Rozwiązania zadań ze zbioru zadań edycji 2020

z Języków Formalnych i Złożoności Obliczeniowej

Bartłomiej Grochowski

Część pierwsza:

Automaty, języki regularne i bezkontekstowe

zadania 1-79

Zamieszczone tu rozwiązania mają służyć
złapaniu pewnych idei i mogą być pomocą przy
rozpoczęciu przygody z JFiZO. Spisywanie rozwiązań,
niestety, nie ułatwi zdania egzaminu.

Uniwersytet Wrocławski
18 czerwca 2023

JFIZO zad 1

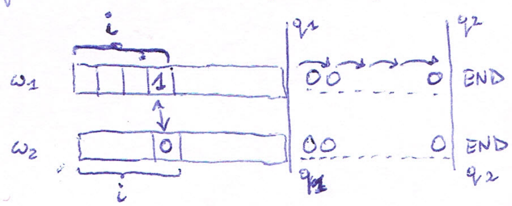
Dowód NIE WPROST → zał., że istnieje automat, który ma < 1024 stanów

Rozważamy 10 literowe słowa (sufiksy) nad alfabetem {0,1}.

Różnych słów jest $2^{10} = 1024$. Jeśli stanów jest mniej niż 1024:

* z zasady szufladkowej istnieją 2 różne słowa w_1, w_2 t. że po ich przeczytaniu DFA jest w tym samym stanie

* jeśli są różne, to na i-tym znaku się różnią.



* dopisujemy odpowiednią ilość takich samych liter. (w_1', w_2')

* czyli po przejściu tego ciągu automat będzie w tym samym stanie q_2 dla słowa w_1' i w_2' BO JEST DETERMINISTYCZNY.

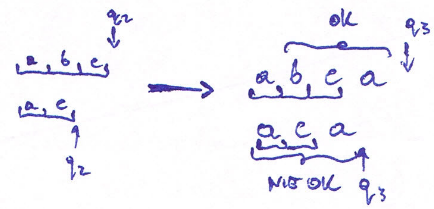
* czyli umiemy skonstruować słowo t. że automat nie będzie rozstrzygał czy słowo należy do języka.

↯

Zad 2 16

L (dobre końcówki) = { ε, a, b, c, ab, ac, bc, ca, cb, ba, perm „abc" }

→ zał., że ma mniej, więc istn. dwa dobre słowa różne i taki sam stan:



Wtedy możemy coś dopisać mając w' i v' t. że ten sam stan, ale $w' \in L$ a $v' \notin L$

↯

Zad 3

$L = \{a^n b^{2n} \mid n \in \mathbb{N}\}$ nie jest regularny:

Zał. że jest → k-stała z lematu.

Weźmy słowo $w = a^k b^{2k}$, $w \in L$

1) $|w| \geq k$
2) $|xy| \leq k$ → dowolny podział xyz jest taki, że

   y to same literki 'a'

   $xyyz$ nie należy do L, bo zwiększyliśmy ilość literek 'a', a nie zwiększyliśmy 'b'.

↯ ■

4

Słowo tylko jednoliterowy alfabet, to utożsamiamy słowa z mnożeniem ich długości.

1) L jest pusty, to L^* też, czyli regularny.

2) L zawiera tylko ε, wtedy L^* też, czyli regularny.

3) L zawiera jakieś słowa, weźmy najkrótsze, niech ma długość n.

Rozważmy języki $L_i = \{ v \in L^* : |v| = k \wedge k \bmod n = i \}$

Oczywiście $L^* = L_0 \cup L_1 \cup \ldots \cup L_{n-1}$

Zauważmy, że jeśli $\underbrace{00\ldots0}_{7} \in L^*$ to $\underbrace{00\ldots0}_{7}\underbrace{0\ldots0}_{n} \in L^*$

Pokażmy, że $\forall_i$ L_i jest regularny, tzn. zbudujemy DFA.

1. L_i jest pusty lub zawiera tylko ε – wtedy jest regularny.
2. L_i ma jakieś słowa, wtedy ma najkrótsze słowo $m(i)$

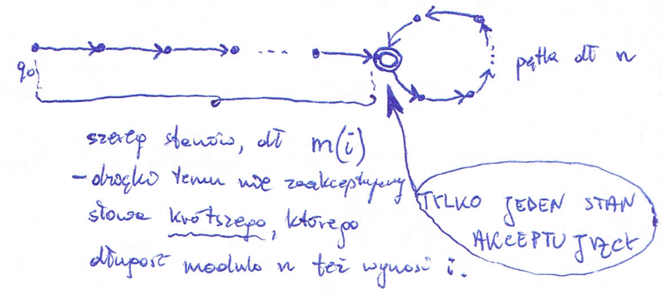
Do L^* będą należały wszystkie słowa długości $m(i)+n$

$m(i) = \underbrace{\quad}_{k}\underbrace{\quad}_{k}\underbrace{\quad}_{k}\underbrace{\ }_{i} \in L^*$

$m(i)+k = \underbrace{\quad}_{k}\underbrace{\quad}_{k}\underbrace{\quad}_{k}\underbrace{\quad}_{k}\underbrace{\ }_{i} \in L^*$ bo $m(i)$ a potem doklejamy k

ALE $x = \underbrace{\quad}_{k}\underbrace{\ }_{i} \notin L$ bo $m(i)$ jest najkrótsze $\equiv_n i$.

Czyli dla niepustego języka L_i zbudujemy automat:



Zatem $\forall_i$ L_i jest regularny.

Suma języków regularnych jest językiem regularnym.

Zatem L^* jest regularny ∎

JFIZO zad 5.

Pokażemy, że ten język nie jest regularny.

Zał. że jest → zachodzi lemat o pompowaniu

→ istnieje stała k z lematu o pompowaniu

$w = \underbrace{x}\underbrace{y}\underbrace{z}$ $\forall m \in \mathbb{N}$ $w' = \underbrace{x}\underbrace{y^m}\underbrace{z} \in L$

CEL: znaleźć takie m, żeby $w' \notin L$.

Niech $p = VAL(w) = VAL(z) + 2^{|z|} \cdot VAL(y) + 2^{|y|+|z|} \cdot VAL(x)$

p - pierwsze, $p > 2^k$ czyli dla w' już rozpatrujemy, że jest w języku i zachodzi lemat o pompowaniu.

Weźmy $m = p$ → pokażemy, że $w' \notin L$.

$$VAL(w') = VAL(z) + 2^{|z|} VAL(y) + 2^{|z|+|y|} VAL(y) + \ldots + 2^{|z|+(p-1)|y|} VAL(y) + 2^{|z|+p|y|} \ldots$$

$$= VAL(z) + VAL(y) \cdot 2^{|z|} \left(1 + 2^{|y|} + \ldots + 2^{(p-1)|y|}\right) + VAL(x) \cdot 2^{p|y|+|z|}$$

Pokażemy, że $VAL(w')$ nie jest pierwsze (że $p \mid VAL(w')$)

$$VAL(w') \equiv_p VAL(z) + VAL(y) \cdot 2^{|z|} \left(\frac{1 \cdot \overbrace{(1 - 2^{p|y|})}^{\equiv_p 1 - 2^{|y|}}}{1 - 2^{|y|}} \right) + VAL(x) \cdot 2^{p|y|+|z|} \equiv_p 2^{|y|+|z|}$$

$$\equiv_p VAL(z) + VAL(y) \cdot 2^{|z|} + VAL(x) \cdot 2^{|y|+|z|} \equiv_p p \equiv_p 0$$

Czyli $p \mid VAL(w')$ czyli w' nie reprezentuje liczby pierwszej $\Rightarrow w' \notin L$

Małe tw. Fermata: $a^p \equiv_p a$

zad 6

1) $L = \{ ww^R x : w, x \neq \varepsilon \}$ nie jest regularny.

Załóżmy, że L jest regularny, wtedy zachodzi tw. o indeksie.

Rozpatrzmy ciąg słów $w_i = (01)^i$

Okazuje się, że $\forall i, j \in \mathbb{N}$ $[w_i]_{\sim L} \neq [w_j]_{\sim L}$ (LEMAT) zatem mamy nieskończenie wiele klas abstrakcji $\sim_L$, więc L nie jest regularny.

→ DOWÓD LEMATU: $\forall i, j$ $i \neq j$ $w_i \not\sim_L w_j$

Weźmy dowolne i, j, $i < j$ (bez straty ogólności).

Skonstruujmy: Załóżmy nie wprost, że $w_i \sim_L w_j$.

$$w_i' = \underbrace{0101\ldots01}_{w_i}\overbrace{\underbrace{10\ldots10}_{w_i^R}\underbrace{0}_{x}}^{y}$$

$$w_j' = \underbrace{0101\ldots\ldots0101}_{w_j}\overbrace{\underbrace{10\ldots10}_{w_i^R}\underbrace{0}_{x}}^{y}$$

Widzimy, że $w_i y \in L$, ale $w_j y \notin L$ (przez te „dwie jedynki")

Czyli $w_i \not\sim_L w_j$ (z definicji)

sprzeczność, zatem udowodniliśmy.

2) $L = \{ xwx : x \neq \varepsilon \}$ nie jest regularny

Załóżmy, że jest reg, wtedy zachodzi lemat o pompowaniu i niech k będzie stałą z lematu. Wtedy $\forall w$ $|w| > k$ ...

Weźmy $w = 1\underbrace{00\ldots0}_{k-1}1\underbrace{00\ldots0}_{k-1}$, $|w| > k$ oczywiście.

Pozostaje istotne pytanie: jakie może być?

1) $y = 1$ → $xy^2z \notin L$ → $\underline{11}00\ldots0100\ldots0$ (te dwie jedynki psują)

2) $y = 10^i$ → $xy^2z \notin L$ → $1\underbrace{0\ldots0}_{\text{mało zer}}10\ldots0\,1\underbrace{00\ldots0}_{k-1 \text{ zer}}$

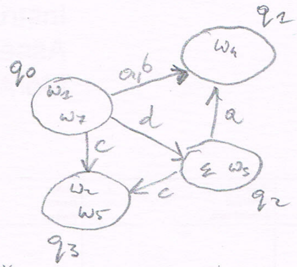
3) $y = 0^i$ → $xy^0z \notin L$ → $1\underbrace{00\ldots0}_{\text{mało}}1\underbrace{00\ldots0}_{k-1 \text{ zer}}$

↯ zatem L nie jest regularny.

zad. 1

TW. $[\sim_L]$ skończone wtw L regularny

($\Rightarrow$)



$DFA = \langle \Sigma, Q, q_0, F, \delta \rangle$

?

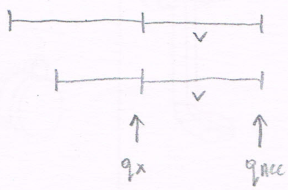
Σ z języka

klasy abstrakcji

klasa dla q_0

mamy

$\forall v \in \Sigma^*$ $wv \in L \Leftrightarrow w'v \in L$



wszystkie stany od q_x do q_{acc} są akceptujące

Czyli albo w i $w' \in L$ albo w i $w' \notin L$.

Czyli sprawdzamy wszystkie stany, czy reprezentanci tych klas $\in L$ i jeśli tak, to oznaczamy te stany jako ACC.

($\Leftarrow$)

L reg $\Rightarrow$ istnieje automat A.

Najmniejszy automat rozpoznający L ma tyle stanów ile klas abstrakcji relacji $\sim_L$.

$\to$ nie wprost że $<$, wtedy $\exists w, w'$ trafią do jednego stanu a będą w dwóch różnych klasach abstrakcji.

Dopisujemy cokolwiek $\to$ dalej będą w tym samym stanie.

Czyli $\forall v \in \Sigma^*$ $wv \in L \Leftrightarrow w'v \in L$ $\to$ czyli w i w' są w tej samej klasie abstrakcji. $\lightning$

DFA ma skończenie wiele stanów, czyli liczba klas abstrakcji $\sim_L$ jest skończona. ∎

Zad 8

a) $i_L \le i_L^{inf}$

$w \not\sim_L v \Rightarrow w \not\sim_L^{inf} v$

jeśli tu w różnych klasach abstrakcji — to tym bardziej tu w różnych (więc jest ich więcej)

Dowód: Kontrapozycja.

$w \sim_L^{inf} v \Rightarrow w \sim_L v$

$w \sim_L^{inf} v \overset{def}{\equiv} \forall_{x,y}\ xwy \in L \Leftrightarrow xvy \in L$, w szczególności dla $x = \varepsilon$

mamy $\forall_y\ wy \in L \Leftrightarrow vy \in L \overset{def}{\equiv} w \sim_L v$ ■

b) $i_L^{inf} \le (i_L)^{i_L}$

i_L skończone, z tw o indeksie wiemy że istnieje DFA rozpoznający L i że minimalny ma $n = i_L$ stanów.

Czego jest n^n? → Możliwych wyników różnych stanów funkcji:

$$F(w) \overset{def}{\equiv} \langle \hat\delta(q_0, w), \hat\delta(q_1, w), \dots, \hat\delta(q_{n-1}, w) \rangle$$

„Na co przejdzie który stan”. , F zależy od automatu.

$w \sim_F v \Leftrightarrow F(w) = F(v)$ // ZACHOWUJE STAN

Pokażemy:

(1) # klas abstrakcji $\sim_F \le n^n$

(2) $i_L^{inf} \le \# \sim_F$

(1) Dowód nie wprost. Załóżmy, że klas abstrakcji jest $> n^n$.

Wypiszmy je wszystkie:

(1) * $F(w) = \langle p_0, \dots, p_{n-1} \rangle$

⋮

(n^n+1) * $F(w) = \langle p_0', \dots, p_{n-1}' \rangle$

⋮

Możliwych wyników jest maksymalnie n^n (to zależy od automatu). Ale skoro klas jest więcej, to istnieją dwie klasy abstrakcji o takiej samej konfiguracji stanów.

Wtedy jakieś słowo „w” może należeć do tych obu klas abstrakcji. ↯, bo może tylko do jednej. ■

(2) $w \not\sim_L^{inf} v \Rightarrow w \not\sim_F v$

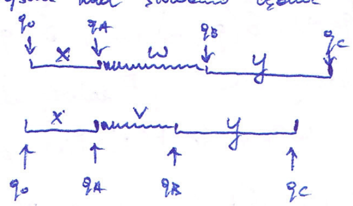
podobnie jak w a) – tych będzie więcej lub tyle samo.

Dowód: Kontrapozycja.

$w \sim_F v \Rightarrow w \sim_L^{inf}$

$w \sim_F v \overset{def_F}{\equiv} F(w) = F(v) \overset{def_F}{\equiv} \hat\delta(q_0, w), \dots, \hat\delta(q_{n-1}, w) = \hat\delta(q_0, v) \dots \hat\delta(q_{n-1}, v)$

Czyli po przejściu nad słowami będzie ten sam stan.



zatem:

$\forall_{x,y}\ \hat\delta(q_0, xwy) = \hat\delta(q_0, xvy) \equiv \forall_{x,y}\ xvy \in L \Leftrightarrow xwy \in L$

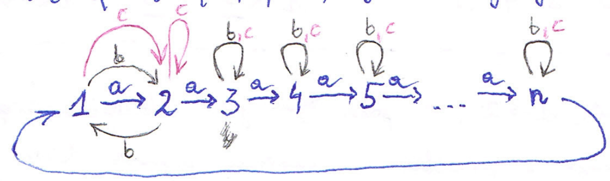
$\overset{def}{\equiv} w \sim_L^{inf} v$ ■

Zad 9 $\forall Q \; \exists \text{DFA} \; A = \langle \Sigma = \{a,b,c\}, Q, q_0, \delta, F \rangle$

takie że $\forall f: Q \to Q \; \exists w \; \forall q \in Q \; \hat{\delta}(q,w) = f(q)$

istnieje słowo, które „realizuje" tę funkcję

Dają nam $Q = \{1, 2, 3, \dots, n\}$. Zbudujemy automat.



* „a" przesuwa stany, $f_a: \begin{matrix} 1 & 2 & 3 & \dots & n \\ 2 & 3 & 4 & & 1 \end{matrix}$

* „b" robi switch stanów 1 i 2, czyli transpozycję.

Zauważmy, że za pomocą literek a i b możemy zrobić transpozycję dowolnych stanów przenumerowując je

Np.: $f_{3\leftrightarrow4}$: $12\underline{34}5 \xrightarrow{aaa} 45\underline{12}3 \xrightarrow{b} 45\underline{21}3 \xrightarrow{aa} 12\underline{43}5$

Czyli $f_{3\leftrightarrow4} = f_{aaabaa}$, dla $n=5$

Skoro permutacje to złożenie transpozycji, to KAŻDĄ permutację możemy modelować przy użyciu samych literek „a" i „b".

Czyli jasło $f: \begin{matrix} 1&2&3&4&5 \\ 2&1&4&5&3 \end{matrix}$ to umiemy ✓.

A co z funkcjami, które nie są permutacjami?

Np.: $f: \begin{matrix} 1&2&3&4&5 \\ 2&1&4&1&1 \end{matrix}$. Do tego jest literka „c".

Mała uwaga: Umiemy robić transpozycję ~~tr~~ tr[i,i+1]. A tr[i,i+2]?

Tak. Przykład:

$1\;2\;\underline{3}\;4\;\underline{5}$ $\; \downarrow ?$ $\; 1\;2\;5\;4\;3$

$1\;2\;\underline{3}\;\underline{4}\;5$

$1\;2\;\underline{4}\;3\;\underline{5}$

$1\;2\;5\;\underline{3}\;\underline{4}$

$1\;2\;\underline{5}\;4\;\underline{3}$ OK ✓

* „c" zamienia $1 \to 2$, a poza tym jest identycznością.

Poprzez składanie z permutacjami, możemy wyrazić takie g_{ij} że $g(i)=j$ a poza tym to identyczność.

Np g_{58}: $\dots, g(3)=3, g(4)=4, g(5)=8, g(6)=6, \dots$

Zauważmy, że każdą funkcję f możemy wyrazić za pomocą złożenia permutacji f' i kilku funkcji g_{ij}. Przykład.

$f: \begin{matrix} 1&2&3&4&5 \\ 1&1&3&2&1 \end{matrix}$ $\qquad f': \begin{matrix} 1&2&3&4&5 \\ 1&4&3&2&5 \end{matrix}$

konflikty, dlaczego to nie permutacja $\qquad$ poprawione

Czyli musimy zmienić $4 \to 1$ i $5 \to 1$, dzięki g_{41} i g_{51}.

$f = f' \circ g_{41} \circ g_{51}$

$12345 \xrightarrow{f'} 14325 \xrightarrow{g_{41}} 11325 \xrightarrow{g_{51}} 11321$ $\quad f$

Zatem istnieje $w \in \{a,b,c\}^*$ które modeluje funkcję f, tzn $\forall q \in Q \; \delta(q,w) = f(q)$ ∎

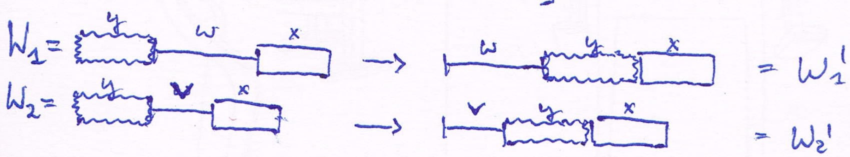
Zad 10 $|\Sigma| = 1 \Rightarrow i_L^{inf} = i_L$ (Liczba klas abstrakcji $\sim_L$ jest równa liczbie klas abstrakcji $\sim_L^{inf}$)

Niech $\Sigma = \{a\}$

Pokażemy, że $\forall w,v \; w \sim_L^{inf} v \Leftrightarrow w \sim_L v$, wtedy będzie tyle samo klas abstrakcji $\sim_L$ i $\sim_L^{inf}$ ($[w]_{\sim_L} = [w]_{\sim_L^{inf}}$)

Weźmy dowolne $w, v \in a^*$, bez straty ogólności niech $|w| \geq |v|$.
w, v to te same litery, różnią się TYLKO długością.

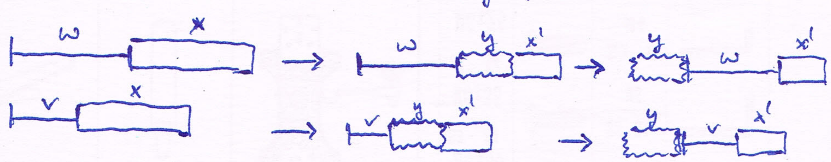
1) ($\Rightarrow$) $[\forall x,y \; ywx \in L \Leftrightarrow yvx \in L] \Rightarrow [\forall x \; wx \in L \Leftrightarrow vx \in L]$



$W_1 = W_1'$ i $W_2 = W_2'$ bo to te same literki i te same długości.

$y + x \to$ nowe x, czyli $\forall x,y \ldots \Rightarrow \forall x \ldots$ ✓

2) ($\Leftarrow$) $[\forall x \; wx \in L \Leftrightarrow vx \in L] \Rightarrow [\forall x,y \; ywx \in L \Leftrightarrow yvx \in L]$



$\forall x$ ___ $\Rightarrow \forall x,y$ ___

✓

11.

$L_n \subseteq \{0,1\}^*$ bez straty ogólności ($a \leftrightarrow 0$, $b \leftrightarrow 1$)

$L_n = \{ ww \mid w \in \{0,1\}^n \}$

np $01{,}01 \in L_2$
$01{,}00 \notin L_2$

Zał. że $\exists$ DFA że $|Q| < 2^n$ i rozpoznaje L_n.

$|\{0,1\}^n| = 2^n$, z zasady szufladkowej

mamy $w, w' \in \{0,1\}^n$ $w \neq w'$ $\hat{\delta}(w, q_0) = \hat{\delta}(w', q_0)$

$\hat{\delta}(ww, q_0) = \hat{\delta}(w'w, q_0)$

$\lightning$ bo $ww \in L_n$ a $w'w \notin L_n$

INFO:

$\{0,1\}^*$ jest podjęzykiem $\{0,1,\#\}^*$ (jeszcze jedna literka)

możemy sobie odrzucić literkę i pokazać, że dla podjęzyka nie działa.

$\longrightarrow L_n$ $\{0,1,\#\}$ automat w szczególności powinien w szczególności akceptować słowa złożone z samych literek $\{0,1\}$

Zad 12

$$L_i = \{ w^k \mid \exists k \in N \wedge |w| = i \}$$

→ minimalna liczba stanów: 2^i

argument jak w poprzednim zadaniu

→ stała z lematu o pompowaniu: i, $p(n) = n$

Weźmy dowolne słowo $w \in L_i$ t. że $|w| > i$.

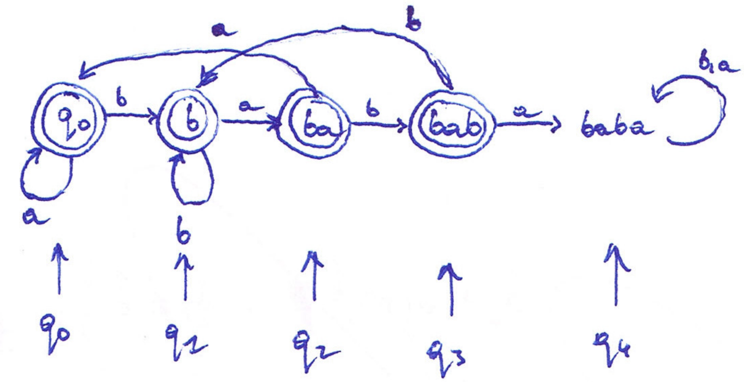
Jest ono postaci $w = vv\ldots v$, $|v| = i$

Pokażmy podział na xyz: $\underbrace{\varepsilon}_{x}\underbrace{\boxed{v}}_{y}\underbrace{\boxed{v\ldots v}}_{z}$

Wtedy $\forall n \in N$ $xy^n z \in L_i$ bo jest postaci $v\ldots v$.

Zad 13

nie zawiera 'baba'



1) $q_1 \to q_1$ nie przez q_0 $= (b + abb)^* = w_1$

2) $q_0 \to q_0 = (a + b w_1 aa)^* = w_2$

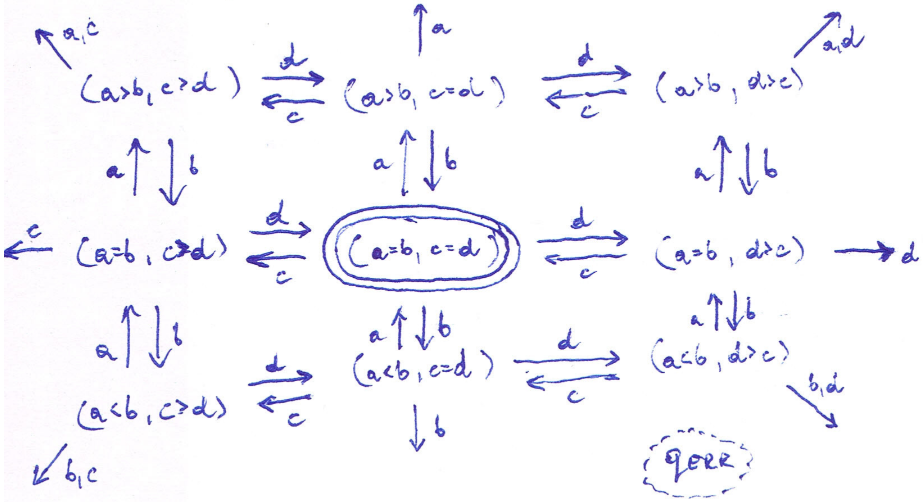
3) $q_1 \to F \setminus \{q_0\}$ nie przez q_0 $= w_1(\underset{q_1}{\varepsilon} + \underset{q_2}{a} + \underset{q_3}{ab}) = w_3$

4) $q_0 \to F = w_2(\varepsilon + b w_3) = w_4$

REGEX $= (a + b(b+abb)^*aa)^*(\varepsilon + b(b+abb)^*(\varepsilon+a+ab))$

Zad 14 cały czas podczas czytania $|a-b| \le 1 \wedge |c-d| \le 1$ else q_{ERR}

$q_0 = q_F = (a=b, c=d)$



REGEX – trudne eliminacje (The Cat)

JFIZO zad 15

$w_1 = (a_0 a_1)^2$

$w_k = (w_{k-1} a_k)^2 \qquad L_k = \{w_k\}$

Język zawiera tylko jedno słowo, więc nie możemy używać + i *.

Najkrótszy regex L_k to w_k.

$$|w_k| = |w_{k-1} a_k w_{k-1} a_k| = 2|w_{k-1}| + 2 = 2(2|w_{k-2}| + 2) + 2 = \ldots \simeq \underline{2^n}$$

Dokładamy „∩". Nowy regex:

$\varphi_1 = w_1$

$\varphi_k = (\varphi_{k-1} a_k)^* \cap \left((a_0 + a_1 + \ldots + a_{k-1})^* a_k (a_0 + a_1 + \ldots + a_{k-1})^* a_k\right)$

$L_{\varphi_k} = \{w_k\} \rightarrow$ regex φ_k opisuje słowo w_k

Dowód: indukcja po k.

\+ $k = 1 \qquad \varphi_1 = w_1$ z def.

* zał. φ_{k-1} opisuje w_{k-1}

$(\varphi_{k-1} a_k)^*$ opisuje $(w_{k-1} a_k)^i$ dla pewnego $i \in \mathbb{N}$.

Trzeba uściślić żeby to „i" było równe 2.

To nam zapewni przecięcie z $(a_0 + a_1 + \ldots + a_{k-1})^* a_k (a_0 + a_1 + \ldots + a_{k-1})^* a_k$

bo φ_{k-1} ma tylko literki $a_0 \ldots a_{k-1}$.

To definiuje słowo w_k.

$$|\varphi_k| = (\varphi_{k-1} a_k)^* \cap \left((a_0 + a_1 + \ldots + a_{k-1})^* a_k (a_0 + \ldots + a_{k-1})^* a_k\right) =$$

$$= |\varphi_{k-1}| + 2k + c = |\varphi_{k-2}| + 2k-1 + 2k + c = \ldots \simeq \underline{n^2}$$

Zad 16

a) $\exists \varphi$ t.że $L_{a\varphi} = L_{\varphi b}$?

NIE. Zał. że istnieje. φ musi się zaczynać od a^* wtedy ~~[illegible]~~ wszystkie słowa z $L_{\varphi b}$ mają x literek 'a', a wszystkie z $L_{a\varphi}$ mają $x+1$ czyli $L_{\varphi b} \neq L_{a\varphi}$

b) $\exists \varphi$ t.że $L_{a^*\varphi} = L_{\varphi b^*}$

Tak. $\varphi = a^* b^*$

$$a^*\varphi = a^*a^*b^* = a^*a^*b^*b^* = a^*b^*b^* = \varphi b^*$$

JFIZO zad 17

A.

1) $0^*10^*+0^*$

→ deterministyczne TAK → jeśli „w" ma „1" to wiemy skąd pochodzi potem „0" na lewo i na prawo wiemy skąd pochodzą

→ online NIE

$0^*10^*+0^*$

$\underbrace{0}_{w}\underbrace{0}_{w_1}$

$\underbrace{0}_{w}\underbrace{1}_{w_2}$ nie zgadzają się na prefiksie

2) $(0+1)^*1(0+1)^*$

→ deterministyczne TAK → bierzemy dwie ostatnie litery słowa

$1\,x$

$(0+1)^*\,1\,(0+1)$

→ online NIE →

$\underbrace{0}_{w}\underbrace{11}_{w_1}$ $\quad$ $\underbrace{01}_{w}\underbrace{11}_{w_2}$

$(0+1)^*1(0+1)$ $\quad$ $(0+1)^*1(0+1)$

różnią się na prefiksie w

3) $(0+1)(0+2)^* + (1+2)(0+1)^* + (0+2)(1+2)^*$

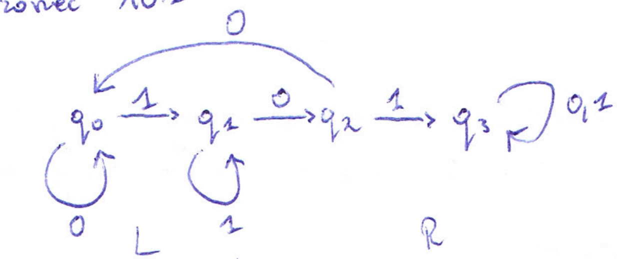
$w = \underbrace{0}$

NIE deterministyczne

→ więcej niż jedno mapowanie

w szczególności nie online

B. Wzorzec 101



$q_3 \in F$

L $\qquad$ R

regex: $(0+11^*00)^*\,11^*\underbrace{01}_{101}\,(0+1)^*$

q_0 $\quad L_A \quad L_B$ $\quad q_0$ $\quad q_3$

pierwsze wystąpienie 101

Jest ono deterministyczne.

$\underbrace{\sim\sim\sim}\ \underbrace{\sim}\ \underbrace{101}\ \sim\sim\sim$

$0 \notin 11^*00$
wtedy $\in$ do L_A
$0 \in 11^*00$
to $0 \in L_B$

być może ciąg „1"

0 i 1 przypisywany do R

JFIZO zad 19

$\varphi = 0^* 1 0^* (\varepsilon + 10^*)$

L_φ to język słów, które mają jedną lub dwie jedynki.

1° Jest deterministyczny regularny:

$$\underbrace{0000}_{0^*}\underbrace{1}_{1}\underbrace{0000}_{0^*}\underbrace{1}_{1}\underbrace{0000}_{0^*}$$

2° Jest online.

Rozpatrzmy wszystkie możliwe prefiksy:

1. $\underbrace{0}$
2. $\underbrace{0}1$
3. $\underbrace{0}1\underbrace{0}$
4. $\underbrace{0}1\underbrace{0}1$
5. $\underbrace{0}1\underbrace{0}1\underbrace{0}$

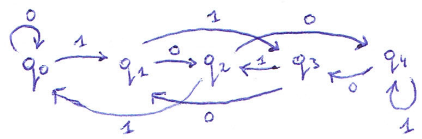
$\longrightarrow$ ZAWSZE JEDNOZNACZNIE ZMAPOWANE ✓

ARYTMETYKA $\mathbb{Z}_5$ to ciało, bo 5 jest pierwsza.

JFIZO zad 20

a) czytamy bity $\longrightarrow$, jak algorytm Hornera, jesteśmy w $\mathbb{Z}_5$
mnożymy $*2$ i dodajemy

stany: q_i - reszta z dzielenia dotychczas przeczytanej liczby przez 5.



Automat:

$Q = \{ q_i$, przeczytana liczba mod $5 \equiv i \}$

$\Sigma = \{0, 1\}$

$q_0 = q_0$

$F = q_0$

$\delta = \{ < q_i, j, q_{(2i+j) \bmod 5} > \}$

0 lub 1

(to też DFA, bo z każdego stanu wychodzi dokładnie jedna strzałka dla osobnej literki)

b) czytamy bity $\longleftarrow$

$2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$2^0 \equiv 1 \bmod 5$
$2^1 \equiv 2 \bmod 5$
$2^2 \equiv 4 \bmod 5$
$2^3 \equiv 3 \bmod 5$
$2^4 \equiv 1 \bmod 5$

cykl

$f: \mathbb{Z}_4 \to \{1,2,3,4\}$

f: $f(0) = 1$
$f(1) = 2$
$f(2) = 4$
$f(3) = 3$

Idea: dodajemy do akumulatora tyle ile trzeba nad ciałem $\mathbb{Z}_5$.
trzeba zapamiętywać na którym indeksie od prawej jesteśmy.

stany: $Q = \{ < i, j > : i \in \mathbb{Z}_5, j \in \mathbb{Z}_4 \}$ (20 stanów)

akumulator mod 5 — indeks od prawej mod 4

Automat:

$Q = \{ <i,j> : i \in \mathbb{Z}_5, j \in \mathbb{Z}_4 \}$

$\Sigma = \{0, 1\}$

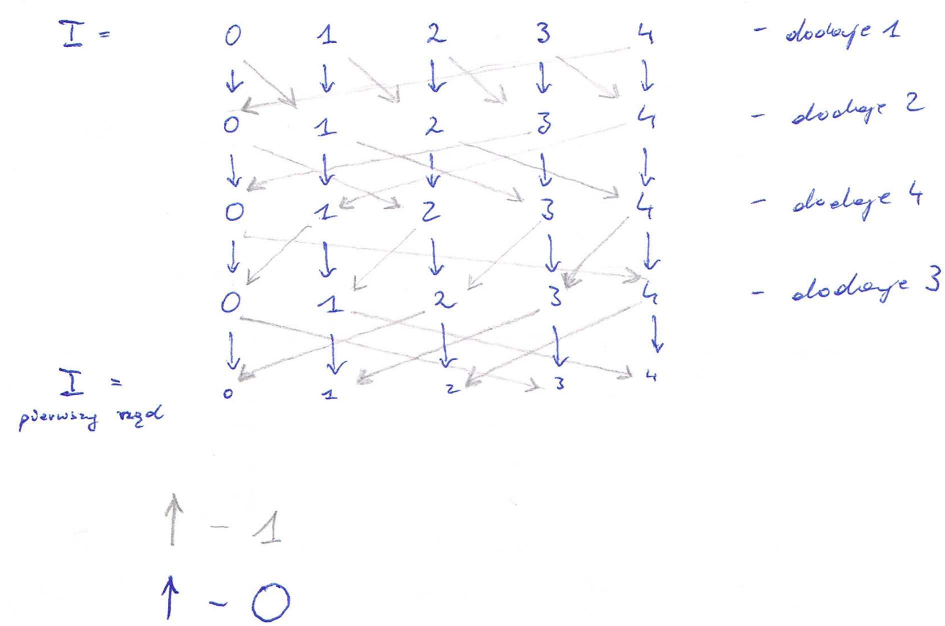
$q_0 = <0,0>$

$F = \{ <0, j> : j \in \mathbb{Z}_4 \}$ - nieważne, jak daleko jest ta pozycja

$\delta = \{ <i,j>, 1, < \overset{\text{mod } 5}{i + f(j)}, \overset{\text{mod } 4}{j+1} > \} \cup \{ <i,j>, 0, <i, j+1> \}$

trzeba dodać — odległość się przesuwa o 1 — nie dodajemy

Rysunek:



JFIZO zad 21

Pokażemy, że tak jest podając konstrukcję takiego NDFA

$A = \langle Q, \Sigma, q_0, F, \delta \rangle$

→ odwracamy wszystkie łuki w diagramie.

→ stan początkowy staje się jedynym stanem końcowym

→ tworzymy nowy stan początkowy q_0' z ε-przejściami do wszystkich stanów końcowych A.

$$A' = \begin{cases} Q' = Q \cup q_0' \\ F' = q_0 \\ \delta' = \{\langle q_i, a, q_j \rangle : \underbrace{\delta(q_j, a, q_i)}_{\text{odwrócenie kierunku}}\} \cup \{\langle q_0', \varepsilon, q \rangle : \underbrace{q \in F}_{\text{stany końcowe } A}\} \end{cases}$$

Lemat: odwrócenie strzałek rozszerza się na dłuższe słowa, czyli:

$$\hat{\delta}(q_i, w, q_j) \Leftrightarrow \hat{\delta}'(q_j, w^R, q_i)$$

Pokażemy, że automat A' rozpoznaje język L^R.

$$w \in L(A) \Leftrightarrow w^R \in L(A')$$

Dowód:

($\Rightarrow$) Weźmy dowolne w t.że $w \in L(A)$. Zachodzi:

$\hat{\delta}(q_0, w, q_k)$ gdzie $q_k \in F$. Z lematu mamy:

$\hat{\delta}'(q_k, w^R, q_0)$ i $\delta'(q_0', \varepsilon, q_k)$.

Czyli $\hat{\delta}'(q_0', w^R, q_0)$, czyli w^R jest akceptowane przez A'

Czyli $w^R \in L(A')$ ✓

($\Leftarrow$) Weźmy dowolne $w \in L(A')$. $\hat{\delta}'(q_0', w, \overset{\text{tylko jeden stan akceptujący.}}{q_0})$

Mamy ε-przejście ze stanu q_0' czyli $\delta'(q_0', \varepsilon, q_k)$ i $\hat{\delta}'(q_k, w, q_0)$ dla pewnego $q_k \in F$.

Z lematu mamy $\hat{\delta}(q_0, w^R, q_k)$, $q_k \in F$, czyli w^R jest akceptowane przez A. Czyli $w^R \in L(A)$ ✓.

→ Czyli automat A' jest ok.

(Dowód lematu)

Indukcja po długości słowa.

1° $|w| = 1$, wtedy $w = a$

$$\hat{\delta}(q_i, w, q_j) \Leftrightarrow \hat{\delta}(q_i, a, q_j) \overset{\text{z def automatu } A'}{\Leftrightarrow} \delta'(q_j, a, q_i) \Leftrightarrow \hat{\delta}'(q_j, w^R, q_i) \checkmark$$

2° zał: $\forall w'.\ |w'| < |w|$ tak jest.

$w = aw'$, wtedy:

$$\hat{\delta}(q_i, w, q_j) \Leftrightarrow \hat{\delta}(q_i, aw', q_j) \overset{\text{def } \delta}{\Leftrightarrow} \exists q_k.\ \delta(q_i, a, q_k) \wedge \hat{\delta}(q_k, w', q_j)$$

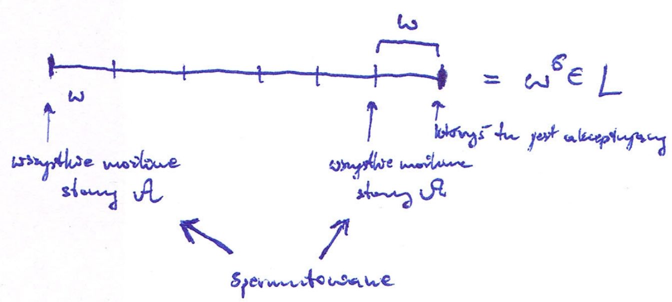
$$\overset{\text{zał ind}}{\Longleftrightarrow} \exists q_k\ \hat{\delta}'(q_j, w'^R, q_k) \wedge \delta'(q_k, a, q_i) \overset{\text{def}}{\Leftrightarrow} \hat{\delta}'(q_j, w'^R a, q_i) \Leftrightarrow \hat{\delta}'(q_j, w^R, q_i)$$

Zad 22 $\{w \mid \exists n \ w^n \in L\} = L'$ regularny.

L reg $\Rightarrow$ istnieje DFA, ma skończenie wiele stanów. $q_1, \dots, q_n$

Niech nowy automat A' ma stany postaci $\langle q_1, \dots, q_n \rangle$

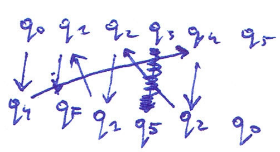
$$\delta'(\langle q_1, \dots, q_n \rangle) = \langle \delta(q_1), \dots, \delta(q_n) \rangle$$



$A': \xrightarrow{q_0} \langle q_1, q_2, \dots, q_n \rangle$

$\downarrow w$

$\langle q_1', q_2', \dots, q_n' \rangle \to q_F$ wtw istnieje cykl:



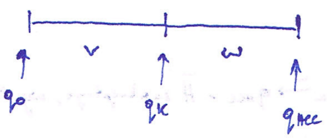
$$q_0 \to q_4 \to q_2 \to q_1 \to q_F \equiv q_0 \xrightarrow{w^4} q_F$$

Skoro stanów jest skończenie wiele, to możemy dla A' wypisać na początku wszystkie stany akceptujące. ✓

Zad 23 $\quad L/2 = \{ w \mid \exists v.\ vw \in L \wedge |v| = |w| \}$

→ regularny.

L reg zatem istnieje automat A. Skonstruujemy $\overline{A}$ rozp. $L/2$.



$vw \in L$

$\overline{Q} = Q^n \times 2^Q$

Q^n ← co się dzieje ze stanami A

2^Q ← zbiór możliwych stanów początkowych ‚q_k' dla czytania ‚w'

$q_0, q_1, q_2, q_3, q_4, q_5, q_6, \{q_0\}$

$\big\downarrow w$

$q_7, q_{130}, \underline{q_F}, q_5, q_6, \underline{q_F}, q_3, \{\underbrace{\ldots\ldots}_{}\}$ (S nad nawiasem)

↳ zb. wszystkich możliwych stanów po przeczytaniu jakiegoś słowa $\Sigma^{|w|}$

q_2 i q_5 są dobrymi stanami początkowymi do czytania ‚w'

$\overline{F} = \{ \langle r_0, r_1, r_2, \ldots, r_{n-1}, S \rangle \mid \exists k.\ r_k \in F \wedge q_k \in S \}$

$\overline{\delta}(\langle r_0, r_1, \ldots, r_{n-1}, S \rangle, a) = \langle \delta(r_0, a), \delta(r_1, a), \ldots, \delta(r_{n-1}, a), \{$ stany takie, że robią się $\delta(q, b)$ $q \in S$, $b \in \Sigma \} \rangle$

$q \in S$ ← stany, które były wcześniej

$b \in \Sigma$ ← wszystkie możliwe literki

$\overline{A}$ akceptuje $w \Leftrightarrow w \in L/2$

$(\Rightarrow)$

$\overline{A}$ akceptuje, czyli istnieje q_k takie, że da się do niego dojść z q_0 literkami z Σ długości $|w|$, tj. $q_0 \to q_k \to q_{acc}$, czyli $\exists v$ t.że $vw \in L$ ✓

$(\Leftarrow)$

$w \in L/2$ czyli $\exists v$ t.że $q_0 \xrightarrow{v} q_k \xrightarrow{w} q_{acc}$ - $\overline{A}$ zaakceptuje, wynika z konstrukcji. ✓

Zad 24

NFA $\mathcal{A}_1$, NFA $\mathcal{A}_2$ $L(\mathcal{A}_1) = L(\mathcal{A}_2)$?

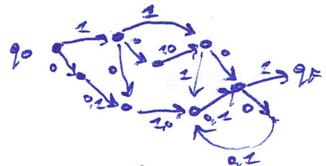
$L_1 = L(\mathcal{A}_1)$ – regularny

$L_2 = L(\mathcal{A}_2)$ – regularny

$$L' = \underbrace{\underbrace{(L_1 \cup L_2)}_{\text{reg}} \cap \underbrace{(\overline{L_1} \cup \overline{L_2})}_{\text{reg}}}_{\text{reg}}$$

$L' = \emptyset \Leftrightarrow L_1 = L_2$

Skonstruujmy automat $\mathcal{A}'$ rozpoznający L' (deterministyczny)



Zapuść BFS od q_0 w grafie skierowanym. Jeśli da się dojść do q_F to $L_1 \neq L_2$, jeśli się nie da to $L' = \emptyset$ czyli $L_1 = L_2$.

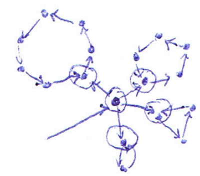
JFIZO zad 25.

Pokażemy taki język L, że da się go rozpoznać NDFA $|Q| < 20$ i nie da się $\overline{L}$ o $|\overline{Q}| < 200$.

Język: $L = \{0^n \mid 210 \nmid n\}$

$\overline{L} = \{0^n \mid 210 \mid n\}$

NDFA dla L:



Jeśli $210 \nmid n$ to $7 \nmid n$
lub to $5 \nmid n$
lub to $3 \nmid n$
lub to $2 \nmid n$

18 stanów

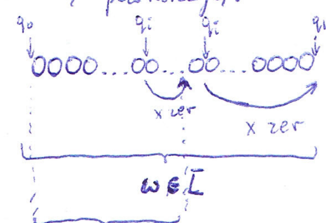
stany ⊙ – nieakceptujące

stany • – akceptujące

Zał. że da się NDFA rozpoznający $\overline{L}$ t. że $|\overline{Q}| < 200$.

Weźmy słowo $w = \underbrace{000\ldots00}_{210}$. $w \in \overline{L}$ NDFA czyta po kolei literki.

Nie wiemy do jakiego stanu przejdzie po przeczytaniu '0' ale stanów jest < 200 a literek 210 to na ścieżce (każdej) od q_0 do q_k istnieją stany q_i, że się powtarzają.



$v \notin \overline{L}$ a zostanie zaakceptowane przez NDFA

Czyli da się od tego pierwszego q_i dojść w x krokach do q_k - stanu akceptującego.

Czyli NDFA zaakceptuje słowo v, które jest krótsze niż 210 znaków.

Czyli $n / 210$ ↯

31 M_n → dowolne słowo, którym brakuje 0 lub 1 liter alfabetu.

NDFA $|Q| \geq 2^{\frac{n}{2}-1}$

$w = \dots a_i \dots a_j$ $a_j > a_i$ (rosnące)

litery się nie powtarzają

$X \subseteq \overline{M_n}$, podzbiór dopełnienia

X - rosnące i nie rozdzielają rodzin

$|X| = 2^{\frac{n}{2}} - 1$

na tyle sposobów możemy wybrać słowo

słowo → sklejone kilka rodzin

słowo, które weźmie wszystkie rodziny → jest jedno

NFA$(M_n) = \langle \dots, Q \rangle$ zał $Q \leq 2^{\frac{n}{2}-1}$

Słów jest więcej niż stanów. (coś szufl.)

Są jakieś słowa, że się zatrzymamy na tym samym stanie.

$v, w \in X$ $|w| \geq |v|$

$\forall_{w \in X}$ $ww^c \in M_n$ → jest przebieg akceptujący.

q_w → jakiś stan w przebiegu akceptującym

$\exists_{v_1 \neq v_2}$ t-że $q_{v_1} = q_{v_2}$

dopełnienie

$$q_0 \xrightarrow[v_2]{v_1} q_{v_1} \xrightarrow[v_2^c]{v_1^c} F$$

$q_0 \to v_1 v_2^c \to F$

JFIZO zad 35

a) CEL: zbudować DFA t. że rozpoznaje język sync(S).
TAK

czyli $\hat{\delta}'(q_0', w) = q'_{acc}$ $\quad \forall w \in sync(S)$

$S \subseteq Q$
$|S| = m$
$S = \{s_1, s_2, \dots, s_m\}$
→ wybrane stany z Q

$A = \langle \Sigma, Q, q_0, F, \delta \rangle$

$A' = \langle \Sigma, Q', q_0', F', \delta' \rangle$

$Q' = Q^{|S|}$ = krotka stanów z S

$q_0' = \langle s_1, s_2, s_3, \dots, s_m \rangle$ – wszystkie stany z S

$F' = \{ \langle q, q, \dots, q \rangle \mid q \in Q \}$ – wszystkie stany muszą być ściągnięte do tego samego.

$\delta'(\langle q_1, \dots, q_m \rangle, a) = \langle \delta(q_1, a), \delta(q_2, a), \dots, \delta(q_m, a) \rangle$

Weźmy dowolne $w \in sync(S)$.

$$\hat{\delta}'(q_0', w) \overset{def}{=} \langle \underset{q_k}{\underset{\parallel}{\hat{\delta}(s_1, w)}}, \underset{q_k}{\underset{\parallel}{\hat{\delta}(s_2, w)}}, \dots, \underset{q_k}{\underset{\parallel}{\hat{\delta}(s_m, w)}} \rangle = \langle q_k, q_k, q_k, \dots, q_k \rangle \quad ■$$

dla jakiegoś q_k, bo po przejściu słowa w będą w tym samym stanie (bo $w \in sync(S)$, więc $\forall s_i, s_j \ \hat{\delta}(s_i, w) = \hat{\delta}(s_j, w)$)

Czyli ten automat zaakceptuje w. ■

SYNC → ściągają do tego samego stanu.

c) TAK $w \in sync(Q)$, pokażemy, że $vwv' \in sync(Q)$

regularny ideał, czyli że jak dokleimy, to nie wyskoczymy poza język.

Weźmy dowolne $q, q' \in Q$ pokażemy, że vwv' ściągnie je do tego samego stanu.

$$\hat{\delta}(q, vwv') = \hat{\delta}(\underbrace{\hat{\delta}(\underbrace{\hat{\delta}(q, v)}_{q_k \in Q}, w)}_{q_w \text{ (ściągnięte do } q_w) \text{ bo } w \in sync(Q)}, v')$$

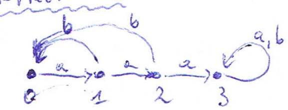
$$\hat{\delta}(q', vwv') = \hat{\delta}(\underbrace{\hat{\delta}(\underbrace{\hat{\delta}(q', v)}_{q_k' \in Q}, w)}_{q_w \text{ (ściągnięte do } q_w) \text{ bo } w \in sync(Q)}, v')$$

To samo, bo DFA od stanu q_w idzie po słowie v' do stanu $q_{v'}$ ■

Czyli dla każdego Q jest reg. ideałem.

b) NIE Intuicja: $w \in$ się udało bo po przejściu v od stanów q i q' wpadliśmy do stanów należących do sync(Q). Teraz

KONTRPRZYKŁAD



$ab \in sync\{q_0, q_1\}$

$\hat{\delta}(q_0, ab) = q_0$

$\hat{\delta}(q_1, ab) = q_0$

$\underbrace{a}_{v}\underbrace{ab}_{w}\underbrace{a}_{v'}$

$\hat{\delta}(q_0, aaba) = q_1$

$\neq$

$\hat{\delta}(q_1, aaba) = q_3$

Czyli nie ściągają do tego samego stanu, czyli $aaba \notin sync\{q_0, q_1\}$

Czyli nie jest regularnym ideałem ■

JFIZO zad 36

a) $S \subseteq Q$, $S = \{s_1, s_2\}$, $sync(S) \neq \emptyset \Rightarrow sync(S)$ zawiera w t. że $|w| \le |Q|^2$

Załóżmy nie wprost, że $\forall w \in sync(S)\ |w| > |Q|^2$.

Weźmy najkrótsze słowo $w \in sync(S)$, $w = a_1 a_2 \dots a_m$, $m > |Q|^2$

Zaczynamy od dwóch stanów s_1, s_2 i czytamy słowo w.

$s_1 \downarrow \quad \downarrow q$ } ten sam stan
$s_2 \quad \quad \downarrow q$
$a_1 a_2 \dots a_m$

$$\begin{array}{llllllll} X: & s_1 & \Rightarrow & \hat{\delta}(s_1, a_1) & \Rightarrow & \hat{\delta}(s_1, a_1 a_2) & \Rightarrow \dots \Rightarrow & \hat{\delta}(s_1, a_1 a_2 \dots a_m) = q \\ Y: & s_2 & & \hat{\delta}(s_2, a_1) & & \hat{\delta}(s_2, a_1 a_2) & \dots & \hat{\delta}(s_2, a_1 a_2 \dots a_m) = q \end{array}$$

(X, Y) możemy wybrać na $|Q| \cdot |Q|$ sposobów, bo to jakieś stany.

Skoro $m > |Q|^2$ to z zasady szufladkowej $\exists\, i < j$ t. że stany są identyczne

Np:

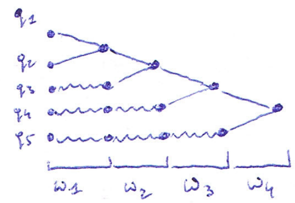
$$\begin{array}{c} s_1 \\ s_2 \end{array} \Rightarrow \begin{array}{c} 1 \\ 5 \end{array} \Rightarrow \dots \Rightarrow \underbrace{\begin{array}{c} 1 \\ 3 \end{array} \Rightarrow \dots \Rightarrow \begin{array}{c} 1 \\ 3 \end{array}}_{\text{Możemy wyciąć ten kawałek}} \Rightarrow \dots \Rightarrow \begin{array}{c} q \\ q \end{array}$$

$\uparrow i \qquad \uparrow j$

czyli jak weźmiemy słowo $a_1 a_2 a_3 \dots a_i a_{j+1} a_{j+2} \dots a_m$ to należy ono do $sync(S)$ (bo zbiegają do tego samego stanu) i jest ono krótsze od w, czyli w nie było najkrótsze w $sync(S)$ ↯

b) $sync(Q)$ to słowa, które zbiegają wszystkie stany do jednego.

Intuicja: Chcemy zbudować takie słowo.



Weźmy dwa dowolne stany z Q np q_1, q_2. Jest to podzbiór dwuelementowy, więc korzystając z a) możemy wybrać słowo $w_1 \in sync\{q_1, q_2\}$, które zbiegnie je do jednego. Itd.

$|Q| - 1$ słów, każde długości $\le |Q|^2 \Rightarrow w = w_1 w_2 \dots w_{n-1}$, $|w| \le |Q|^3$.

TAK NALEŻY BUDOWAĆ TO SŁOWO.

Algorytm:

$Q' = Q$

$w = \varepsilon$

dopóki $|Q'| > 1$

weź dowolne $q_1, q_2 \in Q'$ $q_1 \neq q_2$,

znajdź słowo v t. że $v \in sync\{q_1, q_2\} \wedge |v| < |Q|^2$

$w := wv$

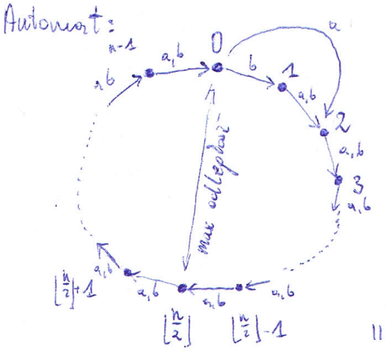
$Q' := Q' \cup \delta(q_1, v)$

−2 stany +1 stan
zmniejszamy o 1

JFIZO zad 37

Weźmy dowolne $n \in \mathbb{N}$, pokażemy jak zbudować taki automat i wybrać stany t. że $\forall w \in sync\{s_1, s_2\}\ |w| \geq \frac{n^2}{4}$

Automat:



$S = \{q_0, q_{\lfloor \frac{n}{2} \rfloor}\}$

$w \in sync(S)$ czyli ściąga stany q_0 i $q_{\lfloor \frac{n}{2} \rfloor}$ do tego samego. $\Rightarrow$ q_0 musi dogonić $q_{\frac{n}{2}}$ lub na odwrót.

„Dogonienie" może być tylko w stanie q_0

„Dogonić" możemy co n-1 literek. Musimy „dogonić" $\lfloor \frac{n}{2} \rfloor$ razy.

$w \in sync\{q_0, q_{\frac{n}{2}}\}$

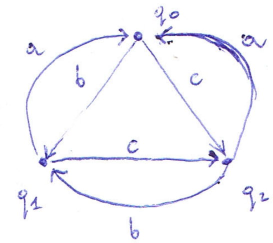
$$|w| \geq (n-1) \cdot \lfloor \tfrac{n}{2} \rfloor \geq (n-1) \cdot \frac{n-1}{2} \overset{?}{\geq} \frac{n^2}{4}$$

$$\frac{n^2 - 2n + 1}{2} \geq \frac{n^2}{4} \quad \text{dla} \quad n \geq 4$$

Czyli dla dostatecznie dużego n umiemy wskazać A i S. ∎

JFIZO zad 38 csync – słowa synchronizujące stany i przechodzące całe słowa (nie stuck)

NIE. Zbudujemy automat, że tak nie jest.



$\Sigma = \{a, b, c\}$

δ niezdefiniowane dla: $\langle q_0, a \rangle$

$\langle q_1, b \rangle$

$\langle q_2, c \rangle$

Rozważmy wszystkie 2-elem podzbiory $S \subseteq Q$.

$csync(\{q_0, q_1\}) = \{c\} \neq \emptyset$

$csync(\{q_1, q_2\}) = \{a\} \neq \emptyset$

$csync(\{q_0, q_2\}) = \{b\} \neq \emptyset$

$csync(Q) = csync(\{q_0, q_1, q_2\}) = \emptyset$ dlaczego?

$q_0 \neq q_1 \neq q_2$ więc $\varepsilon \notin csync(Q)$

od jakiej kolwiek literki zaczniemy, to dla któregoś ze stanów $q_1\ q_2\ q_3$ δ będzie niezdefiniowana, czyli słowo NIE BĘDZIE należało do $csync(Q)$ bo będzie „niebezpieczne"

Zad 39

Analogicznie, jak zadanie 36 a.

a)

$$q_A \longrightarrow \bullet \longrightarrow \cdots q_i \cdots q_i \cdots q_x$$
$$q_B \longrightarrow \bullet \longrightarrow \cdots q_j \cdots q_j \cdots q_x$$
$$q_C \longrightarrow \bullet \longrightarrow \cdots q_k \cdots q_k \cdots q_x$$

$$w = |\underset{a_1}{\text{—}}|\underset{a_2}{\text{—}}| \cdots | \cdots | \cdots |$$

weźmy najkrótsze słowo - niech ma długość $> 2|Q|^3$

wszystkich możliwych konfiguracji stanów jest $|Q|^3$.

$|w| > 2|Q|^3 > |Q|^3$ zatem z zas. szufl. dwie konfiguracje się powtórzą - wytnijmy literę między nimi → mamy krótsze słowo, a założyliśmy, że „w" jest najkrótsze.

↯

b) Analogiczne, krotki konfiguracji $< q_{a_1}, q_{a_2}, \ldots, q_{a_n} >$ wszystkich możliwych jest n^n - słabo, ale zauważmy, że permutacje są ok.

$$\bullet \longrightarrow \bullet \longrightarrow \cdots 1 \cdots 5 \cdots \longrightarrow X$$
$$\bullet \longrightarrow \bullet \longrightarrow \cdots 2 \cdots 1 \cdots \longrightarrow X$$
$$\vdots$$
$$\bullet \longrightarrow \bullet \longrightarrow \cdots 5 \cdots 2 \cdots \longrightarrow X$$

Jeśli umiemy dojść z 1, 2, 5 do X X X to deterministycznie te same litery doprowadzą 5, 1, 2 też do X X X.

Ile jest wszystkich? tyle ile podzbiorów Q, czyli $2^{|Q|}$.

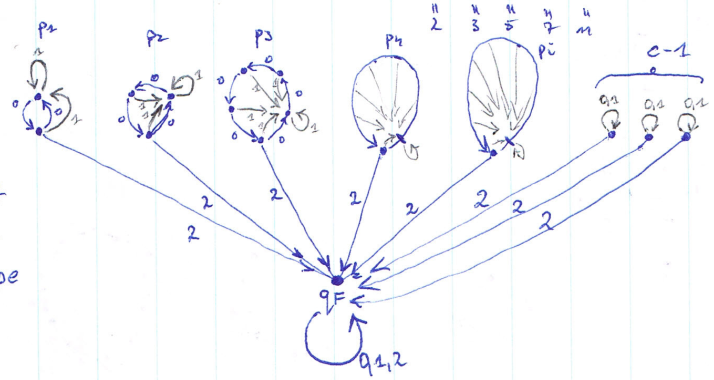
Jeśli weźmiemy słowo dłuższe niż $2^{|Q|}$ to znajdziemy dwie konfiguracje takie same lub spermutowane → możemy wyciąć litery i stworzyć krótsze.

40 ~~M~~ L, XL

Idea: Podobnie jak w M, zsynchronizujemy stany.

Dają nam wielomian $p(n)$. $2^{n\log n} > p(n)$ dla dużych n.

Automat ma $|Q|$ stanów: $|Q| = \underset{2}{p_1} + \underset{3}{p_2} + \underset{5}{p_3} + \underset{7}{p_4} + \underset{11}{p_5} + \dots + p_k + c$



0 = shift
1 = sync
2 = escape

Jak wygląda słowo synchronizujące?

* Jeśli zaczyna się od 0, to stany się przesuną na siebie, nic to nie da, czyli takie słowo nie jest najkrótsze!
* Jeśli zaczyna się od 2, to „utyka", czyli nie jest synchronizujące.

Czyli jest postaci: $\underbrace{1}_{\text{sync}}\,0^i\,\underbrace{2}_{\text{escape}}$ $i = p_1\cdot p_2\cdot p_3\cdot\ldots\cdot p_k$

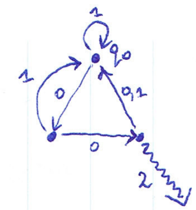
shift do stanu ucieczki   (NWW liczb pierwszych)

$n = p_1+p_2+p_3+\dots+p_k+c < k^2\log k + c$

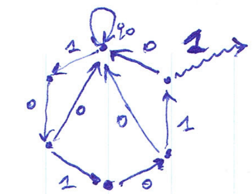
$|w| = p_1\cdot p_2\cdot p_3\cdot\ldots\cdot p_k + 2 > 2^{k\log k}+2$

Czyli długość NAJKRÓTSZEGO słowa sync jest wykładnicza względem $|Q|$

Teraz zakodujemy $\{0,1,2\}$ binarnie.



1 = 00 (sync)
0 = 10 (shift)
2 = 11 (escape)

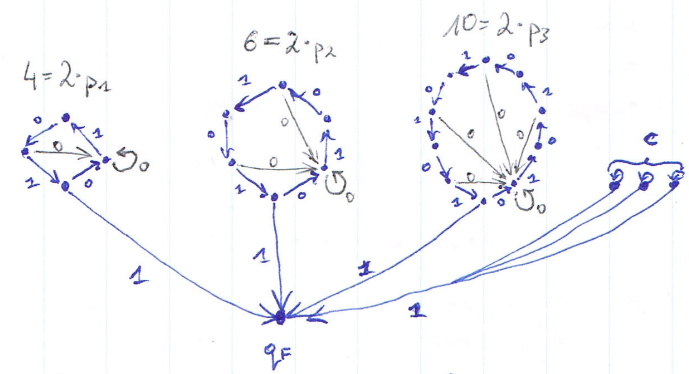


Jaki może być początek?

00 – ok
01 – ⊥
10 – shift, tylko wydłuża słowo
11 – ⊥

$\underbrace{00}_{1}\underbrace{(10)^i}_{0^i}\underbrace{11}_{2}$

Automat:

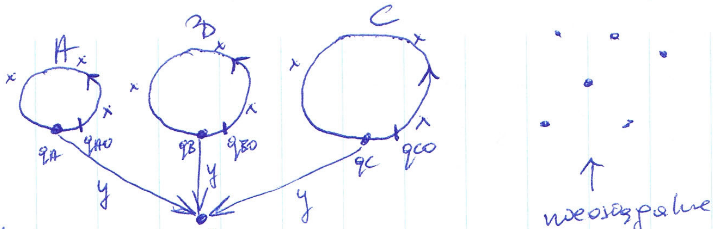


$n = 2\cdot p_1 + 2\cdot p_2 + 2\cdot p_3 + \dots + 2\cdot p_k + c < 2\cdot k^2\log k$

$|w| = 2p_1\cdot 2p_2\cdot\ldots\cdot 2p_k + 4 > 2^k\cdot 2^{k\log k}$

Ponownie, długość najkrótszego słowa sync jest wykładnicza względem $|Q|$.

40 M Automat:



$S = \{q_{A0}, q_{B0}, q_{C0}\}$

→ mogą się zsynchronizować w stanie q_F.

Ale najpierw muszą q_{A0}, q_{B0}, q_{C0} odpowiednio dojść do q_A, q_B, q_C.

Skoro A, B, C to liczby pierwsze to zsynchronizuje je tylko słowo, które jest ich NWW a potem y.

$$x^{A \cdot B \cdot C}\, y$$

Czy istnieją takie A, B, C pierwsze, że

$$\begin{cases} A+B+C+\underset{q_F}{1}+\underset{\text{jakieś nieosiągalne stany}}{l} = n \\ A \cdot B \cdot C > \frac{n^3}{10000} \end{cases}$$

Ze wskazówki:

$$0 < k < A < 2k < B < 4k < C < 8k < n$$

czy istnieje jakieś takie k naturalne?

Szacujemy największe możliwe A, B, C

$$2k + 4k + 8k < n$$

Szacujemy najmniejsze możliwe A, B, C

$$k \cdot 2k \cdot 4k > \frac{n^3}{10000}$$

$$14k < n \Rightarrow k < \frac{n}{14}$$

$$8k^3 > \frac{n^3}{10000} \Longrightarrow k > \frac{n}{2\sqrt[3]{10000}}$$

$$\frac{n}{2\sqrt[3]{10000}} < k < \frac{n}{14}$$

$$\Downarrow$$

$$\frac{n}{43}$$

dla dużych n istnieje takie k, czyli też A, B, C pierwsze. $\in \mathbb{N}$ ✓

JFIZO zad 41

$\langle a,b,c\rangle \in \mathbb{N}^3 : a+b=c$

pierwszy z 3-literowej krotki

$$L_{R_+} = \{ w \in \Sigma_3^* : R_+(l(\Pi_3^1(w)), l(\Pi_3^2(w)), l(\Pi_3^3(w))) \}$$

krotki 3-znakowe

l: zapis binarny od tyłu → wartość

Π_k^j → projekcja na j-ty wyraz z k-literowej krotki

DODAWANIE PISEMNE:

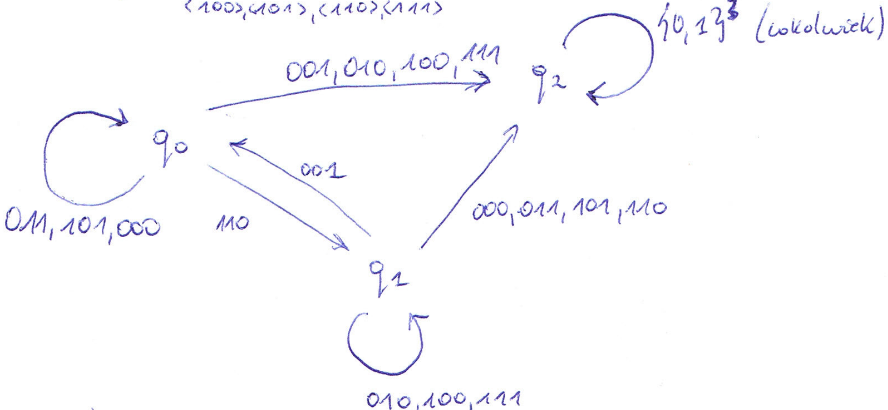
krotka z w

w=<110><010><111><001>

5	1	0	1	0
7	1	1	1	0
12	0	0	1	1
	2^0	2^1	2^2	2^3

Regularny → zbudujemy automat.

Możliwe ~~litery~~ w: <000>,<001>,<010>,<011>, <100>,<101>,<110>,<111>



q_0 - ok

q_1 - przeniesienie

q_2 - fail, nie da się wyjść

$F=\{q_0\}$

JFIZO zad 42

$\langle a,b,c\rangle \in \mathbb{N}^3 : a\cdot b=c$

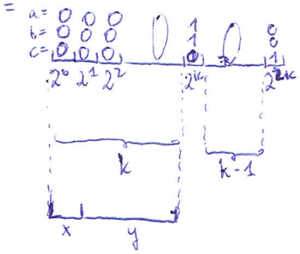
Analogicznie, jak w poprzednim zadaniu - odwrócone zapisy liczb binarnych.

TA RELACJA NIE JEST AUTOMATYCZNA.

Załóżmy nie wprost, że jest automatyczna, więc język L jest regularny.

Wtedy z lematu o pompowaniu: $\exists k.\ \forall w \in L\ |w|>k\ \ldots$ i $|xy|<k$.

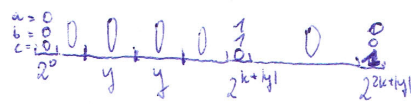
Weźmy słowo w =



$\in L$ bo $2^k \cdot 2^k = 2^{2k}$

Skoro $|xy| \leq k$ to ‚y' składa się z samych zer.

Weźmy słowo $w' = xy^2z$



$\notin L$ bo $2^{k+|y|}\cdot 2^{k+|y|} = 2^{2k+2|y|} \neq 2^{2k+|y|}$ ↯

$w' \notin L$ - sprzeczność.

Czyli ta relacja nie jest automatyczna.

43 Pokażemy, że R' jest relacją automatyczną

$\equiv$ zbudujemy automat.

R jest relacją automatyczną $\Rightarrow \exists$ DFA $\mathcal{A}_R = \langle \Sigma_2, Q, q_0, \delta, F \rangle$

Niech $\mathcal{A}_{R'}$ jest niedeterministycznym automatem:

$\mathcal{A}_{R'} = \langle \Sigma_1, Q, q_0, \delta', F \rangle$ gdzie δ' bierze się z δ

jeśli $\delta\left(q, \begin{bmatrix} r_x \\ m_y \end{bmatrix}, q'\right)$ to $\delta'(q, r_x, q')$

$\mathcal{A}_{R'}$ jest NDFA bo:

$$q_1 \xrightarrow{\begin{bmatrix} a_1 \\ b_1 \end{bmatrix}} q_2, \quad q_1 \xrightarrow{\begin{bmatrix} a_1 \\ b_2 \end{bmatrix}} q_3 \quad \Rightarrow \quad q_1 \xrightarrow{a_1} q_2, \quad q_1 \xrightarrow{a_1} q_3$$

Pokażemy, że $\mathcal{A}_{R'}$ rozstrzyga należenie do R'.

$R'(r) \Leftrightarrow \exists q_{ACC} \in F \ \hat{\delta}'(q_0, r, q_{ACC})$

($\Rightarrow$) $r \in R'$ czyli z def R' $\exists m$ t.że $\langle r, m \rangle \in R$

tzn $\mathcal{A}_R$ akceptuje słowo $\begin{bmatrix} r \\ m \end{bmatrix}$

czyli istnieje przebieg akceptujący

$$q_0 \xrightarrow{\begin{bmatrix} r_1 \\ m_1 \end{bmatrix}} q_x \longrightarrow \ldots \xrightarrow{\begin{bmatrix} r_l \\ m_l \end{bmatrix}} q_{ACC}$$

z definicji δ', będziemy mieli tę samą ścieżkę akceptującą dla automatu $\mathcal{A}_{R'}$ czyli $\exists q_{ACC} \in F \ \hat{\delta}'(q_0, r, q_{ACC})$ ∎

($\Leftarrow$) $\exists q_{ACC} \in F \ \hat{\delta}'(q_0, r, q_{ACC})$, weźmy ten przebieg

$$q_0 \xrightarrow{r_1} q_x \xrightarrow{r_2} \ldots \xrightarrow{r_l} q_{ACC}$$

Dołóżmy literki, z których powstały przejścia:

$$q_A \xrightarrow{r_i} q_B \quad \Rightarrow \quad q_A \xrightarrow{\begin{bmatrix} r_i \\ m_j \end{bmatrix}} q_B$$

Widzimy, że będzie przebieg akceptujący PO TYCH SAMYCH STANACH dla automatu $\mathcal{A}_R$, czyli $\exists m$ t.że $\langle r, m \rangle \in R$, zatem z definicji $r \in R'$. ∎

46 TAK, jest bezkontekstowy.

G:

$S \to \varepsilon \mid S1S0S \mid S0S1S \mid 1S1S0 \mid 0S1S1 \mid 1S0S1$

$L = L_G$

($\supseteq$) $w \in L_G \Rightarrow w \in L$

Na każde 0 przypada jedna 1 lub dwie.

($\subseteq$) nie wprost.

zał. że $\exists w \in L$ t.że $w \notin L_G$

weźmy najkrótsze.

1) $|w|_1 < 2 \cdot |w|_0$

wtedy w tym słowie istnieje 01 lub 10

$w = v_1 01 v_2$

$w' = v_1 v_2$, $v_1 v_2 \in L$ i $\in L_G$ zatem możemy wyprowadzić

↑ krótsze

$S \to v_1 S v_2 \to v_1 S0S1S v_2 \to w$ ✓

2) $|w|_1 = 2 \cdot |w|_0$

wtedy w tym słowie istn. 011 lub 110

n-zer → 2n jedynek

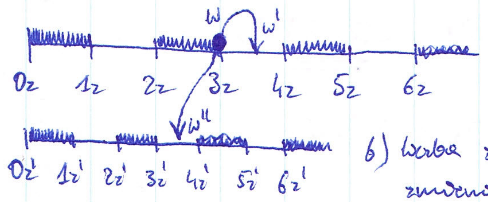
$\_0\_0\_0\_$ n+1 miejsc, czyli w jednym z nich będą dwie jedynki

$w = v_1 011 v_2$ czyli możemy wyprowadzić w analogicznie jak w 1°

3° Trzeba jeszcze zrobić bazę indukcji dla $|w| = 2$ lub $|w| = 3$.

47 Nie -> z lematu o pompowaniu.

Idea: dzielimy na przedziały względem $|w|_0$.
Pompujemy tak, że wpada do „złego" przedziału.



a) liczba zer nie ulega zmianie

b) liczba zer ulega zmianie, więc zmieniają się przedziały.

$$w = \left(1^{2p+1}0\right)^{2p+1}$$

Dla dowolnego podziału $sztyx$, $|zty| \le p$ zy ma:

a) 0 zer, x jedynek, $1 \le x \le p$

b) 1 zero, x jedynek, $0 \le x \le p-1$

a) $w' = sz^2ty^2x$

$|w'|_0 = |w|_0 = \underline{2p+1}$

$|w'|_1 = |w|_1 + x = (2p+1)(2p+1) + x$

$$(2p+1)\underline{(2p+1)} < (2p+1)(2p+1)+x < (2p+2)\underline{(2p+1)}$$

$$4p^2+4p+1 < 4p^2+4p+1+x < 4p^2+6p+1$$

$x \in [1, p]$ więc to prawda

czyli $w' \notin L$ ↯

b) $w' = sz^3ty^3x$

$|w'|_0 = |w|_0 + 2 = \underline{2p+3}$ — intuicja: musi być tu liczba nieparzysta, żeby potem z lewej strony nierówności była nieparzysta.

$|w'|_1 = |w|_1 + 2x = (2p+1)(2p+1) + 2x$

$$\underline{(2p+3)}(2p-1) < (2p+1)(2p+1)+2x < \underline{(2p+3)}2p$$

$$4p^2+4p-3 < 4p^2+4p+1+2x < 4p^2+6p$$

OK

$\cancel{4p^2} 1 + 2x < 2p$

$1 + 2p - 2 < 2p$

$-1 < 0$ ✓

Czyli $w' \notin L$. ↯

48 $|\Sigma| = 1$, $L \in CFL \iff L \in reg$

(⇐) Każdy język regularny jest bezkontekstowy.

(⇒) Dla języka L zachodzi lemat o pompowaniu.

p – stała z lematu.

$L = L_1 \cup L_2$

$L_2 \to \{w \in L \mid |w| > p\}$

$L_1 \to \{w \in L \mid |w| \le p\}$

słów z L_1 jest skończenie wiele, więc jest to język regularny.

Czy L_2 jest regularny?

Weźmy słowo $w \in L_2$, jest odpowiednio długie, więc istnieje dla niego podział $sztyx$. Wtedy też $w' = sz^ity^ix = stx(zy)^i$ należy do języka dla każdego $i \in N$.

Skoro $|zty| \le p$ to $|zy| \le p$ czyli mamy skończenie wiele końcówek (rozróżniamy tylko długości).

$$w' = \underbrace{stx}_{k\cdot a+b}\underbrace{(zy)}_{a}{}^i \qquad |w'| = k\cdot a + b + a^i = a(k+a^{i-1}) + b$$

reszta $|stx|$ mod a

$L_{a,b} = \{w \in L_2 \mid |w| = k\cdot a + b$ dla $k \in N\}$

$L_2 = \bigcup_{a,b} L_{a,b}$; skoro $\underset{|zy| \le p}{a \le p}$ i $\underset{\text{mod } p}{b \le p}$ to mamy sumę skończonej ilości języków. Jeśli $L_{a,b}$ reg to L_2 reg czyli L reg.

Czy $L_{a,b}$ jest językiem regularnym?

* Jeśli nie istnieje słowo $w \in L_2$ t.że $|w| = k\cdot a + b$ to $L_{a,b}$ jest pusty czyli regularny.
* Jeśli istnieje słowo $w \in L_2$ t.że $|w| = k\cdot a + b$ to weźmy najkrótsze. Możemy je rozpoznać wyrażeniem regularnym

$0^b(0^a)^*$ czyli to język regularny ✓.

$\forall_{a,b}$ $L_{a,b} \in reg \Rightarrow L_2 \in reg \Rightarrow L \in reg$ ✓

Zad 49 $G: S \to \neg S \mid (S \Rightarrow S) \mid p \mid q$

G': Trzeba rozbić produkcje $S \to \neg S$ na $S \to NS$
$N \to \neg$

Trzeba rozbić produkcję $S \to (S \Rightarrow S)$

$S \to \underbrace{\underbrace{\underbrace{(S}_{LS \,\rightsquigarrow\, X}\ \underbrace{\Rightarrow}_{\rightsquigarrow\, A}}_{Z}\ \underbrace{S)}_{SR \,\rightsquigarrow\, Y}}$

$S \to ZY$
$Y \to SR$
$R \to ')'$
$Z \to XA$
$A \to '\Rightarrow'$
$X \to LS$
$L \to ')'$

Twierdzenie: $L_G = L_{G'}$, indukcja po długości wyprowadzenia.

($\subseteq$) weźmy dowolne $w \in L_G$

1° długość wyprowadzenia to 1

$w = p$ lub $w = q \to w \in L_{G'}$

2° zał., że $\forall v$ o wyprowadzeniu długości n jest ok

pierwszym krokiem przy tworzeniu w o dł. wyprowadzenia $n+1$ było:

a) $S \to \neg S$ → czyli słowo $w = \neg v$

v się da w $L_{G'}$, $\neg v$ też: $S \to NS \to \neg S$

b) $S \to (S \Rightarrow S)$ — " — – – –

($\supseteq$) analogicznie, po długości wyprowadzenia.

JFI20 zad 50

$L_3 = \{ w \in \{0,1,2\}^* : \neg\exists_{x \in \{0,1,2\}^*}\ w = xx \}$

TAK - jest bezkontekstowy

G: $\Pi = \{$
$S \to A \mid B \mid C \mid AB \mid AC \mid BA \mid BC \mid CA \mid CB$
$X \to 0 \mid 1 \mid 2$
$A \to XAX \mid 0$
$B \to XBX \mid 1$
$C \to XCX \mid 2$
$\}$

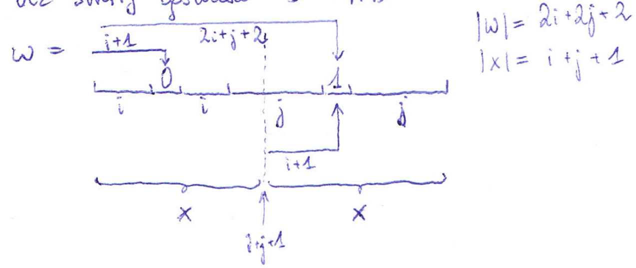
$L_3 = L_G$

($\supseteq$) $w \in L_G \Rightarrow w \in L_3$

1° Wyprowadzenie zaczyna się od $A \mid B \mid C$
→ wtedy $|w|$ jest nieparzysta, więc nie da się podzielić na pół. ✓

2° Wyprowadzenie zaczyna się od $AB \mid AC \mid BA \mid BC \mid CA \mid CB$
bez straty ogólności $S \to AB$

w =

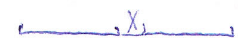


$|w| = 2i + 2j + 2$
$|x| = i + j + 1$

Jeśli podzielimy na pół, to słowa różnią się na $i+1$ pozycji
czyli nie da się podzielić na xx, czyli $w \in L_3$ ✓

($\subseteq$) $w \in L_3 \Rightarrow w \in L_G$ czyli daje się wyprowadzić

1° $|w|$ - nieparzyste →

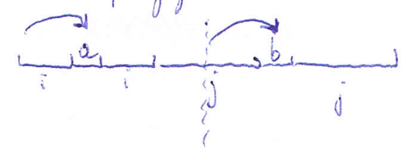


patrzymy na środek i wyprowadzamy ✓.

2° $|w|$ - parzyste →

$w \in L_3$ czyli połówki się różnią na $i+1$ pozycji.

Czyli można zapisać jako



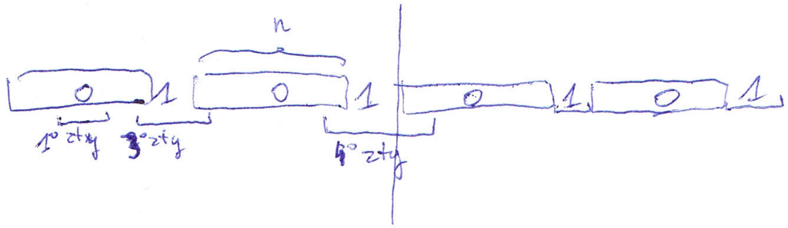
Czyli da się wyprodukować

$S \to AB \to \underbrace{X \ldots X}_{i} A \underbrace{X \ldots X}_{i}\ \underbrace{X \ldots X}_{j} B \underbrace{X \ldots X}_{j}$

zatem $w \in L_G$ ∎

51

NIE JEST BEZKONTEKSTOWA



1° $|zty|_1 = 0$ - same zera

pompujemy → środek się przesuwa x_1 kończy się '0'

x_2 kończy się '1'

2° $|zty|_1 \geq 2$ - niemożliwe

3° $|zty|_1 = 1$

a) $1 \in zy$ (do z lub do y)

$stx \notin L_4$ bo ma za dużo jedynek

b) $1 \in t$ → argument jak w 1° $sz^0\underbrace{1}_{t}y^0x$

4°

$0^n 1 0^{n-i} 1 0^{n-j} 1 0^n 1$ - nie jest kwadratem żadnego słowa.

Zad 54 $0^n 1^{n^2}$ jest bezkontekstowy?

NIE. Załóżmy że jest → zachodzi lemat o pompowaniu, stała k.

Niech $w = 0^k 1^{k^2}$. Pokażemy, że nie istnieje dobry podział sztyx.

$|zty| \leq k$

1° zty to same zera

2° zty to same jedynki > jak rozpompujemy, to źle, bo dodamy '1' a nie dodamy '0' lub na odwrót.

3° zty ma n zer i m jedynek. Rozpatrzmy sz^2ty^2x

1) 1 zero → $w' = 0^{k+1} 1^{k^2+c}$, $c \in [0, k-1]$

$$(k+1)^2 = k^2 + 2k + 1 > k^2 + k - 1 \geq k^2 + c$$

→ nie zachodzi

2) 2 zera → $w' = 0^{k+2} 1^{k^2+c}$, $c \in [0, k-2]$

→ z [illegible]

⋮

k) k zer

→ źle

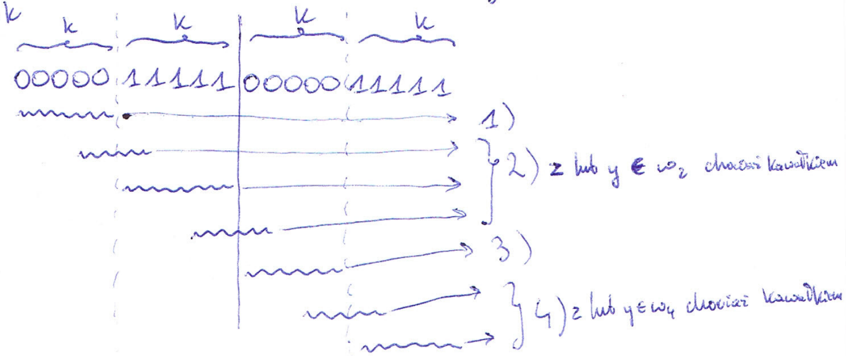
→ zatem nie istnieje taki podział

czyli język nie jest bezkontekstowy.

JFIZO zad 55 $\underbrace{\#1 \geq n}$ | $\underbrace{\#1 = n}$

Załóżmy, że jest bezkontekstowy. ~~[illegible]~~ Zatem zachodzi lemat o pompowaniu. k – stała z lematu. Weźmy $w = 0^k1^k0^k1^k \in L$.

$|zty| \leq k$



1) Pompujemy tak długo, aż wszystkie '1' znajdą się po prawej stronie. ↯

2) sz^0ty^0x – zmniejszy się ilość '1' po lewej, a po prawej nie. ↯

3) sz^0ty^0x – po odpompowaniu '0' środek przesunie się w lewo, czyli jedynki z lewej przejdą na prawo ↯

4) sz^2ty^2x – rozpompujemy jedynki, zwiększy się ich ilość po prawej. ~~[illegible]~~ Środek przesunie się w prawo, ale nie dojdą nawet ~~[illegible]~~ jedynki, bo dodanie dwóch znaków przesuwa środek o 1 w prawo.

$|zty| \leq k$

$|z^2ty^2| < 2k$

56 Weźmy $L = a^n b^m c^m d^{3n}$, $\quad S \to aSddd \mid T$, $\quad T \to \varepsilon \mid bTc$

Zał, że $L/2 \in CFL$, zachodzi lemat o pompowaniu.

p - stała z lematu. Weźmy $w = a^p b^p c^p$

$w \in L/2$, bo $\underbrace{a^p b^p c^p}_{3p} \underbrace{d^{3p}}_{3p} \in L$

Pokażemy, że każdy podział $sztyx$ jest zły.

Jakie może być zty? $|zty| \leq p$, może mieć max dwie różne literki.

1° zy to same literki a

$sz^0ty^0x = a^{p-i} b^p c^p$

$\underbrace{a^{p-i} b^p c^p}_{3p-i} \underbrace{d^{3(p-i)}}_{3p-3i}$ $\qquad 3p - i = 3p - 3i$

$i = 0$ ↯ bo $zy \neq \varepsilon$

2° zy to same literki b

$sz^0ty^0x = a^p b^{p-i} c^p$

literek b mniej niż c ↯

3° zy to same literki c

$sz^2ty^2x = a^p b^p c^{p+i}$

literek b jest mniej niż c ↯

4° zy ma ,i' literek a oraz ,j' literek b

$sz^0ty^0x = a^{p-i} b^{p-j} c^p$

literek b jest mniej niż c

5° zy ma ,i' literek b oraz ,j' literek c

$sz^0ty^0x = a^p b^{p-i} c^{p-j}$

a) $i > j$, wtedy literek b jest więcej niż c ↯

b) $i = j$, wtedy:

$\underbrace{a^p b^{p-i} c^{p-i}}_{3p-2i} \underbrace{d^{3p}}_{3p}$ $\qquad 3p - 2i = 3p$

$i = 0$ ↯ bo $zy \neq \varepsilon$

c) $i < j$

$\underbrace{a^p b^{p-i} c^{p-j}}_{3p-i-j} \underbrace{c^{j-i} d^{3p}}_{3p+j-i}$ $\qquad 3p - i - j = 3p + j - i$

$j = 2i$

$j < \cancel{2}i$ ↯

bo założyliśmy tu, że $i < j$, czyli się nie da.

Inne rozw.:

Weźmy $L = \{a^n b^m c^m \# d^{3n}\}$ ← to musi być podwójne

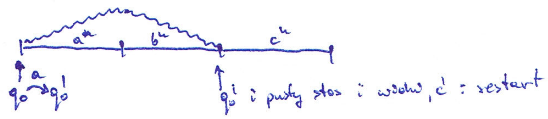
$L/2 = \{a^n b^m c^m \#\}$

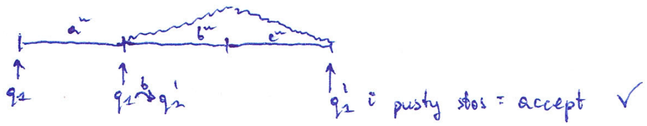
$L/2 \cap a^* b^* c^* \# \in CFL$ ← regularny

$= a^n b^n c^n \#$ ← a to nie jest CFL ↯, czyli $L/2$ nie jest CFL

57 Weźmy język $a^n b^n c^n$, nie jest on bezkontekstowy.

Pokażemy schemat działania automatu:





$F = \{q_0, q_1'\}$ przy pustym stosie.

Tw $w = a^n b^n c^n \iff \mathcal{A}$ akceptuje w

(⇒) * jeśli $n=0$ to automat zaakceptuje w stanie q_0. ✓

* jeśli $n>0$ to słowo $w \neq \varepsilon$, czyli wejdzie do stanu q_0' i dorzuci 'a' na stos; następnie zrzuci wszystkie 'a' i będzie miał pusty stos, zobaczy 'c' więc restart w q_1.

q_1 ignoruje 'a', aż zobaczy 'b' i zmieni stan na q_1' dorzucając 'b' na stos. Zdejmie to wszystko potem przez 'c'.

Czyli będzie w q_1' z pustym stosem ✓

(⇐) $\mathcal{A}$ akceptuje w czyli albo jest w q_0 albo w q_1' z pustym stosem.

* jeśli jest w q_0 to $w = \varepsilon$ ✓
* jeśli jest w q_1' to przebieg akceptujący to $q_0 \to q_0' \to q_1 \to q_1'$

musi być więc tyle samo 'a' co 'b', oraz tyle samo 'b' co 'c'

co implikuje $w = a^n b^n c^n$ ✓

~~58~~ Zad 58

Mamy automat z restartem może zacząć w stanach $q_0, q_1, \ldots, q_{k-1}$ (po którymś restarcie).

Możemy to zasymulować zwykłym DFA.

$MASR = \langle \Sigma, Q, q_0, k, F, \delta \rangle$, $n = |Q|$

k – ilość restartowania

DFA = $\Sigma' = \Sigma$

$Q' = Q^k$

$q_0' = \langle q_0, q_1, \ldots, q_{k-1} \rangle$

$F' = \{ \langle r_0, r_1, \ldots, r_{k-1} \rangle \mid \exists i \ r_i \in F \}$

$\delta'(\langle r_0, r_1, \ldots, r_{k-1} \rangle, a) = \langle \delta(r_0,a), \delta(r_1,a), \ldots, \delta(r_{k-1},a) \rangle$

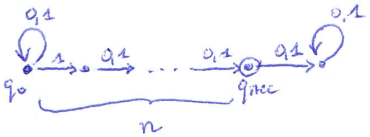
MASR success $\iff$ DFA success

(⇒) mamy automat z restartem po którymś restarcie wystartuje ze stanu q_i i skończy w q_F, a więc będzie istniało takie $i<k$, że $r_i \in F$ w tej krotce DFA zaakceptuje bo wystartowało z q_i.

(⇐) DFA akceptuje, czyli któryś ze stanów w krotce akceptuje. Niech będzie to 'i' → czyli zaczynając od stanu q_i automat z restartem zaakceptuje (być może wcześniej).

59. $L_n = \{w1v \mid |v| = n-1\}$

NFA potrzebuje $n+c$ stanów



DFARestart potrzebuje 2^n stanów.

FAKT Dla każdego DFARestart istnieje słowo v, t. że po jego przeczytaniu automat NIE MOŻE się zrestartować, jeśli chce zaakceptować słowo (bo by się zapętlił).

Weźmy takie v. Niech w_i to wszystkie RÓŻNE słowa dł. n. Jest ich 2^n. Zał, że DFARestart ma mniej niż 2^n stanów.

$w_1' = v w_1$

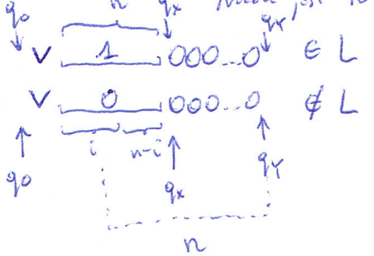
$w_2' = v w_2$

$\vdots$

$w_{2^n}' = v w_{2^n}$

Istnieją dwa takie słowa, że automat po ich przeczytaniu jest w tym samym stanie (zas. szufladkowa).

Skoro są różne, to weźmy pierwsze miejsce w którym się różnią. Niech jest to miejsce 'i', dopiszmy 0^{i-1}

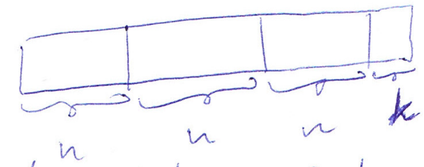


Automat jest w stanie, który musi być akceptujący i nieakceptujący SPRZECZNOŚĆ!

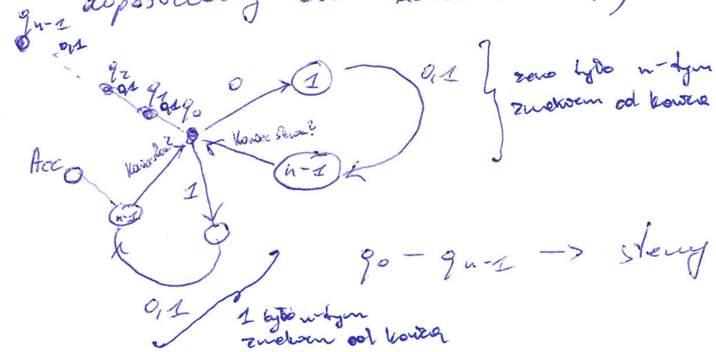
Czyli DFARestart potrzebuje przynajmniej 2^n stanów ✓.

60. $L_n = \{w1v \mid |v| = n-1\}$

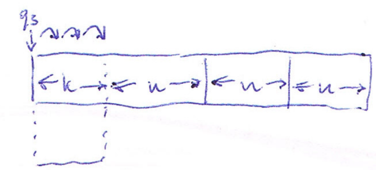
Mamy – mówiący czy literka jest ostatnia.



Automat z restartem może 'przesuwać' kolejno te paski aż dojdzie do k (czyli ostatni pasek dopasowany do końca słowa)



$q_0 - q_{n-1}$ —> stany rozpoczynające.



61

$\Sigma_n = \{1, 2, \dots, n-1\}$

$S_n = \{w \in \Sigma^* :$ słowa, w których wyst. każda litera $\}$

DFA: 2^{n-1} stanów (różne podzbiory Σ_n)

RDFA:

q_1 (pętla: $2, 3, \dots, n-1$) $\xrightarrow{1} R$ – czy w tym słowie jest 1

q_2 (pętla: $1, 3, 4, \dots, n-1$) $\xrightarrow{2} R$ – czy w tym słowie jest 2

$\vdots$

q_{n-1} (pętla: $1, 2, \dots, n-2$) $\xrightarrow{n-1} R$ – czy w tym słowie jest $n-1$

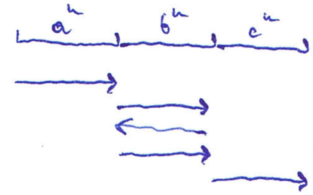
$q_n \to$ Accept

Zad 62

Weźmy język $\{a^n b^n c^n \mid n \in \mathbb{N}\}$

$\to$ nie jest CFL bo nie zachodzi lemat o pompowaniu.

2NPDA jest w stanie go rozpoznać:



1 odkładaj 'a' na stos do wystąpienia 'b'
2 zdejmuj ze stosu 'a' aż zobaczysz 'c'
3 wróć aż zobaczysz 'a'
4 odkładaj 'b' na stos
5. zdejmuj ze stosu 'b' aż do końca

63 a) Splecenie

* $S(\varepsilon,\varepsilon,\varepsilon)$
* $S(s,t,w) \Rightarrow S(t,s,w)$
* $S(s,t,w) \Rightarrow S(s,at,aw)$

$S(\varepsilon,\varepsilon)\to\varepsilon$

$S(a,\varepsilon)=S(\varepsilon,a)\to a$

$S(a,b)=S(b,a)\to ab \,/\, ba$

$S(a_1a_2,\varepsilon)\to a_1a_2$

$S(a_1a_2a_3,\varepsilon)\to a_1a_2a_3$

$S(a_1a_2,b_1)\to b_1a_1a_2 \,/\, a_1b_1a_2 \,/\, a_1a_2b_1$

$S(a_1a_2,b_1b_2)\to b_1b_2a_1a_2 \,/\, b_1a_1b_2a_2 \,/\, b_1a_1a_2b_2 \,/\, a_1a_2b_1b_2 \,/\, a_1b_1a_2b_2 \,/\, a_1b_1b_2a_2$

Zauważmy, że splecenie miesza literki, ale ZACHOWUJE ich kolejność.

$$S(L_1,L_2)=\{w\in\Sigma^* \mid \exists w_1\in L_1, w_2\in L_2 \ S(w_1,w_2,w)\}$$

Mamy DFA $\mathcal{A}_1$ i $\mathcal{A}_2$ rozpoznające L_1 i L_2.

Zbudujemy NFA $\mathcal{A}'$ rozpoznający język $S(L_1,L_2)$

IDEA: Automat zgaduje czy literka należy do słowa w_1 czy do w_2.

$\mathcal{A}'=\langle \Sigma, Q=Q_1\times Q_2, q_0=\langle q_{1,0},q_{2,0}\rangle, \delta', F'\rangle$

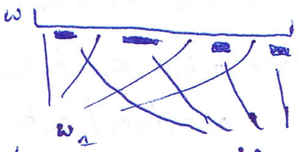
$F'=\{\langle q_{1,i},q_{2,j}\rangle \mid q_{1,i}\in F_1 \wedge q_{2,j}\in F_2\}$

δ':
* $\delta'(\langle q_{1,i},q_{2,j}\rangle, a; \langle \delta_1(q_{1,i},a), q_{2,j}\rangle)$ // literka pochodzi z w_1
* $\delta'(\langle q_{1,i},q_{2,j}\rangle, a; \langle q_{1,i}, \delta_2(q_{2,j},a)\rangle)$ // literka pochodzi z w_2

$\mathcal{A}'$ akceptuje $w \Leftrightarrow w\in S(L_1,L_2)$

($\Rightarrow$) po całym przejściu automat jest w stanie akceptującym, czyli udało się wybrać podsłowa w_1 i w_2, które należą do języków L_1 i L_2. $S(w_1,w_2,w)$.

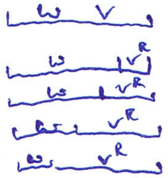
($\Leftarrow$) $w\in S(L_1,L_2)$ czyli można podzielić w na podsłowa t.że $w_1\in L_1$ i $w_2\in L_2$: w



A więc automat będzie w stanie akceptującym dla w_1 z lewej strony pary i dla w_2 z prawej strony pary, czyli będzie to stan akceptujący $\mathcal{A}$ ✓.

Zad 64

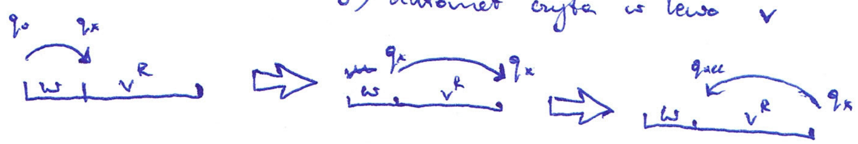
L regularny $\Rightarrow \{wv^R \in \Sigma^* \mid wv \in L\}$ regularny.



→ miejsce podziału jest niewiadomo gdzie

Skonstruujemy 2NFA:

1) duch mówi, kiedy jest koniec słowa „w"
2) automat idzie na koniec słowa
3) automat czyta w lewo v



Zad 65

$L - n$

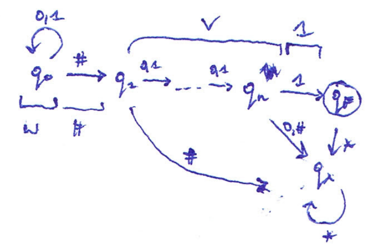
$\text{Lustro}(L) - 2^n$

Jako znany „trudny" język? Np. $\{w \mid 1$ jest n-tym od końca znakiem$\}$

$D_n = \{w1v\# \mid |v| = n\}$ ← 2^n stanów

Trzeba zgadnąć, kiedy jest ta „1".

$L = \{w\#v1 \mid |v| = n;\ w,v \in \{0,1\}^*\}$ ← to jest łatwe, bo „#" mówi, kiedy będzie n znaków o koniec słowa.



$\approx n$ stanów

$D_n = \underbrace{\text{Lustro}(L)}_{L_1} \cap \underbrace{\{0,1\}^*\#}_{L_2}$

wymuszamy lustro tutaj

$w|\#v1$

↳ hash wtedy na końcu wv^R

DFA(L_2) → ≈ 2 stany

DFA(D_n) → $\approx 2^n$ stanów

$\Rightarrow$ DFA(L_1) = $\approx 2^{n-1}$ stanów, czyli wykładniczo.

66 $L \in CFL \Rightarrow Lustro(L) \in CFL$? NIE.

Weźmy $L = x\#\#x^R$, palindromy, oczywiście L jest bezkontekstowy.

$S \to \#\# \mid aSa$, $a \in \Sigma$

Załóżmy, że Lustro(L) jest CFL.

Wtedy $L' = Lustro(L) \cap \underbrace{\Sigma\Sigma^*}_{\neq \varepsilon}\#\underbrace{\Sigma\Sigma^*}_{\neq \varepsilon}\#$ jest bezkontekstowy

bo CFL $\cap$ reg = CFL. Zauważmy, że wymuszamy to, że hashe są ROZDZIELONE, a to oznacza, że lustro MUSIAŁO być ustawione pomiędzy nimi. $x\#|\#x^R$ ← lustro

Zatem $L' = \{ x\#(\#x^R)^R \} = \underbrace{x\#}_{y}\underbrace{x\#}_{y}$

A my wiemy, że $\{ww \mid w \in \Sigma^*\}$ NIE JEST bezkontekstowy.

↯ czyli L' nie jest CFL, czyli Lustro(L) nie jest CFL. ∎

Zad 69 $L_r \subseteq L_{a^*ba^*}$

Niech $L = \{a^n b a^n \mid n \in \mathbb{N}\}$, $L \subseteq L_r$

L^* jest bezkontekstowy, istnieje gramatyka:

$S \to SS$
$S \to aBa \mid b$
$B \to aBa$
$B \to b$

$S \to SS \to \ldots \to SSSS \to \underbrace{a^3ba^3}_{w_1}\underbrace{a^7ba^7}_{w_2}\underbrace{b}_{w_3}\underbrace{a^4ba^4}_{w_4}$

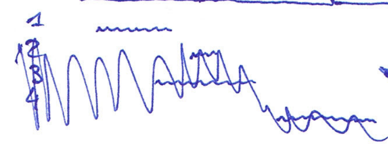
takie słowa należą do L^*.

L^* nie jest regularny:

Zał., że jest, zatem lemat o k stałej. Weźmy $w = a^k b a^k a^k b a^k$.

$w \in L^*$, $|w| \geq k$. Jak może być podział xyz? $(|xy| \leq k)$

$a^k\ b\ a^k\ a^k\ b\ a^k$



$|xy| \leq k$

1 - xy^2z → zamieszczamy ilość a po lewej - ŹLE

2 - xy^0z → $a^k a^k b a^k = a^{3k} b a^k \notin L^*$ - ŹLE

3 - xy^0z → $a^{c_1} a^{c_2} a^k b a^k$ - ŹLE

4 - xy^0z → $a^k b a^{c_1} a^{c_2} b a^k$ - $c_1 < k$, $c_2 < k$ - ŹLE.

↯

Zatem L^* nie jest regularny.

JFIZO zad 72.

1. regularny $\Rightarrow$ konfluentny?

NIE $\quad \varphi = (00)^*$ $\quad \Sigma = \{0\}$

↑ parzysta długość

$w = 00$

$v = 0$ $\quad wxy \in L_\varphi \Leftrightarrow vxy \in L_\varphi$

Nie da się, bo $|w| \% 2 \neq |v| \% 2$, czyli $|wxy| \% 2 \neq |vxy| \% 2$

Czyli zawsze jedno słowo $\in L_\varphi$ a drugie $\notin L_\varphi$

2. konfluentny $\Rightarrow$ regularny?

NIE $L = \{a^n b^n \mid n \in \mathbb{N}\}$ nie jest regularny (lemat o pompowaniu)

ale jest konfluentny: $x = ba$

$\forall y \quad w\underline{ba}y \notin L \wedge v\underline{ba}y \in L$

czyli $\quad wbay \in L \Leftrightarrow vbay \in L$

JFIZO zad 73.

regularny $\wedge$ konfluentny $\Rightarrow$ jednostajnie konfluentny

Czyli „x" słowo „synchronizujące" jest skończone (możemy ograniczyć jego długość)

$w \sim_L w'$ wtw. gdy $\forall v \in \Sigma^* (wv \in L \Leftrightarrow w'v \in L)$.

{Tw o indeksie}
L regularny $\Leftrightarrow$ liczba klas abstrakcji $\sim_L$ jest skończona.

Skoro L regularny to mamy n klas abstrakcji $K_0, K_1, \ldots, K_{n-1}$

„w_i" oznacza reprezentanta klasy abstrakcji K_i

L jest konfluentny zatem:

$\forall w, v \in \Sigma^*. \exists x \in \Sigma^*. \forall y \in \Sigma^*. \quad wxy \in L \Leftrightarrow vxy \in L$

$w \in K_i, v \in K_j \quad i,j \in 0, 1, \ldots, n-1$

Czyli dla każdej pary $\langle K_i, K_j \rangle$ mamy jakieś x_{ij} i znamy jego długość.

Weźmy $c = \max\{|x_{ij}| : i,j \in 0, \ldots, n-1\}$

Wtedy wszystkie $|x_{ij}| \leq c$, czyli L jest jednostajnie konfluentny

Weźmy dowolne $v_1, v_2 \in \Sigma^*$ $\quad v_1 \sim_L w_i, v_2 \sim_L w_j$ $\quad x = x_{ij}$ $\quad |x_{ij}| \leq c$

$\forall y \in \Sigma^* \quad v_1 x_{ij} y \in L \Leftrightarrow v_2 x_{ij} y \in L$?

$$\forall y \in \Sigma^*: \underbrace{v_1}_{w} \underbrace{x_{ij} y}_{v} \in L \Leftrightarrow \underbrace{\overbrace{w_i x_{ij}}^{wx}}_{w'} \underbrace{\overbrace{y}^{y}}_{v} \in L \Leftrightarrow \underbrace{\overbrace{w_j x_{ij}}^{vx}}_{w'} \underbrace{\overbrace{y}^{y}}_{v} \in L \Leftrightarrow \underbrace{v_2}_{w} \underbrace{x_{ij} y}_{v} \in L$$

def $v_1 \sim_L w_i$

konfluentny i $x = x_{ij}$ dla tej pary

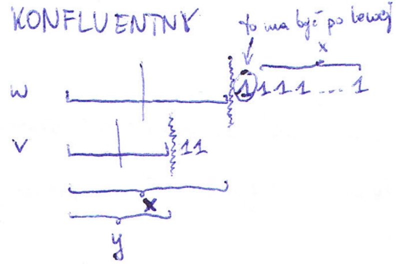
JFIZO zad 74

konfluentny $\wedge$ bezkontekstowy $\wedge$ $\neg$ jednostajnie konfluentny

$\Sigma = \{0,1\}$

$L = \{0^n(0+1)^m \mid n \geq m\}$ – czyli w pierwszej połowie słowa nie ma żadnej jedynki.

1) KONFLUENTNY



Dopiszmy $\max\{|w|,|v|\}+1$ jedynek, wtedy na prawo '1' w kółeczku będzie po lewej stronie. $\forall y$ $wxy \notin L \wedge vxy \notin L$

czyli $\exists x$ $\_ \notin L \Leftrightarrow \_ \in L$

czyli jest konfluentny.

2) NIE JEDNOSTAJNIE KONFLUENTNY

Zał, że jest jedn. konf., czyli $\exists c$ $\forall x$ $|x| \leq c$. Weźmy słowa:

$w = 1$

$v = \underbrace{000\ldots0}_{c}$

wxy nigdy nie należy do L. Musimy znaleźć x, t. że $vx \notin L$.

$\underbrace{000\ldots0}_{c}\underbrace{\underbrace{1,1\ldots\ldots1}_{c}}_{c+1}$ czyli $|x| = c+1$. Sprzeczność! ↯

Zatem L nie jest jednostajnie konfluentny ■

3) BEZKONTEKSTOWY

G: Π: $S \to \varepsilon \mid 0S \mid 0S0 \mid 0S1$

$L = L_G$

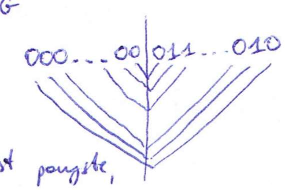
($\Leftarrow$) $w \in L_G \Rightarrow w \in L$

W każdej produkcji albo stawiamy '1' na prawo od środka albo przesuwamy środek w lewo (symbol S na prawo).

($\Rightarrow$) $w \in L \Rightarrow w \in L_G$

1° $|w|$ parzyste:

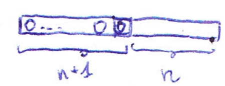
Zapiszmy w:



skoro $w \in L$ i jest parzyste, to możemy poparować, trzymać S na środku i używać $0S0$ i $0S1$.

2° $|w|$ nieparzyste:

$w \in L$ więc jest postaci



czyli można zrobić $S \to 0S \to \ldots$ i analogicznie jak dla parzystych.

75

a) $L_{3/2}$ nie jest regularny. Intuicja: musielibyśmy umieć liczyć długości

$L = L_{a^*}$

Zał, że $L_{3/2}$ jest regularny. p - stała z lematu.

$w = a^p a^p b^p$, $w \in L_{3/2}$, powinien istnieć podział xyz.

$|xy| < p$ więc y to same literki a.

$xy^0z = a^{2p-i} b^p \notin L_{3/2}$, bo jak weźmiemy $\frac{2}{3}$ tego słowa to jest ono postaci $a^i b^j \notin L$. ↯

b weszło do tego kawałka

b) $\bigcup_{i=1}^{\infty} L_{1/i} = \{ w \in \Sigma^* \mid \exists v \in \Sigma^* \; wv \in L \wedge \exists n \in \mathbb{N} \; \frac{|w|}{|wv|} = \frac{1}{n} \}$

prefiks

jakkolwiek da się coś dopisać, jak w zad 23

$\in L_{1/k}$

jak w zad 22

$\mathcal{A} = \langle \Sigma, Q, q_0, \delta, F \rangle$ - automat dla L.

$\mathcal{A}' = \langle \Sigma, Q = \{ Q, S_0, S_1, \dots, S_{n-1} \}, q_0 = \{ q_0, \{q_0\}, \{q_1\}, \dots \{q_{n-1}\} \}, \delta', F' \rangle$

$\delta' \to$ dla q robi krok z δ, dla wszystkich S robi ~~zbiór~~ zbiór stanów taki, że $\exists a \in \Sigma$ że się da dojść jakoś

(osiągalne stany „jakoś") $\to S_i = \{ q \mid \exists v \; \hat{\delta}(q_i, v) = q \}$

$$\begin{array}{ccccc} q_0 & \{q_0\} & \{q_1\} & \dots & \{q_{n-1}\} \\ \downarrow w & \downarrow \Sigma^{|w|} & \downarrow \Sigma^{|w|} & & \downarrow \Sigma^{|w|} \\ q_x & S_0 & S_1 & \dots & S_{n-1} \end{array}$$

$F' \to$ takie stany, że istnieje ciąg $i_1, i_2, \dots, i_l$

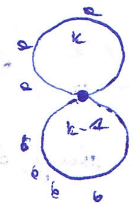
$q_0 \xrightarrow{w} q_x \xrightarrow{S_x \; \Sigma^{|w|}} S_y \to \dots \to S_t$

i $S_t \cap F \neq \emptyset$.

76.

$L^k = \{a^k + b^{k-1}\}^*$

automat rozstrzygający ma ck stanów



1. Wszystkie $n > k^2$ $\exists w \in L^k$ $|w| = n$

np $n = ck - d$

$(b^{k-1})^d (a^k)^{\frac{ck-d-kd+d}{k}} = c-d$

2. $\forall n \geq k^2$ $b^n \in L^k_{1/2}$

$n = ck - d$

$b^n b^{c-d+k-1} = b^{(c+k)k-1}$

Jeśli istnieje j t. że $b^j \notin L^k_{1/2}$ to istnieje największe takie j. (bo od $2k^2$ wszystkie nie należą)

3. $b^{\left\lfloor \frac{(k-1)(k-1)-k}{2} \right\rfloor} \notin$

bo jeśli dla $|w| = (k-1)(k-1) - k$, to $w \notin L^k$

Najkrótsze w t. że $|w| = n$ i $n \equiv 1 \mod k$ to: $(b^{(k-1)})^{(k-1)}$

4. Jeśli istnieje b^{m_k} ← największe $\notin L^k_{1/2}$, to dla m,

$l < m_k$ → z umowy być różne stany, bo inaczej możemy to pompować.

Pokazujemy że to słowo jest kwadratowe.

CEL: znaleźć $b^{m_k} \notin L^k_{1/2}$ takie że m_k jest $O(k^2)$

a w tym celu wystarczy znaleźć l_k należące $O(k)^2$

$l_k \equiv 0 \mod 2$ t. że $\neg\exists w \in L^k$ $|w| = l_k$

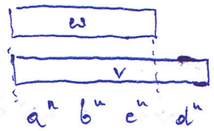
Wzory: $k^2 - 4k + 1 = (k-1)(k-1) - 2k$ dla k nieparzystych

Argument: $k^2 - 4k + 1 \equiv 1 \mod k$

$\min_n \{\exists w \in L^k \ |w| = n \wedge n \equiv 1 \mod k\} = (k-1)^2$.

77 $L_{3/4}$ nie musi być CFL. (Podobnie jak zad 56 dla $L/2$)

$L_{\frac{3}{4}} = \{ w \mid \exists v \in L \ \frac{|w|}{|v|} = \frac{3}{4}$ i w jest prefiksem $v \}$



Niech $L = \{ a^n b^m c^m d^n \}$

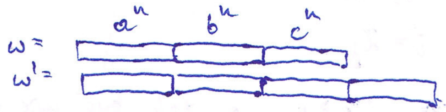
$L \in CFL \to$ gramatyka:

$S \to aSd \mid T$

$T \to bTc \mid \varepsilon$

Zał. nie wprost, że $L_{\frac{3}{4}} \in CFL$. Zachodzi więc lemat o pompowaniu.

n - stała z lematu. Weźmy słowo $a^n b^n c^n$, $\in L_{\frac{3}{4}}$, bo:



$w' \in L$, w jest prefiksem w' i ma odpowiednią długość.

Pokażemy, że ~~...~~ nie istnieje podział sztyx.

$|zty| \leq n$ zatem mamy przypadki: (zy zawiera max dwie różne litery)

1° zy zawiera TYLKO b lub TYLKO c

$sz^0ty^0x = a^n b^{n-i} c^n$ lub $a^n b^n c^{n-i}$

↳ mniej b niż c ↯

↳ mniej c niż b, więc muszą być dalej

$\underbrace{a^n b^n c^{n-i}}_{3n-i} \underbrace{c^i d^n}_{n+i}$ (długości: $3n-i$, $n+i$)

$(3n-i) \cdot \frac{4}{3} \neq 4n$

czyli $\notin L_{\frac{3}{4}}$

2° zy zawiera TYLKO litery a.

$sz^0ty^0x = a^{n-i} b^n c^n$

↳ $\underbrace{a^{n-i} b^n c^n}_{3n-i} \underbrace{d^{n-i}}_{n-i}$ $(3n-i) \cdot \frac{4}{3} \neq 4n-2i$

3° zy zawiera: $|zy|_a = i$, $|zy|_b = j$.

$sz^0ty^0x = a^{n-i} b^{n-j} c^n$

↑↑ nie tyle samo literek b i c ↯

4° zy zawiera: $|zy|_b = i$, $|zy|_c = j$

$sz^0ty^0x = a^n b^{n-i} c^{n-j}$, oczywiście $j > i$ bo inaczej będzie więcej c niż b

↳ $\underbrace{a^n b^{n-i} c^{n-j}}_{3n-i-j} \underbrace{c^{j-i} d^n}_{n+j-i}$, $c^{n-j} c^{j-i} = c^{n-i}$

$(3n-i-j) \cdot \frac{4}{3} = 4n-2i$

prawda dla $i = 2j$

Ale to oznacza, że $i > j$, a tak być nie może!

Jeśli $i = j = 0$ to $zy = \varepsilon$

↯

INNE ROZW:

$L = a^n b^m c^m dddd a^n \in CFL$ ✓

$L_{3/4}$ jest CFL? zał. że tak

$L' = L_{3/4} \cap a^* b^* c^* ddd = \{ a^n b^m c^m ddd \mid n+2m+3 = \frac{3}{4}(2n+2m+4) \}$

→ czyli $n = m$

$= \{ a^n b^n c^n ddd \}$

↳ nie jest CFL ↯

Zad 78

a) Dają nam transducer Moore'a T, stwórzmy Mealy'ego T'

t.że $\forall w \; f_T(w) = f_{T'}(w)$

$$T = \langle \Sigma_0, \Sigma_1, Q, q_0, \delta, \sigma_T \rangle$$

$$T' = \langle \Sigma_0, \Sigma_1, Q, q_0, \delta, \sigma_{T'} \rangle$$

$$\sigma_{T'}(q,a) \overset{def}{=} \sigma_T(\delta(q,a))$$

Dowód, że są równoważne: indukcja po $|w|$

1° $w = \varepsilon$

$f_T(\varepsilon) = \varepsilon = f_T(\varepsilon)$ ✓

2° $w = w'a$

$$f_T(w'a) \overset{def}{=} [f_T(w')][\sigma_T(\hat{\delta}(q_0, w'a))]$$

$$\overset{\text{zał. ind.}}{=} [f_{T'}(w')][\sigma_T(\hat{\delta}(q_0, w'a))]$$

$$\overset{def\ \sigma_{T'}}{=} [f_{T'}(w')][\sigma_{T'}(\hat{\delta}(q_0, w'), a)] \overset{def}{=} f_{T'}(w'a)$$

✓

b) Dają nam transducer Mealy'ego T', zbudujemy $\tilde{T}$ Moore'a.

$$T' = \langle \Sigma_0, \Sigma_1, Q', q_0', \delta', \sigma_{T'} \rangle$$

$$\tilde{T} = \langle \Sigma_0, \Sigma_1, \tilde{Q}, \tilde{q_0}, \tilde{\delta}, \tilde{\sigma}_{\tilde{T}} \rangle$$

stan z PRZESZŁOŚCI

przeczytana literka

UWAGA: Chcielibyśmy napisać $\sigma_{\tilde{T}}(\delta(q,a)) = \sigma_{T'}(q,a)$

Ale nie możemy tak rozwiązać definicji!

Będziemy rozważać stany $\tilde{Q}$ jako $\langle q, a \rangle$

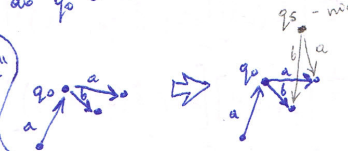
przyszedłem tu ze stanu q — czytając „a"

$$\tilde{Q} = (Q' \times \Sigma_0) \uplus \{q_s\}$$

suma rozłączna

nowy symbol dla stanu początkowego, bo w nim zaczynamy, nie ma przeszłości. A do q_0 z automatu możemy jakoś znów wrócić.

q_s - nie da się tu wrócić



$$\tilde{q_0} = q_s$$

przyszedłem tu z q_0, czytając „a"

$$\tilde{\delta}(q_s, a) = \langle q_0, a \rangle$$

$$\tilde{\delta}(\langle q,a \rangle, b) = \langle \underbrace{\delta'(q,a)}_{\text{przyszedłem z tego stanu}}, b \rangle$$

czytając „b"

$$\tilde{\sigma}_{\tilde{T}}(q_s) = \varepsilon$$

$$\tilde{\sigma}_{\tilde{T}}(\langle q,a \rangle) = \sigma_{T'}(q,a)$$

Dowód indukcyjny podobny do a).

$$f_{T'}(w'a) = [f_{T'}(w')][\sigma'(\delta'(q_0, w'), a)]$$

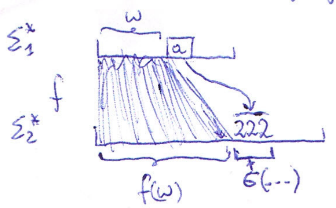
$$\overset{\text{zał. ind.}}{=} [f_{\tilde{T}}(w')][\ —\|—\ ]$$

$$\overset{def\ \tilde{\delta}}{=} [f_{\tilde{T}}(w')][\tilde{\sigma}(\langle \underbrace{\delta'(q_0, w')}, a \rangle)] = f_{\tilde{T}}(w'a)$$

da się z q_0 więc da się z q_s dla $\tilde{\delta}$

JFIZO zad 79

TRANSDUCER – tłumaczy języki, przelatuje po w_A i wypluwa w_B



Mealy: stan × literka Σ_1 ⟶ słowo Σ_2

$A \le_{rep} B \wedge B$ regularny $\Rightarrow A$ regularny ?

B regularny, czyli istnieje DFA $\mathcal{B}(\Sigma_1, Q_B, q_{B0}, F_B, \delta_B)$

$A \le_{rep} B$ czyli istnieje transducer Mealy'ego $T(\Sigma, \Sigma_1, Q_T, q_{T0}, \delta_T, \sigma)$

Skonstruujemy DFA $\mathcal{A}$, który rozpoznaje A.

$$Q_A = Q_T \times Q_B$$

$$q_{A0} = \langle q_{T0}, q_{B0} \rangle$$

$$F_A = \{\langle q_T, q_B \rangle \mid q_T \in Q_T \wedge q_B \in F_B\}$$ ← B akceptuje słowo po konwersji

$$\delta_A(\underbrace{\langle q_T, q_B \rangle}_{q_A}, a) = \langle \delta_T(q_T, a), \underbrace{\hat{\delta}_B(q_B, \underbrace{\sigma(q_T, a)}_{\text{transducer czyta i produkuje słowo}})}_{\text{automat } B \text{ idzie po słowie i zmienia stan}} \rangle$$

$$\hat{\delta}_A(\underbrace{\langle q_T, q_B \rangle}_{q_A}, w) = \langle \underbrace{\hat{\delta}_T(q_T, w)}_{\text{transducer zmienia stan}}, \underbrace{\hat{\delta}_B(q_B, \underbrace{f_T(w)}_{\text{transducer produkuje słowo}})}_{\text{automat } B \text{ zmienia stan}} \rangle$$

$w \xrightarrow{\text{TRANSDUCER}} v$ DFA $A(w)$ akceptuje $\Leftrightarrow$ DFA $B(v)$ akceptuje

$v = f_T(w)$ WIEMY ŻE $w \in A \Leftrightarrow f_T(w) \in B$

Dowód, że $A = L_{\mathcal{A}}$

$w \in A \Rightarrow \mathcal{A}$ akceptuje w

$$w \in A \Rightarrow f_T(w) \in B \Rightarrow \mathcal{B} \text{ akceptuje } f_T(w) \Rightarrow \hat{\delta}_B(q_{B0}, f_T(w)) \in F_B$$

$$\hat{\delta}_A(\langle q_{T0}, q_{B0} \rangle, w) = \langle \underbrace{\hat{\delta}_T(q_{T0}, w)}_{\text{jakiś stan } T}, \underbrace{\hat{\delta}_B(q_{B0}, f_T(w))}_{\text{akceptujący } B} \rangle \in F_A$$ ✓

$w \notin A \Rightarrow \mathcal{A}$ nie akceptuje w

$$w \notin A \Rightarrow f_T(w) \notin B \Rightarrow \hat{\delta}_B(q_{B0}, f_T(w)) \notin F_B \Rightarrow \text{analogicznie} \notin F_A$$