = 0 ; i < x ; i++)
    write('0');
```

którego złożoność Kołmogorowa (ze względu na zapisaną w kodzie wartość liczby x) ewidentnie rośnie jak $l(x)$, a przy n rosnącym do nieskończoności, również x oraz $l(x)$ rosną do nieskończoności.

Słowa tego nie musimy jednak generować od zera. Należy pamiętać, że program otrzymuje na wejście zbiór $(uv')^{-1}L$, którego jest ono elementem. Wystarczy więc je przepisać, wraz z ewentualnym sprawdzeniem, czy jest ono właśnie postaci 0^x. Ewentualnym, gdyż w tym akurat przypadku możemy to pominąć zważywszy na fakt, że będzie to pierwsze słowo owego zbioru. Wypisanie będzie więc postaci:

```
char c;

do{
    write(c = read());
} while(c != '$'); // oznaczenie końca słowa podanego na wejście
```

W takim przypadku długość programu, a zatem i jego złożoność Kołmogorowa, nie zależy zupełnie od długości słowa v''.
Warunek 1 jest więc spełniony. Jeżeli teraz weźmiemy program, który zwróci pierwsze słowo z $(uv)^{-1}L$ to warunek 2 będzie również spełniony, a zwróconym słowem będzie $w = 10^n$. Słowo to również jest łatwe do wypisania, składa się z jedynki, a następnie z samych zer. Zatem złożoność programu wypisującego nie rośnie w nieskończoność. Wiemy natomiast, że $C(w) = C(\underline{n}) + O(1) \geq l(n)$, co jest sprzeczne z tezą lematu. Zatem język L nie jest deterministycznym językiem bezkontekstowym.

Przykład 3.4 Zbiór słów z jedynką w drugiej połówce
Niech $L = \{\, xy \mid x, y \in \{0,1\}^*,\ |x| = |y|,\ y \text{ zawiera } '1' \,\}$. Załóżmy, że język ten jest deterministyczny. Niech $y = 0, x = 1, u = 0^n1$, gdzie $|u|$ jest parzyste oraz $v = 0^{n+1}$. Jak wyżej, dla nieściśliwego n mamy $C(v) = C(\underline{n}) + O(1) \geq l(n)$, co dla odpowiednio dużych n spełnia warunek 3. Dla każdego podziału $v = v'v''$ potrzebujemy tylko jednego bitu informacji mówiącego czy v' jest parzyste, czy też nie, aby otrzymać v''. Będzie to pierwsze słowo składające się z samych 0 o parzystości długości takiej samej jak v' nie należące do $(uv')^{-1}L$. Rozważmy program generujący ciągi zer o długości parzystej (albo nieparzystej) i sprawdzający, czy należą one do $(uv')^{-1}L$. Pierwsze nie należące słowo jest zwracane. Program taki spełnia warunek 1. Program generujący słowa w porządku leksykograficznym, sprawdzający czy należą one do $(uv)^{-1}L$ i zwracający pierwsze nie należące słowo do $(uv)^{-1}L$ zaczynające się od '1' spełnia natomiast warunek 2. Zwróci on wartość $w = 10^{2n+3}$. Wiemy natomiast, że $C(w) = C(\underline{n}) \geq l(n)$, co jest sprzeczne z tezą lematu. Zatem język L nie jest deterministycznym językiem bezkontekstowym.

Wybieranie pierwszego słowa nie należącego do zbioru podanego na wejście

W przypadku powyższego języka na pewnym etapie rozwiązania pojawia się program zwracający pierwsze słowo o określonych własnościach nie należące do zbioru podanego na wejście. Przykładowo w przypadku wyznaczania słowa v'' należy generować słowa składające się z parzystej (bądź nieparzystej) liczby zer i sprawdzać czy nie należą one do zbioru $(uv')^{-1}L$ podanego na wejście. Program będzie więc za pomocą odpowiedniej funkcji generował ciągi zer. Przykładowa funkcja w języku C++ wygląda następująco:

```
char word[1024*1024];  // duży obszar pamięci - najlepiej nieograniczony

char *gen(bool even) { // even - czy parzysta długość słowa
    static char *ptr = word;
```

```
    if(ptr == word) {  // przy pierwszym wywołaniu

        if(even) // jesli parzysta to pierwszym słowem będzie '00', w przeciwnym razie '0'
            *ptr++ = '0';
        *ptr++ = '0';
    }
    else { // za każdym kolejnym wywołaniem dopisujemy dwa zera
        *ptr++ = '0';
        *ptr++ = '0';
    }
    *ptr = '$'; // oznaczenie końca słowa
    return word;
}
```

Przy generowaniu słów o nieparzystej długości oraz przy wejściu (zbiorze $(uv')^{-1}L$) uporządkowanym leksykograficznie działanie programu polega na:

1. wygenerowaniu poprzez wywołanie `gen(false)` słowa o nieparzystej długości składającego się z samych zer (przy pierwszym wywolaniu - '0')
2. wczytaniu pierwszego słowa z wejścia
3. jeśli te dwa słowa nie są równe, zwróceniu wygenerowanego słowa. STOP.
4. jeśli te słowa były równe powrót do punktu 1., a więc wygenerowanie kolejnego słowa oraz wczytanie kolejnego słowa z wejścia

Obszar pamięci `word` pełni rolę taśmy i jego rozmiar nie wpływa na złożoność Kołmogorowa programu.
Przy wyznaczaniu słowa w również na wejście dostajemy uporządkowane leksykograficznie słowa należące do zbioru $(uv)^{-1}L$. W tym przypadku potrzebujemy jednak funkcji generującej wszystkie możliwe słowa alfabetu zerojedynkowego w porządku leksykograficznym. Przykładowa implementacja takiej funkcji to:

```
char word[1024*1024];

char *genall(void) {
    static char *ptr = word;
    char *digit = ptr - 1;

    for(;;) {
        if(digit < word) {
            *ptr++ = '0';
            *ptr = '$';
            break;
        }
        if(*digit == '0') {
            *digit = '1';
```

```
            break;
        }
        else {
            *digit-- = '0';
        }
    }
    return word;
}
```

Przykład 3.5 Zbiór słów z alfabetu trzyznakowego
Niech $L = \{ 0^i 1^j 2^k \mid i, j, k \geq 0,\ i = j \text{ lub } j = k \}$. Załóżmy, że język ten jest deterministyczny. Niech $y = 0, x = 0, u = 0^n$ oraz $v = 1^n$. Dla nieściśliwego n mamy spełniony warunek 3. Dla każdego podziału $v = v'v''$, v'' jest pierwszym leksykograficznie słowem z $(uv')^{-1}L$. Program zwracający pierwsze słowo z $(uv')^{-1}L$ spełnia więc warunek 1. Pierwszym leksykograficznie niepustym słowem należącym do $(uv)^{-1}L$ jest 12^{n+1}. Program zwracający pierwsze słowo, czyli $w = 12^{n+1}$ spełnia więc warunek 2, natomiast ponownie otrzymujemy sprzeczność z tezą lematu. Zatem język L nie jest deterministycznym językiem bezkontekstowym.

Przykład 3.6 Dopasowywanie wzorca
Niech $L = \{ x\#yx^Rz \mid x, y, z \in \{0,1\}^* \}$. Załóżmy, że język ten jest deterministyczny. Niech $y = 1, x = \#, u = 1^n\#$ oraz $v = 1^{n-1}0$. Dla nieściśliwego n mamy spełniony warunek 3. Dla każdego podziału $v = v'v''$, v'' da się odtworzyć z pierwszego leksykograficznie słowa z $(uv')^{-1}L$, jeśli weźmiemy program, który wczytuje pierwsze leksykograficznie słowo z $(uv')^{-1}L$, a następnie zamieni ostatni jego znak z '1' na '0'. Warunek 1. jest więc spełniony. Pierwszym leksykograficznie niepustym słowem należącym do $(uv)^{-1}L$ jest 1^n, więc $w = 1^n$ spełnia warunek 2. Kolejny raz otrzymujemy jednak sprzeczność z tezą lematu. Zatem język L nie jest deterministycznym językiem bezkontekstowym.

Przykład 3.7 Język złożony z antypalindromów
Niech $L = \{ x \in \{0,1\}^* \mid x \neq x^R \}$. Załóżmy, że język ten jest deterministyczny. Niech $y = 0,\ x = 1,\ u = 0^n 1$ oraz $v = 0^n$. Dla nieściśliwego n mamy spełniony warunek 3. Dla każdego podziału $v = v'v''$, v'' jest pierwszym leksykograficznie słowem nie należącym do $(uv')^{-1}L$. Możemy więc wziąć program, który będzie generował w porządku leksykograficznym ciągi i zwróci pierwsze słowo nie należące do $(uv')^{-1}L$. Warunek 1. jest więc spełniony. Jeżeli teraz weźmiemy program, który zwróci pierwsze słowo nie należące do $(uv)^{-1}L$ to warunek 2 będzie również spełniony, a zwróconym słowem będzie $w = 10^n$. Wiemy natomiast, że $C(w) = C(\underline{n}) + O(1) \geq l(n)$ co jest sprzeczne z tezą lematu. Zatem język L nie jest deterministycznym językiem bezkontekstowym.

Przykład 3.8 Język antyDycka nad alfabetem { (,)}
Język ten w oczywisty sposób jest deterministycznym językiem bezkontekstowym, gdyż da się skonstruować deterministyczny automat ze stosem akceptujący słowa tego języka. Automat napotykając znak '(' przepisuje go na stos, napotykając znak ')' zdejmuje ze stosu odpowiadający mu znak '(', albo jeśli stos jest pusty przechodzi do stanu akceptującego. Po przeczytaniu całego wejścia jeżeli stos nie jest pusty, to automat przechodzi do stanu akceptującego, w przeciwnym razie wczytane słowo nie zostaje zaakceptowane.

Możemy sprawdzić, że przy rozpatrywaniu tego języka nie otrzymujemy sprzeczności z tezą lematu. Niech $y = (,\ x =),\ u = (^n)$ oraz $v =)^{n-1}$. Dla nieściśliwego n mamy spełniony warunek 3. Dla każdego podziału $v = v'v''$, v'' jest pierwszym leksykograficznie słowem nie należącym do $(uv')^{-1}L$. Możemy więc wziąć program, który będzie generował ciągi nawiasów w porządku leksykograficznym i zwróci pierwsze słowo nie należące do $(uv')^{-1}L$ i mamy spełniony warunek 1. Jeżeli teraz weźmiemy program, który będzie generował ciągi nawiasów w porządku leksykograficznym i zwróci pierwsze słowo nie należące do $(uv)^{-1}L$ to warunek 2. będzie również spełniony, a zwróconym słowem będzie $w = ()$. Dla takiego w jest spełniony warunek: $C(w) \leq c'$, gdyż $C(w) \leq |w| + O(1) = 2 + O(1)$.

Przykład 3.9 Język słów nie będących postaci ww
Niech $L = \{0,1\}^* - \{\, ww \mid w \in \{0,1\}^* \,\}$. Załóżmy, że język ten jest deterministyczny. Niech $y = 0,\ x = 1,\ u = 0^n1$ oraz $v = 0^{n-1}1$. Dla nieściśliwego n mamy spełniony warunek 3. Dla każdego podziału $v = v'v''$, v'' da się otrzymać z pierwszego leksykograficznie słowa postaci ...001 nie należącego do $(uv')^{-1}L$ poprzez zamianę przedostatniego znaku z 0 na 1 oraz usunięcie znaku ostatniego więc warunek 1 jest spełniony. Program przechodzący po znakach 0 na przedostatni znak gdy ostatnim znakiem jest 1 nie jest duży. Modyfikacje znakowe to także tylko kilka instrukcji. Rozmiar kodu przepisującego zmodyfikowany ciąg również nie zależy od jego długości. Dlatego możemy twierdzić, że rozmiar takiego programu jest ograniczony. Dalej, pierwszym leksykograficznie, niepustym słowem nie należącym do $(uv)^{-1}L$ jest $0^n10^{n-1}1$, więc $w = 0^n10^{n-1}1$ spełnia warunek 2. Kolejny raz otrzymujemy więc sprzeczność z tezą lematu. Zatem język L nie jest deterministycznym językiem bezkontekstowym.

4 Własności zamkniętości

Własności zamkniętości stanowią odrębną technikę pozwalającą wnioskować o umiejscowieniu języka w hierarchii Chomsky'ego. Stosowane są także pomocniczo przy dowodach wykorzystujących twierdzenia zaprezentowane w poprzednich rozdziałach.
Oznaczmy przez REG - klasę języków regularnych, LIN - liniowych, DCFL - deterministycznych bezkontekstowych, CFL - bezkontekstowych, CSL - kontekstowych, RE - rekurencyjnie przeliczalnych.
Przemieszanie między słowami definiujemy w sposób rekurencyjny następująco:
$u \text{ ш } \epsilon = \epsilon \text{ ш } u = \{u\}$
$(au \text{ ш } bv) = a(u \text{ ш } bv) \cup b(au \text{ ш } v)$
Działanie rozszerza się w naturalny sposób na języki:
$L_1 \text{ ш } L_2 = \cup_{u \in L_1, v \in L_2}(u \text{ ш } v)$
Własności zamkniętości poszczególnych klas języków przedstawia poniższa tabela (patrz m.in. [8]).

	RE	CS	CFL	DCFL	LIN	REG
suma	+	+	+	−	+	+
iloczyn	+	+	−	−	−	+
dopełnienie	−	+	−	+	−	+
konkatenacja	+	+	+	−	−	+
$\star$ Kleene'go	+	+	+	−	−	+
przecięcie z językiem regularnym	+	+	+	+	+	+
homomorfizm	+	−	+	−	−	+
homomorfizm bez ϵ	+	+	+	−	−	+
podstawienie	+	−	+	−	+	+
podstawienie bez ϵ	+	+	+	−	+	+
odwrotny morfizm	+	+	+	+?	+	+
iloraz lewostronny/prawostronny	+	−	−	−/?	−	+
iloraz lewostronny/prawostronny z językiem regularnym	+	−	+	−/+?	+	+
przemieszanie (III)	+	+	−	−	−	+
odbicie lustrzane	+	+	+	−	+	+

Przykład 4.1 Własności zamkniętości klasy języków deterministycznych bezkontekstowych. Wykorzystany zostanie fakt, że języki $\{ a^n b^n \} \cup \{ a^n b^{2n} \}$, $\{ a^n b^n x \} \cup \{ a^n b^{2n} y \}$, nie są językami deterministycznymi bezkontekstowymi natomiast $\{ xa^n b^n \} \cup \{ ya^n b^{2n} \}$ jest językiem deterministycznym bezkontekstowym.

- Brak zamkniętości ze względu na homomorfizm.
  Weźmy język $L = \{ xa^{n-1} b^n \} \cup \{ ya^{n-1} b^{2n} \} \in DCFL$ i homomorfizm h zdefiniowany następująco: $h(x) = h(a) = a$, $h(y) = h(b) = b$. Język $h(L) = \{ a^n b^n \} \cup \{ a^n b^{2n} \}$ nie jest DCFL.
- Brak zamkniętości ze względu na iloraz lewostronny.
  Weźmy język $L = \{ xa^n b^n \} \cup \{ ya^n b^{2n} \} \in DCFL$. Język $\{x, y\}^{-1} L = \{ a^n b^n x \} \cup \{ a^n b^{2n} y \}$ nie jest DCFL.
- Brak zamkniętości ze względu na przemieszanie.
  Weźmy język $L = \{ a^n b^n \} \in DCFL$ i przemieszajmy go z samym sobą. Język $L' = L \amalg L = \{ a^n w b^n \mid |w|_a = |w|_b = n \}$ nie jest DCFL. Rzeczywiście, $L' \cap a^* b^* a^* b^* = \{ a^{n+k} b^{n-k} a^{n-k} b^{n+k} \mid n \geq 0, n \geq k \geq 0 \}$ nie jest bezkontekstowy, co można łatwo wykazać z lematu o pompowaniu.
- Brak zamkniętości ze względu na odbicie lustrzane.
  Weźmy język $L = \{ xb^n a^n \} \cup \{ yb^{2n} a^n \} \in DCFL$. Język $L^R = \{ a^n b^n x \} \cup \{ a^n b^{2n} y \}$ nie jest DCFL.

Przykład 4.2 Własności zamkniętości klasy języków LL.

- Brak zamkniętości ze względu na dopełnienie.
  Zachodzi twierdzenie: dopełnienie języka $LL(k)$, który nie jest regularny nigdy nie jest $LL(k)$. $L_1 = \{ a^m b^n \mid 1 \leq m < n \}$ jest językiem $LL(1)$, który nie jest regularny, a więc jego dopełnienie L_2 nie jest $LL(k)$.
- Brak zamkniętości ze względu na sumę.
  Język $L_3 = \{ a^n b^n \mid n \geq 1 \} \cup \{ a^n c^n \mid n \geq 1 \}$ nie jest $LL(k)$, jednakże jest sumą dwóch języków $LL(1)$: $L_4 = \{ a^n b^n \mid n \geq 1 \}$ oraz $L_5 = \{ a^n c^n \mid n \geq 1 \}$.
- Brak zamkniętości ze względu na iloczyn.
  Języki $L_6 = \{ a^n (b + c)^n \mid n \geq 1 \}$ oraz $L_7 = \{ a^n b^m \mid n, m \geq 1 \} \cup \{ a^n c^m \mid n, m \geq 1 \}$ są

językami $LL(1)$ ale ich iloczyn to L_3, który nie jest $LL(k)$.

- Brak zamkniętości ze względu na odbicie lustrzane.
  Język $L_8 = \{ b^n a^n \mid n \geq 1 \} \cup \{ c^n a^n \mid n \geq 1 \}$ jest $LL(1)$, jego odbicie to L_3 i nie jest $LL(k)$.
- Brak zamkniętości ze względu na konkatenację.
  $L_9 = \{ a^m b^n \mid 1 \leq n \leq m \}$ nie jest $LL(k)$ gdyż jego suma z językiem $LL(1)$ L_1 jest językiem regularnym: $L_9 \cup L_1 = \{ a^m b^n \mid m, n \geq 1 \}$ (na podstawie twierdzenia: jeśli skończona suma rozłącznych języków $LL(k)$ jest regularna to wszystkie języki tworzące sumę są regularne). Z drugiej strony L_9 jest konkatenacją dwóch języków $LL(1)$: a^* oraz $\{ a^n b^n \mid n \geq 1 \}$. Tak więc języki $LL(k)$ nie są zamknięte ze względu na konkatenację.
- Brak zamkniętości ze względu na homomorfizm.
  Język $L_{10} = \{ da^m b^{m+1} \mid m \geq 0 \} \cup \{ ea^m c^{m+1} \mid m \geq 0 \}$ jest $LL(1)$ ale jego obraz poprzez homomorfizm zamieniający d i e na a oraz nie zamieniający a, b, c to L_3, który nie jest $LL(k)$.

Literatura

[1] Gastin Paul, *Langages formels, calculabilité, complexité et analyse d'algorithmes*, LIAFA, Université Paris 7.

[2] Glier O., *Kolmogorov Complexity and deterministic context–free languages*, SIAM J. Comput., 32(2003), pp. 1389-1394.

[3] Hopcroft J.E., Ullman J.D. *Wprowadzenie do teorii automatów, języków i obliczeń*, PWN, 2000.

[4] Paul M.B. Vitányi, Ming Li *A New Approach to Formal Language Theory by Kolmogorov Complexity*, SIAM J. Comput., 24(1995), pp. 398-410.

[5] Moore Cristopher, Lakdawala Porus *Queues, Stacks, and Transcendentality at the Transition to Chaos*, Santa Fe Institute, Santa Fe, Horni Bhabha Center for Science Education, Mumbai, 1998

[6] Papadimitriou Christos H., *Złożoność obliczeniowa*, Warszawa, Wydawnictwo Naukowo–Techniczne, 2002.

[7] Ramos-Jiménez G., López-Muñoz J., Morales-Bueno R., *Comparison of Parkikh's condition to other conditions for context-free languages*, Theoretical Computer Science 202(1998), pp. 231-244.

[8] Salomaa Arto, Rozenberg Grzegorz, *Handbook of Formal Languages*

[9] Sheng Yu, *A pumping lemma for deterministic context–free languages*, Information Processing Letters, vol. 31, 1989