Języki, algorytmy i obliczenia
Rozwiązania zadań z gwiazdką - seria 2

Paweł Parys (parys@mimuw.edu.pl)

25 stycznia 2005

Notacja

Przez $w_1, w_2, \ldots, w_{|w|}$ oznaczam kolejne litery słowa w. Przez $\lambda x.wyrażenie$ oznaczam funkcję, która dla argumentu x przyjmuje wartość *wyrażenie*.

Zadanie 1

Teza zadania nie jest prawdziwa. Weźmy jakikolwiek automat, który akceptuje słowo puste. Wówczas dla słowa pustego musi on wykonać przynajmniej jeden krok, czyli dla żadnej stałej $|C|$ liczba kroków nie będzie $\leqslant C \cdot 0$.

Udowodnię podobne twierdzenie, z tym że nierówność będzie zachodzić tylko dla słów niepustych. Ustalmy konkretny automat A. Niech Q będzie jego zbiorem stanów, Γ zbiorem symboli stosowych, natomiast δ_{max} niech będzie taką stałą, że każde przejście automatu A powoduje odłożenienie na stos co najwyżej δ_{max} symboli. Weźmy $C_1 = \delta_{max}(|Q|^2|\Gamma| + 2)$, $C_2 = 4C_1 + 2\delta_{max} + 4$, $C_3 = |Q| \cdot (|\Gamma| + 1)^{C_1+\delta_{max}+1}$, $C_4 = 2 + C_3(2 + 6\delta_{max} + C_1)$, $C = 1 + C_4 + C_2C_4$. Weźmy dowolne niepuste słowo $w \in Z(A)$. Rozważmy najkrótsze obliczenie automatu A akceptujące to słowo. Udowodnię, że liczba kroków w tym obliczeniu jest nie większa niż $C|w|$. Ponumerujmy kroki naszego obliczenia przez $1, 2, \ldots, n$. Niech $s(t)$ będzie wysokością stosu w kroku t. Przez wysokość stosu w jakimś kroku rozumiem wysokość stosu po zdjęciu przez automat symbolu odczytywanego w tym kroku, a przed włożeniem nowych symboli na stos. Pozycje stosu również numerujemy kolejno: $1, 2, \ldots$ zaczynając od dołu. Przez $p(i, t)$ będziemy oznaczać ostatni numer kroku przed krokiem t, w którym został położony symbol na pozycji stosu i. Analogicznie, przez $z(i, t)$ będziemy oznaczać pierwszy numer kroku po kroku t, w którym został zdjęty symbol z pozycji stosu i. Niech $r_1, r_2, \ldots, r_{|w|}$ będą numerami kroków, w których odczytywaliśmy symbol z wejścia. Niech $m_0 = 1, m_{|w|} = n$ oraz niech m_k dla $k = 1, 2, \ldots, |w|-1$ będzie numerem tego wśród kroków $r_k, r_k+1, \ldots, r_{k+1}$, w którym wysokość stosu była najmniejsza (pierwszym z takich kroków jeśli było kilka). Powiemy, że w danym kroku t przedział $[i, j]$ pozycji stosu jest *czysty*, jeśli żaden z kroków $p(i, t), p(i, t) + 1, \ldots p(j, t)$ oraz żaden z kroków $z(j, t), z(j, t) + 1, \ldots, z(i, t)$

nie wczytywał litery z wejścia (uwaga! nie w każdym z wymienionych kroków wkładaliśmy lub zdejmowaliśmy któryś z symboli leżących na stosie w kroku t). W ustalonym kroku t pozycję stosu i nazwiemy *podstawową*, jeśli była najniższą z pozycji stosu zajętych w kroku $p(i,t)$.

Lemat 1 *Niech t będzie dowolnym krokiem, $[i,j]$ dowolnym przedziałem pozycji na stosie w tym kroku. Jeśli przedział ten jest czysty, to jego długość wynosi co najwyżej C_1.*

Dowód Ustalmy krok t oraz przedział $[i,j]$. Załóżmy wbrew tezie, że jest on czysty i że jego długość jest większa niż $C_1 = \delta_{max}(|Q|^2|\Gamma| + 2)$ i dojdziemy do sprzeczności. Niech k będzie dowolną pozycją z tego przedziału. Zauważmy, że najniższy symbol położony w kroku $p(k,t)$ nadal leży na stosie w kroku t, ponieważ leży on głębiej niż k, więc nie mógł być zdjęty wcześniej. Oznacza to, że na stosie może leżeć co najwyżej $\delta_{max} - 1$ pozycji niepodstawowych pod rząd. Zatem w naszym przedziale jest co najmniej $|Q|^2|\Gamma| + 1$ pozycji podstawowych. Każdej z tych pozycji przypiszemy pewną trójkę: Pozycji k przypiszemy trójkę (q_0, a_0, q_1), gdzie q_0 jest stanem przed wykonaniem kroku $p(k,t)$, a_0 symbolem na szczycie stosu przed wykonaniem kroku $p(k,t)$, natomiast q_1 stanem po wykonaniu kroku $z(k,t)$. Zauważmy, że jest tylko $|Q|^2|\Gamma|$ różnych trójek, natomiast rozważamy aż $|Q|^2|\Gamma| + 1$ pozycji podstawowych. Zatem pewne dwie pozycje k, l $(k < l)$ mają przypisaną tą samą trójkę (q_0, a_0, q_1). Rozważmy obliczenie, w którym nie będzie kroków $p(k,t), p(k,t)+1, \ldots, p(l,t)-1$ oraz $z(l,t)+1, z(l,t)+2, \ldots, z(k,t)$. Zauważmy, że to obliczenie jest krótsze. Przeczy to wyborowi obliczenia. Pozostaje uzasadnić, że jest to poprawne obliczenie. Ponieważ k, l wybraliśmy z przedziału czystego, to usunięte kroki nie wczytywały nic z wejścia. Stan przed krokami $p(k,t)$ i $p(l,t)$ jest taki sam, bo jest to q_0. Tak samo stan po wykonaniu $z(l,t)$ i $z(k,t)$, bo jest to q_1. Obliczenia od $p(l,t)$ do $z(l,t)$ w ogóle nie patrzą w głąb stosu, zależą tylko od pozycji na szczycie stosu, ale zarówno przed $p(l,t)$, jak i przed $p(k,t)$ leży tam symbol a_0. Tym bardziej obliczenia te nie modyfikują stosu poniżej pozycji l. Również obliczenia usuwane nie modyfikują stosu poniżej pozycji k. Oznacza to, że usunięcie omawianego kawałka obliczenia nie wpływa na możliwość wykonania reszty, co kończy dowód.

Lemat 2 *Niech $[i,j]$ będzie dowolnym przedziałem pozycji stosu w pewnym kroku m_k. Jeśli żaden spośród kroków $p(i,m_k), p(i,m_k)+1, \ldots, p(j,m_k)$ (tutaj są wszystkie kroki po kolei) oraz $z(j,m_k), z(j-1,m_k), \ldots, z(i,m_k)$ (a tutaj tylko kroki, w których zdjęliśmy ze stosu coś z naszego przedziału) nie jest jednym z kroków $m_0, m_1, \ldots, m_{|w|}$, to przedział $[i,j]$ jest czysty.*

Przyjmijmy prawdziwość założeń i załóżmy, że teza nie jest prawdziwa. Najpierw rozważmy przypadek, gdy że dla pewnego l mamy $p(i,m_k) \leqslant r_l \leqslant p(j,m_k)$. Popatrzmy ile może być równe m_l. Zgodnie z definicją $m_l \geqslant r_l$. Założyliśmy, że m_l nie jest żadnym z kroków $p(i,m_k), p(i,m_k)+1, \ldots, p(j,m_k)$, czyli $m_l > p(j,m_k)$. Ponadto oczywiście $m_l \leqslant m_k$ (bo $l \leqslant k$). Oznacza to, że $s(m_l) > j$, bo po położeniu symbolu na j-tej pozycji stosu w kroku $p(j,m_k)$ (zgodnie z definicją $p(j,m_k)$) leży on tam na pewno co najmniej do kroku

m_k. Z definicji m_l wynika, że w kroku m_l wysokość stosu była najmniejsza między r_l i r_{l+1}, nie może być więc większa niż wysokość w kroku $p(j, m_k)$, która to jest mniejsza niż j. Sprzeczność.

Pozostaje jeszcze możliwość, że dla pewnego l mamy $z(j, m_k) \leqslant r_l \leqslant z(i, m_k)$. Weźmy najmniejsze z l spełniających ten warunek (oznacza to, że $r_{l-1} < z(j, m_k)$). Popatrzmy ile może być równe m_{l-1}. Rozważmy dwa przypadki:

- $m_{l-1} < z(j, m_k)$. Wtedy podobnie jak poprzednio: Oczywiście $m_{l-1} \geqslant m_k$ (bo $l-1 \geqslant k$). Oznacza to, że $s(m_{l-1}) > j$, bo (zgodnie z definicją $z(j, m_k)$) symbol na j-tej pozycji stosu z chwili m_k leży tam aż w kroku $z(j, m_k)$. Z definicji m_{l-1} wynika, że w kroku m_{l-1} wysokość stosu była najmniejsza między r_{l-1} i r_l, nie może być więc większa niż wysokość w kroku $z(j, m_k)$, która to jest równa $j-1$. Sprzeczność.

- $z(j, m_k) \leqslant m_{l-1}$ Niech c będzie najmniejszą z tych liczb $i \leqslant c \leqslant j$, że $z(c, m_k) \leqslant m_{l-1}$. Zgodnie z definicją m_{l-1} mamy $s(m_{l-1}) \leqslant s(z(c, m_k)) = c-1$. Jeśli $c = i$, to mamy $z(c, m_k) \leqslant m_{l-1} \leqslant r_l \leqslant z(c, m_k)$, czyli $z(c, m_k) = m_{l-1}$. Jeśli $c > i$, to: Zauważmy, że $s(m_{l-1}) \geqslant c-1$, bo na pozycji stosu $c-1$ cały czas od kroku m_k leży symbol, który zostanie ściągnięty dopiero w kroku $z(c-1, m_k)$, więc w kroku m_{l-1} który jest pomiędzy rozmiar stosu nie może zejść poniżej $c-1$. Oznacza to że $s(m_{l-1}) = c-1$. Wiemy też, że m_{l-1} jest pierwszym z kroków pomiędzy r_{l-1} i r_l, dla którego to zachodzi. Ale $z(c, m_k)$ również leży pomiędzy r_{l-1} i r_l i $s(z(c, m_k)) = c-1$, co oznacza, że $z(c, m_k) \geqslant m_{l-1}$, czyli w tym przypadku również $z(c, m_k) = m_{l-1}$. To jednak jest sprzeczne z założeniem, że m_{l-1} nie jest żadnym z kroków $z(j, m_k), z(j-1, m_k), \ldots, z(i, m_k)$.

Lemat 3

$$\sum_{i=1}^{|w|} |s(m_i) - s(m_{i-1})| \leqslant C_2|w|$$

Dowód Niech R będzie zbiorem tych i, że $s(m_i) > s(m_{i-1})$. Udowodnię, że

$$\sum_{i \in R} |s(m_i) - s(m_{i-1})| \leqslant (2C_1 + \delta_{max} + 2)|w| = \frac{C_2}{2}|w|$$

Z tej nierówności natychmiast wynika teza lematu, gdyż suma po pozostałych elementach jest równa tej sumie (bo $s(m_0) = s(m_{|w|}) = 0$).

Niech R' będzie zbiorem tych k, że $s(m_{k-1}) + \delta_{max} + 1 \leqslant s(m_k)$. Dla każdego $k \in R'$ popatrzmy na przedział $[i, j] = [s(m_{k-1}) + \delta_{max} + 1, s(m_k)]$ w kroku m_k. Na pewno żaden spośród kroków $p(i, m_k), p(i, m_k) + 1, \ldots, p(j, m_k)$ nie jest którymś z $m_0, m_1, \ldots, m_{|w|}$, bo między m_{k-1} i m_k nie ma już żadnego. Załóżmy, że wśród $z(j, m_k), z(j-1, m_k), \ldots, z(i, m_k)$ jest dokładnie c_k z kroków $m_0, m_1, \ldots, m_{|w|}$. Udowodnię, że wówczas przedział $[i, j]$ może mieć długość co najwyżej $(C_1+1)(c_k+1)$. Możemy bowiem przedział $[i, j]$ podzielić na c_k+1 takich przedziałów $[i = i_1, j_1], [i_2 = j_1 + 2, i_3], \ldots [i_{c_k+1} = j_{c_k} + 2, j_{c_k+1} = j]$ (z których być może niektóre są puste), że dla każdego a wśród $z(j_a, m_k), z(j_a - 1, m_k), \ldots, z(i_a, m_k)$ nie

ma już żadnego z kroków $m_0, m_1, \ldots, m_{|w|}$. Zgodnie z lematem 2 każdy z tych przedziałów jest czysty. Jeśli przedział $[i, j]$ miałby długość większą niż $(C_1+1)(c_k+1)$, to suma długości tych małych przedziałów byłaby większa niż $C_1(c_k+1)$, czyli któryś z nich miałby długość większą niż C_1. Jest to jednak sprzeczne z lematem 8.

Zauważmy jednak, że każda spośród liczb $m_0, m_1, \ldots, m_{|w|}$ może być jedną z $z(j, m_k)$, $z(j+1, m_k), \ldots, z(i, m_k)$ tylko dla jednego m_k (bo są to kroki, w których zdejmujemy ze stosu coś, co położyliśmy między krokiem m_{k-1} i m_k). Inaczej mówiąc suma wszystkich c_k jest nie większa niż $|w|+1$. Oznacza to, że (w ostatnim przejściu wykorzystujemy nierówność $|R'| \leqslant |R| \leqslant |w|-1$):

$$\begin{aligned}
\sum_{k \in R} |s(m_k) - s(m_{k-1})| &\leqslant |R|\delta_{max} + \sum_{k \in R'} (s(m_k) - s(m_{k-1}) - \delta_{max}) \leqslant \\
&\leqslant |R|\delta_{max} + \sum_{k \in R'} (c_k + 1)(C_1 + 1) \leqslant \\
&\leqslant |R|\delta_{max} + (|R'| + |w| + 1)(C_1 + 1) \leqslant \\
&\leqslant |w|(\delta_{max} + 2C_1 + 2)
\end{aligned}$$

Lemat 4 *Jeśli pomiędzy krokiem a i b (włącznie) nie wczytujemy nic z wejścia oraz dla każdych dwóch kroków c, d takich, że $a \leqslant c \leqslant d \leqslant b$, zachodzi $|s(c) - s(d)| \leqslant C_1 + \delta_{max}$, to $b - a + 1 \leqslant C_3$.*

Dowód Załóżmy, że $b - a + 1 > C_3$ i dojdziemy do sprzeczności. Niech s_{min} i s_{max} oznaczają minimalną i maksymalną wysokość stosu podczas kroków od a do b. Niech $S = \{s_{min}+1, s_{min}+2, \ldots, s_{max}+1\}$. Każdemu krokowi przypiszemy parę $(q, f) \in Q \times (S \to (\Gamma \cup \{*\}))$, gdzie q jest stanem przed wykonaniem danego kroku, natomiast f zawartością stosu na pozycjach z S przed wykonaniem danego kroku, przy czym jeśli jakaś pozycja jest wtedy pusta, to wartością f jest $*$. Zauważmy, że różnych par jest tylko $|Q| \cdot (|\Gamma|+1)^{|S|}$, przy czym $|S| \leqslant C_1 + \delta_{max} + 1$. Natomiast kroków mamy więcej niż $C_3 = |Q| \cdot (|\Gamma|+1)^{C_1+\delta_{max}+1}$. Zatem dla pewnych dwóch kroków c i d mamy tą samą parę (q, f). Zauważmy, że stan stosu przed krokami c i d jest taki sam, gdyż kroki pośrednie nie modyfikowały stosu głębiej niż na pozycji $s_{min}+1$. Również stan przed c i d jest taki sam. Możemy więc usuniąć kroki $c, c+1, \ldots, d-1$ otrzymując poprawne, ale krótsze obliczenie. Przeczy to jednak wyborowi obliczenia jako najkrótsze.

Lemat 5 *Jeśli pomiędzy krokiem a i b (włącznie) nie wczytujemy nic z wejścia oraz dla każdego c takiego, że $a \leqslant c \leqslant b$, zachodzi $s(c) \geqslant min(s(a), s(b)) - \delta_{max}$, to $b - a + 1 \leqslant C_3(1 + |s(b) - s(a)|)$.*

Ustalmy a i b. Załóżmy, że $s(a) \leqslant s(b)$ (w przeciwnym przypadku sytuacja jest prawie symetryczna). Podzielimy przedział kroków $[a, b]$ na $k \leqslant s(b) - s(a) + 1$ sąsiednich przedziałów kroków: $[a = a_1, b_1 = a_2 - 1], [a_2, b_2 = a_3 - 1], \ldots [a_k, b_k = b]$, z których każdy będzie spełniał założenia lematu 4. Jeśli dla każdego c ($a \leqslant c \leqslant b$) zachodzi $s(c) - s(a) \leqslant C_1$, to cały przedział $[a, b]$ spełnia założenia lematu 4, bo poziom stosu waha się między $s(a) - \delta_{max}$

i $s(a) + C_1$. W przeciwnym przypadku robimy tak: Niech a_2 będzie pierwszym takim krokiem, że $s(a_2) - s(a) > C_1$. Dalej (dla $i \geqslant 3$) postępujemy indukcyjnie (tak długo jak się da): niech a_i będzie pierwszym takim krokiem $a_{i-1} < a_i \leqslant b$, że $s(a_i) > s(a_{i-1})$. Niech k będzie ostatnim takim i, że udało się wyznaczyć a_i. Ponadto niech $b_i = a_{i+1} - 1$ dla $i = 1, 2, \ldots, k-1$ oraz $b_k = b$. W ten sposób uzyskujemy jakiś podział na sąsiednie przedziały. Trzeba udowodnić, że spełniają one założenia lematu 4 oraz że $k \leqslant s(b) - s(a) + 1$.

Udowodnię najpierw, że dla każdych i, c, ($i > 1$, $a_i \leqslant c \leqslant b$) mamy $s(c) \geqslant s(a_i) - C_1$. Popatrzmy w tym celu na przedział $[max(s(c), s(a)) + 1, s(a_i)]$ pozycji stosu w kroku a_i. Przedział ten jest czysty, bo $p(max(s(c), s(a)) + 1, a_i) \geqslant a$ oraz $z(max(s(c), s(a)) + 1, a_i) \leqslant c$. Zatem, z lematu 8 otrzymujemy, że $s(a_i) - max(s(c), s(a)) \leqslant C_1$. Ale $s(a_i) - s(a) > C_1$, czyli $s(a_i) - s(c) \leqslant C_1$.

Oznacza to właśnie, że dla kroków w przedziale $[a_i, b_i]$ (dla $i > 1$) poziom stosu wynosi co najmniej $s(a_i) - C_1$. Wynosi on także nie więcej niż $s(a_i)$, bo pierwszy krok, w którym zużywamy więcej stosu, należy do następnego przedziału. Natomiast dla kroków z przedziału $[a_1, b_1]$ poziom stosu waha się między $s(a) - \delta_{max}$ i $s(a) + C_1$. Zatem każdy z przedziałów $[a_i, b_i]$ spełnia założenia lematu 4. Ponieważ $s(a_2) < s(a_3) < \ldots < s(a_k)$ to $k - 2 \leqslant s(a_k) - s(a_2)$. Ponadto ponieważ $a(a_k) \leqslant s(b) + C_1$ oraz $s(a_2) \geqslant s(a) + C_1 + 1$, to $k \leqslant 2 + (s(b) + C_1) - (s(a) + C_1 + 1) = s(b) - s(a) + 1$.

Korzystając z lematu 4 dla każdego z przedziałów otrzymujemy:

$$b - a + 1 = \sum_{i=1}^{k} (b_i - a_i + 1) \leqslant kC_3 \leqslant C_3(s(b) - s(a) + 1)$$

Lemat 6 $m_k - m_{k-1} \leqslant C_4(1 + |s(m_k) - s(m_{k-1})|)$

Dowód Ustalmy k. Najpierw udowodnimy, że $s(r_k) \leqslant max(s(m_k), s(m_{k-1})) + \delta_{max} + C_1$. Załóżmy przeciwnie, że $s(r_k) > max(s(m_k), s(m_{k-1})) + \delta_{max} + C_1$ i popatrzmy na przedział $[i, j] = [max(s(m_k), s(m_{k-1})) + \delta_{max} + 1, s(r_k)]$ pozycji stosu w chwili r_k. Przedział ten jest czysty, bo $p(j, r_k) < r_k < z(j, r_k)$ oraz $r_{k-1} \leqslant m_{k-1} < p(i, r_k)$, $z(i, r_k) < m_k \leqslant r_{k+1}$ (o ile r_{k-1} i r_{k+1} istnieją). Jednak długość tego przedziału jest większa niż C_1. Z lematu 8 dostajemy sprzeczność, czyli nasza nierówność jest prawdziwa. Ponieważ $s(r_k) \geqslant max(s(m_k), s(m_{k-1}))$, to wynika z niej bezpośrednio, że: $|s(m_k) - s(r_k)| + |s(r_k) - s(m_{k-1})| \leqslant |s(m_k) - s(m_{k-1})| + 2(\delta_{max} + C_1)$.

Zachodzą następujące nierówności (przejście między pierwszą i drugą linijką uzyskujemy korzystając z lematu 5 dla kroków $[m_{k-1} + 1, r_k - 1]$ oraz $[r_k + 1, m_k - 1]$ — jego założenia są spełnione na podstawie definicji m_{k-1} i m_k):

$$\begin{aligned}
m_k - m_{k-1} &= 2 + ((m_k - 1) - (r_k + 1) + 1) + ((r_k - 1) - (m_{k-1} + 1) + 1) \leqslant \\
&\leqslant 2 + C_3(2 + |s(m_k - 1) - s(r_k + 1)| + |s(r_k - 1) - s(m_{k-1} + 1)|) \leqslant \\
&\leqslant 2 + C_3(2 + 4\delta_{max} + |s(m_k) - s(r_k)| + |s(r_k) - s(m_{k-1})|) \leqslant \\
&\leqslant 2 + C_3(2 + 4\delta_{max} + |s(m_k) - s(m_{k-1})| + 2(\delta_{max} + C_1)) \leqslant \\
&\leqslant (2 + C_3(2 + 6\delta_{max} + C_1))(1 + |s(m_k) - s(m_{k-1})|) = C_4(1 + |s(m_k) - s(m_{k-1})|)
\end{aligned}$$

Podsumowanie Liczba kroków naszego obliczenia to $m_{|w|} - m_0 + 1 = 1 + \sum_{k=1}^{|w|}(m_k - m_{k-1})$. Korzystając z lematu 6 jest ona nie większa niż $1 + C_4 \sum_{k=1}^{|w|}(1 + |s(m_k) - s(m_{k-1})|)$. Następnie korzystając z lematu 3 dostajemy, że jest to nie więcej niż $1 + C_4(1 + C_2)|w|$. Dla $|w| \geqslant 1$ jest to $\leqslant (1 + C_4 + C_2C_4)|w| = C|w|$, co właśnie mieliśmy dowieść.

Zadanie 2

Klasa języków bezkontekstowych nie jest zamnkięta na wymienione operacje. Rozważmy najpierw język $L = \{a^n b^n c^m d^{2m} : n, m \geqslant 1\}$. Jest to niewątpliwie język bezkonstekstowy, generuje go na przykład gramatyka: $S \to XY$, $X \to aXb|ab$, $Y \to cYdd|cdd$. Udowodnię teraz, że język $\frac{1}{2}L$ nie jest bezkontekstowy. Załóżmy przeciwnie i skorzystajmy z lematu Ogdena. Niech M będzie stałą z tego lematu. Niech $\alpha = a^M b^M c^{2M}$. Należy ono do $\frac{1}{2}L$, bo $a^M b^M c^{2M} d^{4M} \in L$. Wyróżniamy wszystkie litery c, jest ich $\geqslant M$. Niech więc $\alpha = \alpha_1\gamma_1\beta\gamma_2\alpha_2$ będzie odpowiednim podziałem istniejącym na mocy lematu. Zauważmy, że jeśli któraś z części γ_1, γ_2 zawiera dwie różne litery, to słowo $\alpha_1\gamma_1^2\beta\gamma_2^2\alpha_2 \notin \frac{1}{2}L$, bo nie będzie postaci $a^n b^n c^m$. Ponadto, ponieważ $\gamma_1\gamma_2$ zawiera co najmniej jedną wyróżnioną pozycję, to γ_2 zawiera tylko litery c. Jeśli γ_1 zawiera jakąś literę a, to słowo $\alpha_1\gamma_1^2\beta\gamma_2^2\alpha_2$ zawiera więcej a niż b, więc nie należy do $\frac{1}{2}L$ (ponieważ zawiera też c). Podobnie jeśli γ_2 zawiera jakąś literę b. Zatem γ_1 i γ_2 zawierają tylko litery c. Ponieważ $\gamma_1\gamma_2$ jest niepuste, to $\alpha_1\gamma_1^2\beta\gamma_2^2\alpha_2 = a^M b^M c^{2M+k}$ dla pewnego $k \geqslant 1$. Jeśli $a^M b^M c^{2M+k} \in \frac{1}{2}L$, to $|a^M b^M c^{2M+k}| = |c^l d^{4M+2k+2l}|$ dla pewnego $l > 0$. To oznacza jednak, że $k = 2k + 3l$, co przeczy założeniu $k \geqslant 1$, sprzeczność.

Rozważmy teraz język $\{a^j b^j c^k a^l b^m c^n : j, k, l, m, n \geqslant 1, k \neq l\}$. Język ten jest bezkontekstowy, generuje go na przykład gramatyka: $S \to XYBC$, $X \to aXb|ab$, $Y \to cYa|cCa|cAa$, $C \to Cc|c$, $A \to Aa|a$, $B \to Bb|b$. Zauważmy, że jeśli $ww = a^j b^j c^k a^l b^m c^n \in L$, to $w = a^j b^j c^k$ oraz $j = l$. Wynika z tego, że $j \neq k$. W drugą stronę, jeśli $w = a^j b^j c^k$, gdzie $j, k \geqslant 1$, $j \neq k$, to $ww \in L$. Zatem $\sqrt{L} = \{a^j b^j c^k : j, k \geqslant 1, j \neq k\}$. Język ten nie jest bezkontekstowy. Załóżmy przeciwnie i skorzystajmy z lematu Ogdena. Niech M będzie stałą z tego lematu. Niech $\alpha = a^M b^M c^{M+M!} \in \sqrt{L}$. Wyróżniamy wszystkie litery a, jest ich $\geqslant M$. Niech więc $\alpha = \alpha_1\gamma_1\beta\gamma_2\alpha_2$ będzie odpowiednim podziałem istniejącym na mocy lematu. Zauważmy, że jeśli któraś z części γ_1, γ_2 zawiera dwie różne litery, to słowo $\alpha_1\gamma_1^2\beta\gamma_2^2\alpha_2 \notin \sqrt{L}$, bo nie będzie postaci $a^j b^j c^k$. Jeśli γ_2 nie zawiera żadnej litery b, to ponieważ $\gamma_1\gamma_2$ zawiera co najmniej jedną literę a, to słowo $\alpha_1\gamma_1^2\beta\gamma_2^2\alpha_2$ zawiera więcej a niż b, więc nie należy do $\sqrt{L}$. Podobnie dochodzimy do wniosku, że γ_1 musi zawierać dokładnie tyle samo liter a, co γ_2 liter b. Oznaczmy tą liczbę przez k i rozważmy $i = 1 + \frac{M!}{k}$ (ponieważ $1 \leqslant k \leqslant M$, to ma to sens). Wówczas $\alpha_1\gamma_1^i\beta\gamma_2^i\alpha_2 = a^{M+M!} b^{M+M!} c^{M+M!} \notin \sqrt{L}$, sprzeczność.

Zadanie 3

Idea dowodu Jeśli podzielimy sobie słowo w jakimś miejscu na dwie części, to zauważmy, że maszyna może przenieść tylko skończoną ilość informacji z lewej strony na prawą. Py-

taniem może być tylko jeden ze skończonej liczby stanów maszyny przy przechodzeniu do lewej części, odpowiedzią również. Na to samo pytanie uzyskiwana jest zawsze ta sama odpowiedź, bo maszyna nie może nic sobie zapisać w lewej części, bo tam już znajduje się słowo. Stworzymy więc automat, który od razu liczy odpowiedzi na wszystkie możliwe zapytania maszyny dotyczące lewej części.

W jedną stronę twierdzenie jest oczywiste. Każdy automat skończony możemy symulować za pomocą maszyny Turinga z ograniczeniem read once o tych samych stanach po prostu przesuwając się w prawo i nic nie zapisując. Weźmy więc dowolną maszynę Turinga M z ograniczeniem read once działającą nad alfabetem symboli terminalnych Σ. Załóżmy dla uproszczenia, że jej zbiór stanów to $Q^M = \{1, \ldots, n\}$. Załóżmy też, że maszyna M akceptuje zawsze z głowicą na prawo od początkowego słowa w. Każdą maszynę można łatwo zamienić na równoważną taką — gdy maszyna ma już akceptować, to jeszcze przesuwamy głowicę w prawo aż do napotkania blanka i dopiero wtedy naprawdę akceptujemy (przy tej przeróbce zwiększamy liczbę stanów o jeden).

Pośrednim krokiem będzie zdefiniowanie maszyn $\widetilde{M}_f$ dla wszystkich funkcji $f\colon Q^M \to P(Q^M)$ w następujący sposób:

- Alfabety jak w maszynie M rozszerzone o nowy symbol terminalny \$.
- Stany i stany końcowe jak w maszynie M.
- Przejścia takie jak w maszynie M oraz dodatkowo $(q, \$) \to (\$, p, +1)$ dla każdego $q \in Q^M$, $p \in f(q)$.

Dla dowolnej maszyny M' i dowolnego $Q_I \subset Q^M$ przez $D_2(M', Q_I)$ będę oznaczał maszynę M', która zaczyna obliczenia z głowicą nad drugą komórką taśmy w pewnym stanie $q_I \in Q_I$.

Ponadto dla $f\colon Q^M \to P(Q^M)$, $Q \subset Q^M$, $a \in \Sigma$ niech $next(f, Q, a)$ będzie zbiorem tych stanów $q_3 \in Q^M$, że jeśli na słowie $\$a$ uruchomimy maszynę $D_2(\widetilde{M}_f, Q)$ to istnieje obliczenie, w którym maszyna pierwszy raz znajdzie się na pozycji trzeciej (czyli za a) w stanie q_3.

Określę teraz deterministyczny automat skończony A, który ma być równoważny maszynie M:

- Jego zbiorem stanów będzie $P(Q^M) \times (Q^M \to P(Q^M))$.
- Stan początkowy to $(Q_I, (\lambda q.\phi))$, gdzie Q_I jest zbiorem stanów początkowych maszyny M.
- Przejścia: dla każdego stanu (Q, f) automatu A oraz $a \in \Sigma$ niech $((Q, f), a) \to (next(f, Q, a), \lambda q.next(f, \{q\}, a))$.
- Stany końcowe to takie stany (Q, f), że maszyna $D_2(\widetilde{M}_f, Q)$ akceptuje słowo \$.

Trzeba teraz udowodnić równoważność A i M. Wynika ona natychmiast z następującego lematu dla $k = |w|$ i z określenia stanów końcowych dla A.

Lemat 7 *Niech $w \in \Sigma^*$ będzie dowolnym słowem oraz niech $0 \leqslant k \leqslant |w|$. Niech (Q, f) będzie stanem automatu A po wczytaniu słowa $w_1 \dots w_k$. Wówczas maszyna M akceptuje słowo w wtedy i tylko wtedy, gdy maszyna $D_2(\widetilde{M}_f, Q)$ akceptuje słowo $\$w_{k+1} \dots w_{|w|}$.*

Dowód Będzie to dowód indukcyjny ze względu na k. Dla $k = 0$ teza jest oczywista: Wtedy automat A jest w stanie $(Q_I, \lambda q.\phi)$, gdzie Q_I jest zbiorem stanów początkowych maszyny M. Obliczeniu akceptującemu maszyny M na słowie w odpowiada więc obliczenie akceptujące maszyny $D_2(\widetilde{M}_f, Q_I)$ na słowie $\$w$. W drugą stronę odpowiedniość również zachodzi, gdyż obliczenie akceptujące $D_2(\widetilde{M}_f, Q_I)$ nie może wchodzić nad \$, bo nie ma żadnych przejść po odczytaniu \$.

Udowodnię teraz tezę dla dowolnego $k \geqslant 1$ zakładając, że jest ona prawdziwa dla $k-1$. Niech (Q_0, f_0) będzie stanem automatu A po wczytaniu $k-1$ liter słowa w, natomiast (Q_1, f_1) po wczytaniu k liter tego słowa. Po skorzystaniu z założenia indukcyjnego wystarczy udowodnić równoważność pomiędzy działaniem $D_2(\widetilde{M}_{f_0}, Q_0)$ na słowie $\$w_k \dots w_{|w|}$ oraz $D_2(\widetilde{M}_{f_1}, Q_1)$ na słowie $\$w_{k+1} \dots w_{|w|}$. Rozważmy najpierw obliczenie akceptujące maszyny $D_2(\widetilde{M}_{f_1}, Q_1)$ na słowie $\$w_{k+1} \dots w_{|w|}$. Pokażę odpowiadające mu obliczenie maszyny $D_2(\widetilde{M}_{f_0}, Q_0)$ na słowie $\$w_k \dots w_{|w|}$. Niech q_1 będzie pierwszym stanem obliczenia maszyny $D_2(\widetilde{M}_{f_1}, Q_1)$. Ponieważ $q_1 \in Q_1 = next(f_0, Q_0, w_k)$, to istnieje obliczenie $D_2(\widetilde{M}_{f_0}, Q_0)$ dla słowa $\$w_k$ (czyli także dla słowa $\$w_k \dots w_{|w|}$), które prowadzi do stanu q_1 na pozycji trzeciej. No to na początku wykonujemy to obliczenie. Później, dopóki maszyna jest nad literą w_{k+1} lub dalej, możemy wykonywać obliczenie maszyny $D_2(\widetilde{M}_{f_1}, Q_1)$. Zostają jeszcze pewne momenty, w których maszyna wchodzi nad symbol \$ w pewnym stanie q_0. Wtedy w następnym kroku już z niego schodzi na prawo w pewnym stanie q_1. Jeśli tak jest, to $q_1 \in f_1(q_0) = next(f_0, \{q_0\}, w_k)$, czyli istnieje obliczenie $D_2(\widetilde{M}_{f_0}, Q_0)$ dla słowa $\$w_k$ (czyli także dla słowa $\$w_k \dots w_{|w|}$), które prowadzi do stanu q_1 na pozycji trzeciej. Zastępujemy więc rozważany jeden krok maszyny $D_2(\widetilde{M}_{f_1}, Q_1)$ tym właśnie obliczeniem. Udowodniliśmy w ten sposób jedną stronę równoważności.

Rozważmy teraz obliczenie akceptujące maszyny $D_2(\widetilde{M}_{f_0}, Q_0)$ na słowie $\$w_k \dots w_{|w|}$. Pokażę odpowiadające mu obliczenie maszyny $D_2(\widetilde{M}_{f_1}, Q_1)$ na słowie $\$w_{k+1} \dots w_{|w|}$. Niech q_1 będzie stanem w którym obliczenie maszyny $D_2(\widetilde{M}_{f_0}, Q_0)$ wchodzi po raz pierwszy nad literę w_{k+1}. To samo obliczenie do tego momentu jest także poprawne dla słowa $\$w_k$. Oznacza to, że $q_1 \in next(f_0, Q_0, q_k) = Q_1$. Możemy więc zacząć odpowiadające obliczenie maszyny $D_2(\widetilde{M}_{f_1}, Q_1)$ po prostu od tego momentu, pomijając początkowy fragment. Później, dopóki maszyna jest nad literą w_{k+1} lub dalej, możemy wykonywać obliczenie maszyny $D_2(\widetilde{M}_{f_0}, Q_0)$. Zostają jeszcze pewne momenty, w których maszyna wchodzi nad symbol w_k w pewnym stanie q_0. Wtedy po pewnym czasie przechodzi po raz pierwszy na pozycję nad w_{k+1} w pewnym stanie q_1. Ten fragment obliczenia jest też obliczeniem maszymy $D_2(\widetilde{M}_{f_0}, \{q_0\})$ dla słowa $\$w_k$, w którym pierwszy raz znajdzie się na pozycji trzeciej w stanie q_1. Oznacza to, że $q_1 \in next(f_0, \{q_0\}, w_k) = f_1(q_0)$. Możemy więc cały ten fragment obliczenia zastąpić jednym krokiem: przejściem ze stanu q_0 przy odczycie \$ do stanu

q_1 na jedną pozyję w prawo. W ten sposób pokazaliśmy drugą stronę równoważności.

Jeśli pozwolimy maszynie zapisywać najwyżej raz, to uzyskamy pełnowartościową maszynę Turinga. Działanie dowolnej maszyny będziemy symulować na takiej w ten sposób, że po wykonaniu każdego kroku symulowanej maszyny będziemy kopiować słowo na którym operujemy na nowe miejsce. Dokładniej: Alfabet rozszerzamy w ten sposób, że dla każdego symbolu (terminalnego lub nieterminalnego, także blanka) dodajemy trzy nowe odpowiadające mu symbole nieterminalne. W ten sposób z każdym symbolem możemy też pamiętać informację, czy głowica symulowanej maszyny Turinga znajduje się na tej właśnie pozycji oraz czy symbol został już skopiowany. Dodatkowo dodajemy jeszcze symbol nieterminalny \$, który będzie służył do oddzielania kolejnych kopii taśmy symulowanej maszyny. Stan symulowanej maszyny będziemy pamiętać poprzez stany maszyny symulującej. Przygotowanie symulacji wygląda tak, że za słowem początkowym stawiamy \$ i kopiujemy tam słowo początkowe, przy czym nad pierwszym symbolem zaznaczamy, że tam stoi głowica. Takie kopiowanie wygląda tak, że w stanie zapamiętujemy symbol i zaznaczamy, że został już skopiowany, następnie jedziemy w prawo do końca i tam zapisujemy pamiętany symbol, po czym przesuwamy się w lewo aż do napotkania symbolu skopiowanego (przy czym pierwszy krok jest nieco inny, trzeba postawić \$ i zmodyfikować symbol). Następnie powtarzamy symulację pojedyńczego kroku maszyny. Taka symulacja wygląda tak, że za naszym słowem stawiamy \$ i kopiujemy tam to słowo, przy czym zamiast symbolu pod głowicą zapisujemy nowy postawiony tam symbol oraz znacznik głowicy stawiamy być może o jedną pozycję w prawo lub w lewo. Kopiowanie to wygląda podobnie jak poprzednio, przy czym jeśli na odczytanej pozycji stała głowica, to trzeba zapisać nowy symbol zgodnie z regułami symulowanej maszyny (na podstawie pamiętanego jej stanu). Również zawsze przy odczycie sprawdzamy jedną pozycję na lewo i jedną pozycję i jeśli tam stała głowica, to sprawdzamy czy zgodnie z regułami symulowanej maszyny przesunie się ona na pozycję bieżącą i jeśli tak, to to zaznaczamy. Są jeszcze szczególne przypadki. Gdy głowica jest na końcu taśmy i przesuwa się w prawo, wtedy trzeba dopisać dodatkowo nową pozycję. Trzeba też uważać co się stanie, gdy symylowana maszyna próbuje wykonać nieprawidłową operację wyjścia przed początek taśmy. Podczas tego kopiowania zapamiętujemy też jaki będzie nowy stan i po skopiowaniu do niego przechodzimy. Jeśli jest to stan akceptujący to akceptujemy. Jasne jest, że maszyny te są równoważne. Ponadto wszystkie operacje nadpisania to zastąpienie któregoś z symboli jego odpowiednikiem oznaczającym, że został on już skopiowany. A więc nadpisujemy najwyżej raz.

Zadanie 4

Idea dowodu Najpierw udowodnię, że obliczenie działające w czasie liniowym może w każdym miejscu przebywać tylko stałą ilość czasu. Nie może więcej, bo nie zdąży przynieść sobie informacji o długości słowa, czyli nie będzie wiedziało, ile czasu może w danym miejscu przebywać. Wynika z tego, że jeśli podzielimy sobie słowo w którymś miejscu na dwie części, to tylko skończoną liczbę informacji można przenieść między jedną częścią a drugą.

Pytaniem jest stan, w którym wchodzimy do lewej części, odpowiedzią stan, w którym wychodzimy, a możemy zadać tylko ograniczoną liczbę pytań. Stworzymy automat skończony, który od razu liczy wszystkie możliwe ciągi pytań i odpowiedzi.

W jedną stronę twierdzenie jest oczywiste. Każdy automat skończony można za pomocą maszyny turniga po prostu przesuwając się w prawo i zmieniając stan, wykonamy w ten sposób liczbę kroków równą długości słowa. Ustalmy pewną maszynę Turinga M pracującą w czasie liniowym z pewną stałą C. Załóżmy dla uproszczenia, że maszyna M zatrzymuje się zawsze z głowicą na prawo od początkowego słowa. Każdą maszynę można łatwo zamienić na równoważną taką — gdy maszyna ma już akceptować, to jeszcze przesuwamy głowicę w prawo aż do napotkania blanka i dopiero wtedy naprawdę się akceptujemy (przy tej przeróbce zwiększamy liczbę stanów o jeden), podczas tego przesuwania wykonamy nie więcej kroków niż wykonaliśmy do tej pory plus początkowa długość słowa. Żeby nie dopuścić do tego, że maszyna postawi blanka gdzieś w środku początkowego słowa, to dodajemy do alfabetu dodatkowy symbol odpowiadający blankowi, będziemy go zawsze stawiać zamiast blanka, a przy odczycie nie rozróżniamy jego i prawdziwego blanka.

Niech zbiorem stanów naszej maszyny będzie Q^M, a jej alfabetem symboli terminalnych Σ. Niech $C_1 = C^2+2C^2(|Q|+1)^C$. Rozmiarem obliczenia dla danej pozycji $r(p)$ niech będzie liczba kroków, w których głowica przechodzi pomiędzy pozycjami p i $p+1$ (w którąkolwiek stronę). Niech $k(p,i)$ będzie numerem kroku, w którym maszyna przechodzi pomiędzy pozycjami p i $p+1$ po raz i-ty. Pozycję p nazwiemy *niską*, jeśli $r(p) \leqslant C$. Udowodnię najpierw następujący lemat:

Lemat 8 *Dla każdego słowa w, pozycji na taśmie p_{max} ($1 \leqslant p_{max} \leqslant |w|$) oraz ustalonego obliczenia maszyny M na tym słowie, mamy $r(p_{max}) \leqslant C_1$.*

Załóżmy, że teza nie jest spełniona. Wybierzmy w, p_{max} oraz obliczenie, dla których $r(p_{max}) > C_1 = C^2+2C^2(|Q|+1)^C$, przy czym wyboru dokonujemy w ten sposób, aby słowo w było najkrótsze możliwe. Wiemy, że maszyna działa w czasie liniowym, a dokładniej, że wynokuje najwyżej $C|w|$ kroków. Zatem $C_1 < r(p_{max}) \leqslant C|w|$, czyli inaczej $|w| \geqslant C_1/C = C + 2C(|Q|+1)^C$.

Zauważmy teraz, że co najmniej $1+2(|Q|+1)^C \leqslant |w|/C$ pozycji p pomiędzy 1 i $|w|$ jest niskich, bo jeśli k jest liczbą takich pozycji, to mamy $C|w| \geqslant \sum_{p=1}^{|w|} r(p) \geqslant (|w|-k)(C+1)$, czyli $kC \geqslant |w|$.

Każdej pozycji niskiej p przypiszemy funkcję $f\colon \{1,2,\ldots,C\} \to (Q \cup \{*\})$ taką, że $f(a)$ dla $a \leqslant r(p)$ będzie stanem po kroku $k(p,a)$, natomiast dla $a > r(p)$ będzie $f(a) = *$. Różnych takich funkcji jest tylko $(|Q|+1)^C$. Natomiast pozycji niskich jest przynajmniej $1+2(|Q|+1)^C$. Zatem wśród pozycji od 1 do $p_{max}-1$ lub wśród pozycji od $p_{max}+1$ do $|w|$ jest ich co najmniej $1+(|Q|+1)^C$. Przyjmijmy, że wśród pozycji od 1 do $p_{max}-1$ (w przeciwnym przypadku jest analogicznie). Wówczas na tym fragmencie jest więcej pozycji niskich niż funkcji, czyli dla pewnych dwóch pozycji niskich $p_1 < p_2$ mamy tą samą funkcję f. Rozważmy wówczas słowo $v = w_1 \ldots w_{p_1} w_{p_2+1} \ldots w_{|w|}$. Nasze obliczenie na słowie w przenosi się na słowo v w naturalny sposób, tzn. wykonujemy kolejno kroki: $1, 2, \ldots, k(p_1,1), k(p_2,1)+$

$1, k(p_2,1)+1, \ldots, k(p_2,2), k(p_1,2)+1, k(p_1,2)+2, \ldots, k(p_1,3), k(p_2,3)+1, k(p_2,3)+2, \ldots$. Dzięki temu, że p_1 i p_2 mają taką samą funkcję f, to obliczenia wykonane nad każdą pozycją słowa v będą dokładnie takie same jak nad odpowiadającą jej pozycją słowa w (natomiast oczywiście globalnie może się zdarzyć, że jakieś kroki są w innej kolejności, np. że $k(p_2,2) > k(p_1,2)$). Zauważmy, że pozycja p_{max} nie została usunięta z naszego słowa i dla odpowiadającej jej pozycji nadal mamy taki sam duży rozmiar obliczenia. W ten sposób skonstruowaliśmy krótsze słowo v nie spełniające tezy, co przeczy wyborowi w jako najkrótszego takiego.

Dla konkretnej maszyny M', słowa wejściowego w, obliczenia i pozycji p funkcją przejścia nazwiemy funkcję $f\colon \{1,\ldots,C_1\} \to Q^M \cup \{*\}$ taką, że $f(a)$ dla $a \leqslant r(p)$ będzie stanem po kroku $k(p,a)$, natomiast dla $a > r(p)$ będzie $f(a) = *$.

Pośrednim krokiem będzie zdefiniowanie maszyn $\widetilde{M_S}$ dla wszystkich zbiorów $S \in P(\{1,\ldots,C_1\} \to Q^M \cup \{*\})$ w następujący sposób:

- Alfabety jak w maszynie M rozszerzone o nowe symbol terminalne \$, #.
- Stany: $Q^M \times S \times \{1,\ldots,C_1\}$.
- Przejścia:
  1. $((q_0,f,n),a_0) \to (a_1,(q_1,f,n),m)$ dla każdego $(q_0,a_0) \to (a_1,q_1,m)$ przejścia maszyny M oraz $f \in S$, $n \in \{1,\ldots,C_1\}$;
  2. $((f(n),f,n),\$) \to (\$,(f(n{+}1),f,n{+}2),+1)$ dla każdego $f \in S, n \in \{1,\ldots,C_1\}$, jeśli $f(n) \neq * \neq f(n+1)$;
  3. $((q_0,f,n),\#) \to (\#,(q_1,f,n),-1)$ dla każdego $f \in S$, $n \in \{1,\ldots,C_1\}$, $q_0,q_1 \in Q^M$.
- Stany akceptujące: $Q_F \times S \times \{1,\ldots,C_1\}$ gdzie Q_F jest zbiorem stanów akceptujących maszyny M.
- Stany początkowe: $(f(1),f,2)$ dla każdego $f \in S$, jeśli $f(1) \neq *$. Gdy nie ma żadnego stanu początkowego to zakładamy, że maszyna od razu zatrzymuje się.
- Maszyna zawsze zaczyna z głowicą na drugiej pozycji taśmy.

Ponadto dla $S \in P(\{1,\ldots,C_1\} \to Q^M \cup \{*\})$, $a \in \Sigma$ niech $next(S,a)$ będzie zbiorem funkcji przejścia dla pozycji drugiej maszyny $\widetilde{M_S}$, słowa \$a#, dla wszystkich możliwych obliczeń (rozważamy tu także obliczenia, które zatrzymują się w pewnym momencie mimo że możliwe są dalsze kroki).

Określę teraz deterministyczny automat skończony A, który ma być równoważny maszynie M:

- Jego zbiorem stanów będzie $P(\{1,\ldots,C_1\} \to (Q^M \cup \{*\}))$.

- Stan początkowy to $\{f\colon f(1) \in Q_I, f(n) = * \text{ dla } n > 1\}$, gdzie Q_I jest zbiorem stanów początkowych maszyny M.
- Przejścia: $(S, a) \to next(S, a)$ dla każdego stanu S automatu A oraz $a \in \Sigma$.
- Stany końcowe to takie stany S, że maszyna $\widetilde{M}_S$ akceptuje słowo \$.

Trzeba teraz udowodnić równoważność A i M. Wynika ona natychmiast z następującego lematu dla $k = |w|$ i z określenia stanów końcowych dla A.

Lemat 9 *Niech $w \in \Sigma^*$ będzie dowolnym słowem oraz niech $0 \leqslant k \leqslant |w|$. Niech S będzie stanem automatu A po wczytaniu słowa $w_1 \dots w_k$. Wówczas maszyna M akceptuje słowo w wtedy i tylko wtedy, gdy maszyna $\widetilde{M}_S$ akceptuje słowo $\$w_{k+1} \dots w_{|w|}$.*

Dowód Będzie to dowód indukcyjny ze względu na k. Dla $k = 0$ teza jest oczywista: Wtedy automat A jest w stanie $q_I^A = \{f\colon f(1) \in Q_I, f(n) = * \text{ dla } n > 1\}$, gdzie Q_I jest zbiorem stanów początkowych maszyny M. Obliczeniu akceptującemu maszyny M na słowie w odpowiada więc obliczenie akceptujące maszyny $\widetilde{M}_S$ na słowie $\$w$ za pomocą przejść typu 1 dla $f \in q_I^A$, $n = 2$. W drugą stronę odpowiedniość również zachodzi, przejściom typu 1 nie wchodzącym nad \$ odpowiadają przejścia maszyny M, natomiast przejścia innych typów nie wystąpią, ani nie wejdziemy nad \$, bo nie ma żadnych przejść po odczytaniu \$ (a rozważamy obliczenie akceptujące).

Udowodnię teraz tezę dla dowolnego $k \geqslant 1$ zakładając, że jest ona prawdziwa dla $k - 1$. Niech S_0 będzie stanem automatu A po wczytaniu $k - 1$ liter słowa w, natomiast S_1 po wczytaniu k liter tego słowa. Po skorzystaniu z założenia indukcyjnego wystarczy udowodnić równoważność pomiędzy działaniem $\widetilde{M}_{S_0}$ na słowie $\$w_k \dots w_{|w|}$ oraz $\widetilde{M}_{S_1}$ na słowie $\$w_{k+1} \dots w_{|w|}$. Rozważmy najpierw obliczenie akceptujące O_1 maszyny $\widetilde{M}_{S_1}$ na słowie $\$w_{k+1} \dots w_{|w|}$. Pokażę odpowiadające mu obliczenie maszyny $\widetilde{M}_{S_0}$ na słowie $\$w_k \dots w_{|w|}$. Niech $(f_1(1), f_1, 2)$ będzie pierwszym stanem obliczenia O_1. Zauważmy, że w żadnym z przejść maszyny $\widetilde{M}_{S_1}$ nie zmienia się drugi element (czyli f_1). Z określenia przejść typu 2 wynika, że f_1 jest funkcją przejścia dla pozycji 1. Wiemy też, że $f \in next(S_0, w_k)$. Istnieje więc obliczenie O_0 maszyny $\widetilde{M}_{S_0}$ na słowie $\$w_k\#$ takie, że f_1 jest funkcją przejścia dla pozycji 2. Robimy więc tak: Wykonujemy obliczenie O_0 aż do momentu, gdy wejdzie ono nad trzecią pozycję. Wtedy zaczynamy wykonywać O_1, z tym że nie uwzględniamy drugiego i trzeciego elementu stanu (czyli f i n), są one stale takie jak w ostanim dotychczas kroku O_0. Tak postępujemy aż do momentu gdy O_1 wejdzie nad \$, wówczas znowu wykonujemy O_0, itd. Będzie to poprawne obliczenie $\widetilde{M}_{S_0}$ na słowie $\$w_k \dots w_{|w|}$. Kroki nad \$ i w_k wzięte z O_0 są oczywiście poprawne. Kroki nad w_{k+1} i dalej wzięte z O_1 są poprawne, mimo że zmieniliśmy f i n, bo przejścia typu 1 są takie same niezależnie od f i n. Natomiast wejścia nad w_{k+1} wyglądają tak samo w O_1 jak w nowym obliczeniu. Również wejścia nad w_k wyglądają tak samo w O_0 jak w nowym obliczeniu. Udowodniliśmy w ten sposób jedną stronę równoważności.

Rozważmy teraz obliczenie akceptujące O_0 maszyny $\widetilde{M}_{S_0}$ na słowie $\$w_k \dots w_{|w|}$. Pokażę odpowiadające mu obliczenie maszyny $\widetilde{M}_{S_1}$ na słowie $\$w_{k+1} \dots w_{|w|}$. Niech f_1 będzie funkcją

przejścia dla pozycji 2 (czyli między w_k i w_{k+1}) tego obliczenia. Rozważmy następujące obliczenie $\widetilde{M}_{S_0}$ na słowie $\$w_k\#$: Wykonujemy wszystkie kroki obliczenia O_0 wykonywane nad pozycją 1 lub 2. Gdy obliczenie to wychodzi nad pozycję 3, to pomijamy wszystkie kroki aż do powrotu na pozycję 2, a krok wracający na pozycję 2 zastępujemy jednym krokiem wracającym na pozycję 2 w takim samym stanie. Taki krok jest możliwy, bo jak jesteśmy nad # to możemy przejść w lewo w dowolnym stanie. Zauważmy, że funkcją przejścia dla pozycji 2 tak skonstruowanego obliczenia nadal jest f_1, co oznacza, że $f_1 \in next(S_0, w_k) = S_1$. Pokażę teraz obliczenie maszyny $\widetilde{M}_{S_1}$ na słowie $\$w_{k+1} \dots w_{|w|}$. Zaczynamy w stanie $(f_1(1), f_1, 2)$ nad w_{k+1}. Obliczenie O_0 również kiedyś wejdzie nad w_{k+1}, po raz pierwszy zrobi to w stanie $(f_1(1), \cdot, \cdot)$. Będziemy od tego momentu wykonywać obliczenie O_0, z tym że nie uwzględniamy drugiego i trzeciego elementu stanu, będą one stale takie jak w pierwszym kroku (czyli f_1 i 2). Tak postępujemy aż do momentu, gdy O_0 wejdzie nad w_k (czyli nasze obliczenie nad \$). Wtedy pomijamy wszystkie kroki O_0 wykonane nad \$ i w_k, wykonujemy dopiero krok wracający nad w_{k+1}. Taki krok jest możliwy, weszliśmy nad \$ w stanie $f_1(2)$, a chcemy znad niego wrócić w stanie $f_1(3)$. Później znowu wykonujemy kroki O_0 aż do powrotu nad \$, itd. Wszystkie „skopiowane" kroki są poprawne, bo przejścia typu 1 są takie same niezależnie od f i n. Natomiast przejścia znad \$ w prawo dlatego, że funkcją przejścia obliczenia O_0 nad pozycją 2 jest właśnie f_1. W ten sposób pokazaliśmy drugą stronę równoważności.

W dowodzie tym korzystaliśmy cały czas niejawnie z tego, że wszystkie rozważane obliczenia mają dla każdej pozycji rozmiar co najwyżej C_1. Dzięki temu funkcja przejścia opisuje wszystkie przejścia między rozważanymi pozycjami, a nie tylko pierwsze C_1. Dla obliczenia maszyny M wynika to z lematu 8. Trzeba też zwrócić uwagę, że równoważne obliczenia kolejnych maszyn $\widetilde{M}_S$ mają na każdej pozycji taki sam rozmiar obliczenia jak na odpowiadającej pozycji obliczenie maszyny M, czyli również nie większy niż C_1. Również niejawnie korzystamy z tego, że wszystkie te obliczenia skończą się na prawo od rozważanych pozycji (tak założyliśmy o obliczeniach akceptujących).

Zadanie 6

Idea dowodu Udowodnię, że jeśli maszyna zużywa mniej niż $c \cdot \log\log|w|$ komórek pamięci, to tak naprawdę zużywa stałą ich liczbę. Nie może więcej, bo za pomocą tak małej ilości pamięci nie jest w stanie skoncentrować informacji o długości słowa, nie będzie więc wiedziała ile komórek może zużyć. Skończoną liczbę komórek możemy symulować za pomocą stanu maszyny, więc sytuacja wygląda tak jak w zadaniu 3.

Przyjmuję, żę rozważana w tym zadaniu maszyna ma dwie taśmy. Na jednej (górnej) zapisane jest słowo, które należy rozpoznać. Na tej taśmie nie można pisać. Druga (dolna) taśma początkowo jest pusta, na tej taśmie można pisać. Przyjmuję taki sposób liczenia komórek, że liczę wszystkie komórki taśmy do zapisywania, nad którymi znalazła się maszyna, również te, na których zapisała blanka. Alternatywnym sposobem liczenia byłby taki, w którym liczymy tylko komórki, w których maszyna zapisała symbol różny od

blanka. Wtedy maszyna mogłaby sobie wybierać, w których komórkach będzie zapisywać zostawiając jakieś przerwy między nimi. Mój dowód nie przenosi się na ten sposób liczenia, bo wtedy można przechowywać dużo więcej informacji na taśmie (poprzez rozmieszczenie wykorzystanych komórek oraz poprzez aktualną pozycję). Jest jeszcze drobna (sięgająca najwyżej jeden) rozbieżność między liczbą komórek, nad którymi znalazła się maszyna, a liczbą komórek, do których coś zapisała, bo maszyna może wejść nad jakąś pozycję i zakończyć działanie nic tam jeszcze nie zapisując. Jednak widać, że takie różnice rzędu jeden nie mają zupełnie wpływu na rozważania asymptotyczne jak w tym zadaniu.

Twierdzenie udowodnię metodą nie wprost. Ustalmy pewną maszynę M, dla której teza nie jest prawdziwa. Inaczej mówiąc, dla każdej stałej $c > 0$ tylko dla skończenie wielu słów w maszyna M wykorzystuje co najmniej $c \cdot \log\log|w|$ komórek pamięci. Załóżmy dla uproszczenia, że maszyna M akceptuje zawsze z głowicą na prawo od początkowego słowa w. Każdą maszynę można łatwo zamienić na równoważną taką — gdy maszyna ma już akceptować, to jeszcze przesuwamy głowicę w prawo aż do napotkania blanka i dopiero wtedy naprawdę akceptujemy (przy tej przeróbce zwiększamy liczbę stanów o jeden).

Niech q będzie liczbą stanów naszej maszyny. Niech s będzie liczbą symboli terminalych oraz nieterminalnych w alfabecie naszej maszyny. Załóżmy, że $\log\log s > 0$ (jeśli tak nie jest, to możemy dodać jakieś nieużywane symbole nieterminalne). Rozważmy

$$c = \frac{1}{\log(s+2)}\left(1 - \frac{\log\log(s+1)}{\log\log(2s+2)}\right)$$

Zauważmy, że $c > 0$. Wybierzmy k_0 takie, że:

- Każde ze skończenie wielu słów w, które zużywa co najmniej $c \cdot \log\log|w|$ komórek pamięci zużywa ich mniej niż k_0.
- $k_0 > c\log\log(2s+2)$
- Dla każdego $k \geqslant k_0$ zachodzi $2kqs^k + 2 \leqslant (s+1)^k$. (Istnienie takiej stałej wynika z tego, że funkcja wykładnicza $\left(\frac{s+1}{s}\right)^k$ rośnie szybciej niż funkcja $2kq + \frac{2}{s^k}$.)
- Dla każdego $k \geqslant k_0$ zachodzi $k(s+1)^k + 1 \leqslant (s+2)^k$.

Udowodnię, żę każde obliczenie akceptujące zużywa mniej niż k_0 komórek pamięci. Załóżmy przeciwnie. Rozważmy słowo w, dla którego maszyna zużywa $k \geqslant k_0$ komórek pamięci, przy czym słowo w niech będzie najkrótsze możliwe. Z definicji k_0 wiemy, że $k < c\log\log|w|$. Niech n_{max} będzie (dowolnym) krokiem, w którym dolna głowica znajdowała się na pozycji k, natomiast p_{max} pozycją górnej taśmy przed tym krokiem. Niech $T = \{1,\ldots,k\} \times (\Sigma \cup \Gamma)^k \times Q$, gdzie Q jest zbiorem stanów maszyny M, natomist Σ i Γ jej alfabetami symboli terminalnych i nieterminalnych. Elementy T będą opisywać *T-stan* maszyny przy ustalonej pozycji na górnej taśmie (trzebaby to nazwać „stanem", ale to słowo jest już zajęte). Pierwszy element tej trójki będzie oznaczał pozycję głowicy na dolnej taśmie, drugi będzie to zawartość dolnej taśmy, natomiast trzeci — stan maszyny.

Każdej pozycji p na górnej taśmie przypiszemy parę $(t_0, f) \in T \times (T \to T \cup \{*\})$ w następujący sposób: Niech t_0 będzie T-stanem maszyny po pierwszym wejściu górnej głowicy na pozycję $p+1$. Dla każdego T-stanu t, rozważmy moment (o ile istnieje, jest najwyżej jeden), w którym maszyna przeszła z pozycji $p+1$ na pozycję p mając po tym przejściu T-stan t. Niech $f(t)$ będzie T-stanem w pierwszym momencie po rozważanym, gdy maszyna znalazła się na pozycji $p+1$ (istnieje, gdyż maszyna kończy na prawo od słowa w). Dla pozostałych t niech $f(t) = *$. Zauważmy, że różnych takich par jest

$$N = ksq \cdot (ks^k q + 1)^{ks^k q}$$

Mamy dalej (korzystając z własności k_0 i ponieważ $k \geqslant k_0$):

$$\begin{aligned} 2N &< 2(ks^k q+1)^{ks^k q+1} \leqslant (2ks^k q+2)^{ks^k q+1} \leqslant (s+1)^{k(s+1)^k} \\ &\leqslant (s+1)^{(s+2)^k} \leqslant (s+1)^{(s+2)^{c\log\log|w|}} \end{aligned}$$

Ponieważ $c\log\log(2s+2) \leqslant k_0 \leqslant k \leqslant c\log\log|w|$, to:

$$c = \frac{1}{\log(s+2)}\left(1 - \frac{\log\log(s+1)}{\log\log(2s+2)}\right) \leqslant \frac{1}{\log(s+2)}\left(1 - \frac{\log\log(s+1)}{\log\log|w|}\right)$$

Mnożąc przez $\log(s+2)\log\log|w|$ dostajemy:

$$c\log(s+2)\log\log|w| + \log\log(s+1) \leqslant \log\log|w|$$

Potęgując (tzn. podnosząc 2 do potęgi strona nierówności) otrzymujemy:

$$(s+2)^{c\log\log|w|}\log(s+1) \leqslant \log|w|$$

I jeszcze raz potęgujemy:

$$(s+1)^{(s+2)^{c\log\log|w|}} \leqslant |w|$$

Dostaliśmy więc, że $2N < |w|$. Oznacza to, że pewnym dwóm pozycjom p_0, p_1 wśród $1, \ldots, p_{max}-1$ lub wśród $p_{max}, \ldots, |w|$ przypisaliśmy tą samą parę (t_0, f). Niech $v = w_1 \ldots w_{p_0} w_{p_1+1} \ldots w_{|w|}$ (ze słowa w usuwamy litery na pozycjach od p_0+1 do p_1).

Pokażę obliczenie akceptujące słowo v, które również zużywa k komórek pamięci. Będzie to sprzeczne z założeniem, że w było najkrótsze. Na początku wykonujemy takie same kroki jak w obliczeniu na słowie w, aż do momentu wejścia nad pozycję p_0+1. Rozważmy teraz pierwszy moment, w którym obliczenie to wchodzi na pozycję p_1+1. Wykonujemy to obliczenie od tego momentu na odpowiadających pozycjach słowa v, aż do momentu gdy wejdzie ono spowrotem na pozycję p_1 (albo się skończy). Jest to poprawny ciąg kroków, bo na górnej taśmie odczytujemy odpowiadające sobie pozycje, natomiast w momencie wejścia na p_0+1 i w momencie wejścia na p_1+1 jest taki sam T-stan t_0. Następnie powtarzamy następującą czynność:

Krok l obliczenia na w polega na tym, żę przechodzimy z pozycji p_1+1 na p_1 w pewnym T-stanie t. Rozważmy moment, w którym obliczenie to znalazło się na pozycji p_0 w T-stanie

t (może się zdarzyć, że jest to wcześniej lub później niż l). Wykonyjemy to obliczenie od tego momentu na odpowiadających pozycjach słowa v, aż do momentu gdy wejdzie ono na pozycję p_0+1. Jest to oczywiście poprawny ciąg kroków, bo na górnej taśmie odczytujemy odpowiadające sobie pozycje, a zaczynamy w takim samym T-stanie t. Po wejściu na pozycję p_0+1 jesteśmy w T-stanie $f(t)$. Niech r będzie pierwszym krokiem po kroku l, w którym oryginalne obliczenie wchodzi na pozycję p_1+1. Po tym kroku również będzymy w T-stanie $f(t)$. Dalej wykonujemy kroki od kroku $r+1$ aż do momentu, gdy wejścia ono spowrotem nad p_1 albo się skończy. Jest to poprawne, bo z górnej taśmy odczytujemy to samo i jesteśmy w tym samym T-stanie. Jeśli obliczenie się nie skończyło, to wykonujemy dalej czynność opisaną w tym akapicie.

Trzeba uzasadnić, że każdy z kroków obliczenia na słowie w, przed którym głowica znajdowała się na nieusuniętej pozycji, zostanie wykonany dokładnie raz. Jasno widać, że będzie tak z obliczeniami na prawo od p_1, one wykonują się w tej samej kolejności co poprzednio. W międzyczasie wykonują się jakieś obliczenia na pozycjach do p_0. Obliczenie przed pierwszym wejściem na p_0+1 wykona się dokładnie raz, na samym początku. Każdy z pozostałych fragmentów wykonywał się jako odpowiedź na wejście z pozycji p_0+1 na pozycję p_0 w pewnym T-stanie. Teraz natomiast wykona się jako odpowiedź na wejście z pozycji p_1+1 na pozycję p_1 w tym samym T-stanie, czyli również dokładnie raz.

W szczególności więc wykona się krok n_{max}, gdyż pozycja p_{max} nie została usunięta. Skonstruowane obliczenie na słowie v wykorzystuje więc także k komórek pamięci. Przeczy to wyborowi słowa w, co kończy dowód.

Wiemy więc, że każde obliczenie akceptujące zużywa mniej niż k_0 komórek pamięci. Możemy więc maszynę M zasymulować za pomocą maszyny, która w ogóle nic nie zapisuje. Będzie ona pamiętała pierwsze k_0-1 komórek dolnej taśmy oraz pozycję głowicy na niej w stanie, jest tego skończenie wiele. Zauważmy, że jeśli maszyna M przechodzi nad komórkę k_0, to już na pewno nie zaakceptuje słowa, możemy więc nie symulować dalej, tylko zatrzymać się w stanie nieakceptującym. Maszyna która nic nie zapisuje akceptuje język regularny, zgodnie z tym co zostało udowodnione w zadaniu 3.