

Rozwiązania wybranych zadań z egzaminów z Języków Formalnych i Złożoności Obliczeniowej

2006-2021

Bartłomiej Grochowski

Żeby ułatwić zdanie tego przedmiotu, zebrałem tutaj swoje rozwiązania zadań z egzaminów z poprzednich lat. Część z nich jest moja własna, część to pomysły zasugerowane na repetytorium i elegancko zapisane.

Spis treści:

1. Części pierwsze - str. 1
2. Części drugie - str. 19
2. Części trzecie - str. 55

Uniwersytet Wrocławski
18 czerwca 2023

$$L = \{ w \in \{a,b,c\}^* \mid \text{zdanie z } [w_k, w_l, w_m] \text{ ma co najmniej dwa równe parzyste}\}$$

L nie jest CFL. Zat, że L jest CFL, wtedy

$$L' = L \cap \underbrace{(aa)^*}_{2} \underbrace{(bbb)^*}_{3} \underbrace{(ccc)^*}_{5} \text{ jest CFL.}$$

FAT: jeśli zdanie ma co najmniej dwa równe parzyste, to musi być ich k. NWW(2,3,5) równe $L' = \{a^n b^n c^n \mid n \bmod 30 = 0\}$

Lemat: L' nie jest CFL, czyli $\nexists L$ taki że L' nie jest CFL.

Dowód lematu: Zat, że L' jest CFL, zauważmy iż p, p-stała złożona
niech k to największa liczba $\geq p$ t.ż. $k \bmod 30 = 0$.

$$a^k b^k c^k \in L', \text{ zty } \leq p \in k \text{ czyli } \frac{k}{p} \text{ jest całkowite i ma dwa różne reszty.}$$

→ zakładając wybranie się x , to brzmiemy np. poważnie
tylko jeden lub dwa bloki, a trzeci nie
 $six \notin L'$ → L' nie jest CFL.

a) NIE, niech $L = L_0^*$

żeś, iż $2L$ jest regularny, zauważmy iż p - stała złożona.

$$\text{Wtedy istnieje } \omega = \underbrace{0^p 1^p}_w \in L \text{ do } \omega \in L, |w| > p$$

zatem powinien istnieć położenie xyz. $|xy| < p$ zatem

oba środkowe położenia y to same zero. $y \neq \varepsilon$.

$$xyz = \underbrace{0^{p-i}}_w \underbrace{1^p}_1 \notin 2L \text{ do } \omega = 0^p 1^p \notin L$$

b) TAK, M-automat dla L . Zbudujmy NFA M' dla L_2 .

$$Q' = Q \cup \{q_{ij} \mid q_i \in Q \wedge q_j \in Q\}$$

stały przejście

$$q'_0 = q_0$$

$$F' = F$$

$$S' \text{ tiene sou } S:$$

$$- jeśli w S było \delta(q_i, a) = q_f \text{ to:}$$

$$S'(q_i, 0, q_{if}) \wedge S'(q_i, 1, q_{if}) \wedge S'(q_{if}, 0, q_f) \wedge S'(q_{if}, 1, q_f)$$

→ M' robi dwa kroki, interesują nas TPLKO dążyąc.

Szukając cięwadła: M' akceptuje wV ⇒ wtedy jego każde akceptująceq:

$q_0 \rightarrow q_{0x} \rightarrow q_x \rightarrow \dots \rightarrow q_{xcc}$, popatrując na co drugi stan $q_0 \rightarrow q_{0x} \rightarrow \dots \rightarrow q_{xcc}$
i to będzie cięwadło akceptujące dla JAKIEGOŚ zdania z L (2x kroksze)

NIE. Zostanie L jest CFL, natomiast $L^1 = L \cap \overbrace{(abc)^*(cba)^*}^{rep} b^*$
 też jest CFL. Powinien zachodzić lop, p-stała z lematu.

$$w = \underbrace{(abc)^p}_{6p} \underbrace{(cba)^p}_{6p-1} \underbrace{b^{6p-1}}_{1} \in L^1 \text{ bo zaczyne się od paryego pełno-} \\ \text{tnikiego bloku i kończy na pełnym bloku.}$$

Jakie może być sztyx?

Iztyk p zatem zy załącza max o dwa różne bloki.

1° zy całe w bloku III \rightarrow taki rozpompowany, że pełnowłasem nie
 będzie również niż pełnowłasem.

2° zy całe w bloku I \rightarrow rozpompowany tak, że nie będzie pełnowłasem

3° zy całe w bloku II \rightarrow taki rozpompowany, że pełnowłasem nie będzie
 również niż pełnowłasem

FAKT: aby zawsze y nie mogło być w kawieku w dwóch
 różnych blokach bo jeje rozpompowany, to aby nie było dwóch pełnowłasów

$$4^{\circ} z = (abc)^i, y = (cba)^j$$

\hookrightarrow sztyx pełnowłasem będzie kiedyś od pełnego dupleksu

$$5^{\circ} z = (cba)^i, y = b^j$$

\hookrightarrow jeśli $i \neq 0$ to jak odpompowany (sztyx) to nie będzie pełnowłasem.

\hookrightarrow jeśli $i = 0$ to innego przypadku 1°

Indeks \equiv ilość stanów minimalnego DFA.

$$\text{Indeks} = 2^3 = 8.$$

Dowód zat. że $\exists \text{ DFA. } \exists |Q| < 2^3$.

Różnych stanów dopasujących jest 2^3 , zatem istnieją dwie RÓŻNE słowa w, w' t.c. $\hat{\delta}(q_0, w) = \hat{\delta}(q_0, w')$.

w	x	0	y	q_0
w'	x	1	y	

$x = x'$, bierny po pierwszą literę
na której są różni

Rozpatrujemy dwa przypadki:

- 1° y ma parzyste wielkość
 y' ma parzyste wielkość

wtedy: $x0y$
 $x1y'$ mają równą parzystą jedynkę
a ten sam stan

- 2° y ma parzyste wielkość
 y' ma nieparzystą wielkość

wtedy: $x0y0^{k+1}$
 $x1y'0^{k+2}$
 $q_0 \xrightarrow{x} q_1 \xrightarrow{y} q_2 \xrightarrow{0} \dots \xrightarrow{0} q_{k+1} \xrightarrow{y'} q_{k+2}$
 $|M| = 9$

$3^0 : 4^0$ są analogiczne jak $1^0 : 2^0$.

DFA minimalny mający dokładnie 2^3 stanów:

Q - wszystkie możliwe kombinacje bitów 2^3 (2^3)

F - te stanów, które parzyste wielkość.

S - shift w lewo o jeden i dodać cyfrę cyfę

$L = \{w \mid w \text{ jest zapsem binarnym } 3^{2^n}\}$ nie jest CFL.

Zad. i.e. L jest CFL, wtedy zachodzi teoremt o pozwoleniu.

p -strefa z lematu. DEYGOSĆ zapisu liczby n to $\lceil \log_2 n \rceil$.

Weźmy sobie dowolne duże k , t.i.e:

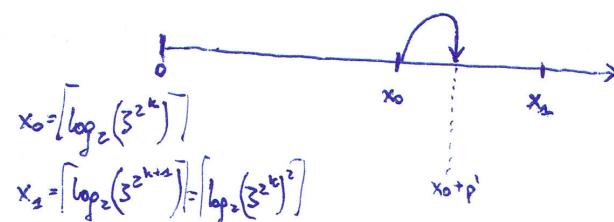
$$(1) \quad \lceil \log_2(3^{2^k}) \rceil \gg p$$

$$(2) \quad \lceil \log_2(3^{2^{k+1}}) \rceil = \lceil \log_2(3^{2^k})^2 \rceil \gg p$$

Niech w będzie zapsem liczby 3^{2^k} , $w \in L$ i $|w| > p$ bo (1).

Dla dowolnego podzielnego słyxa, $|sztyx| = |sztyx| + p'$

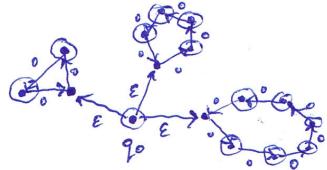
$$w' = sztyx \notin L \quad !!! \quad (2) \quad 1 \leq p' \leq p$$



$x_0, x_1 \rightarrow$ dłuższy zapsem binarnego kolejnych liczb z jacyka.

$w' \notin L$ spowodowane! teoremt nie zachodzi, oyleg nie jest CFL.

JF120 EGZ 2020 I zad 1



$$\text{Länge Schiene} = 1 + 3 + 5 + 7 = 16$$

$$\Sigma = \{0\}$$

Haben wir rechteckige Zylinder $\{w \mid |w| \text{ ist durch } 3 \text{ teilbar}\}$

$$1) \forall w \quad |w| \leq 100 \Rightarrow w \in L$$

1. ϵ - Zeile, da q_0 jetzt akzeptierend

2. 0^k , $1 \leq k \leq 100$ - die Zeile k ist leer, kleine w sind nicht teilbar
purer $3, 5, 7$ untersetzt mit 100,
 w ist zuerst bei der ersten Prüfung akzeptiert.

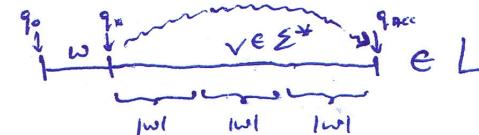
$$2) \exists r \quad M > 100 \wedge r \in L \quad (\text{also } L \neq \Sigma^*)$$

Zeile, wie $0^{100}, 0^{200}, 0^{300}, \dots$

Die Zeile s_0 ist abweig prüfbar akzeptiert, da sie
alle 3er Zeilen sind purer $3, 5, 7$.

JF120 EGZ 2020 II zad 2

$$L' = \{ w \mid \exists v \quad |v|=k \cdot |w| \wedge v \in L \}$$



Idee:

$$q_0 \{q_0\} \{q_1\} \dots \{q_m\}$$

$$w \downarrow \Sigma^{\omega} \downarrow \Sigma^{\omega} \downarrow \Sigma^{|w|}$$

$$q_x \quad s_0 \quad s_1 \quad \dots \quad s_{n-1}$$

$$q_0 \xrightarrow{w} q_x \xrightarrow{\Sigma^{\omega}} s_x \xrightarrow{\Sigma^{\omega}} s_y \xrightarrow{\Sigma^{\omega}} s_k \xrightarrow{\Sigma^{\omega}} q_{acc}$$

$$S'_0 = \langle \Sigma, Q', q'_0, S'_1, F' \rangle$$

$$Q' = Q \times P(Q)^{|Q|}$$

$$q'_0 = \langle q_0, \{q_0\}, \{q_1\}, \dots, \{q_{m-1}\} \rangle$$

$$S'_1(\langle q_0, s_0, s_1, \dots, s_{n-1} \rangle; x) = \langle \delta(q_0, x), \{q \in Q \mid \exists q_n \in s_0 \quad \delta(q_0, a) = q_n \wedge \delta(q_n, b) = q_0\}, \dots \rangle$$

String q_0 ist akzeptierend für
denen p von q_0 aus
durch x erreichbar

$$F' = \{q_x, s_0, s_1, \dots, s_{n-1} \mid \underbrace{\exists k_0, k_1, k_2, \dots, k_l}_{\text{wegen}}, \underbrace{l < n}_{\text{siehe weiter unten}}$$

wegen siehe weiter unten

$$q_x = q_{k_0}, q_{k_l} \in F, \forall i \quad q_{k_{i+1}} \in s_{k_i} \}$$

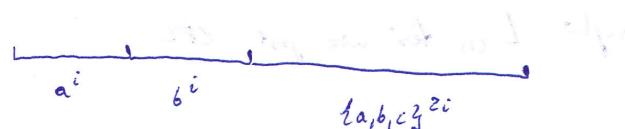
\rightarrow WERKAPPLICO DUZO STANOWI DFA ✓.

NIE wynika. Wsomy $L = \{a^n b^n \mid n \in \mathbb{N}\}$

L jest CFL, bo trywialne konstrukcje go gwarantują $S \rightarrow^* \Sigma^*$ asb

Wtedy $L_{(2)} = \{wv \in \{a,b,c\}^* \mid |w|=|v| \wedge w \in L\}$

czyli perowane potowe słowa $a^k b^k$ a potem jest całkowite.



Zatem $L_{(2)}$ jest berkantektoru \rightarrow zakończenie kanci o pomponami.

Niech k będzie stały i konkretny. Wsomy słowa $a^k b^k (a+b+c)^{2k}$.

aaaaaa 666666] bacababab
t potowe

1. jeśli zbyt większe w przodzie potowor, to skończyc słowo cyli potowor w lewo. Wtedy perowane potowe większe $a^k b^i c^{2k-i} \rightarrow$ E.

2. jeśli zbyt większe w przedku b^k do końca zero.

3. jeśli zbyt większe w przedku a^k to przypisujemy tek.

czyli potowor przypisuje się na końcu a:

aaa ooo] 666666

Zauważmy, że $L_{(2)}$ nie jest CFL

INNE ROZWIAZANIE:

$L = a^n b^n \#$, jest CFL.

Zatem $L_{(2)}$ jest CFL oraz $a^n b^n \# \sum^{2n+1}$ jest CFL.

$L' = L_{(2)} \cap a^* b^* \# \# c^*$ też musi być CFL bo
 $CFL \cap reg = CFL$.

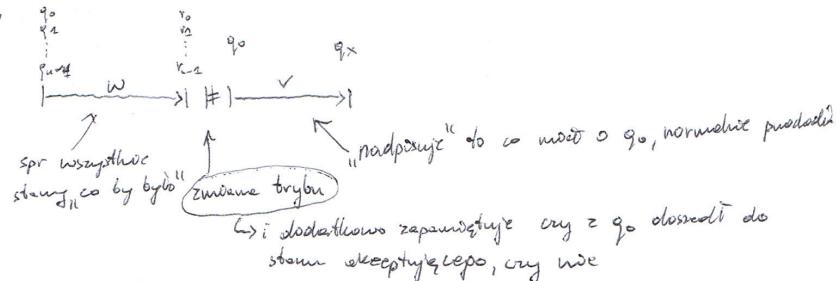
$L' = \{a^n b^n \# \# c^{2n}\}$ nie jest CFL! sprzeczność!

czyli $L_{(2)}$ też nie jest CFL.

$L_{\#}$ regulowany, n stanów. $L_{\#}$ da się rozpoznać DFA o n^k stanach

$$L_{\#} = \{ w \# v \mid v w \in L \}$$

Idea:



$$\text{DFA } \mathcal{A}^1 = \langle \Sigma, Q^1, q_0^1, \delta^1, F^1 \rangle$$

$$Q^1 = Q^{1|1} \times \{0,1\} \times \{0,1\} \leftarrow 4^n \text{ stanów}$$

\uparrow dla $a \in \{0,1\}$ \uparrow dla $r_0 \in \{0,1\}$

poolej a po $\#$ \uparrow dla $r_0 = q_0$ mimoższość do stanu akceptującego po przekształceniu w .

$$q_0^1 = \langle q_0, q_1, \dots, q_{n-1}, 0, 0 \rangle$$

$$\delta^1(\langle r_0, r_1, \dots, r_{n-1}, 0, 0 \rangle; a) = \langle \delta(r_0, a), \delta(r_1, a), \dots, 0, 0 \rangle$$

$$\delta^1(\langle r_0, r_1, \dots, r_{n-1}, 0, 0 \rangle; \#) = \langle q_0, r_1, r_2, \dots, r_{n-1}, 1, X \rangle$$

gdzie $X=1$ jeśli $r_0 \in F$, $X=0$ wpp.

$$\delta^1(\langle r_0, r_1, \dots, r_{n-1}, 1, X \rangle; a) = \langle \delta(r_0), r_1, \dots, r_{n-1}, 1, X \rangle$$

tylko tu krok

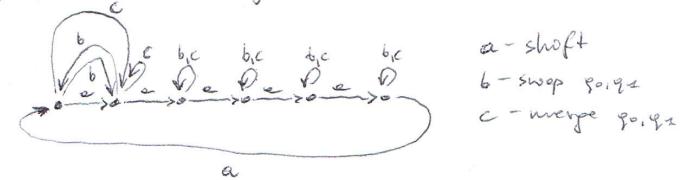
$$F^1 = \{ \langle r_0, r_1, \dots, r_{n-1}, 1, X \rangle \mid (\exists q_A \ r_0 = q_A \wedge r_A \in F \wedge A \neq 0) \} \parallel q_0 \xrightarrow{\vee} q_n \xrightarrow{\omega} r_n$$

lub $r_0 = q_0 \wedge X = 1 \}$ $\parallel q_0 \xrightarrow{\vee} q_0 \xrightarrow{\omega} q_{acc}$

nadpuszcza inf.
ale nie popiera $w X$.

Na zajęciach rozważaliśmy automat o n stanach:

$$A_n:$$



$$\text{Nech } L_n = \{ w \in \{a,b,c\}^* \mid \delta_{A_n}(q_0, w) = q_0 \}$$

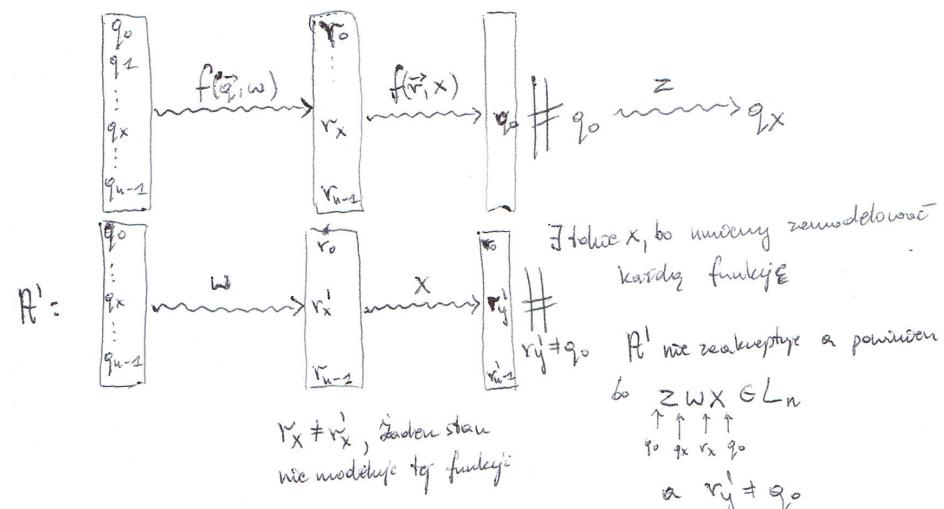
Ustwadźmy, że istnieje słowa, które zachowują się jak funkcja $Q \rightarrow Q$.

Np: $\langle 1 \ 2 \ 3 \ 4 \ 5 \ 6 \rangle$ wszystkie te słowa funkcji jest n^n
 $\langle 2 \ 2 \ 5 \ 3 \ 1 \ 6 \rangle$ funkcje mówiąc we co przejdą konkretnie stan po wczytaniu słowa

Want, aby $L_{\#}$ da się rozpoznać automatem \mathcal{A}^1 o mniej niż n^k stanach.

Wtedy istnieje funkcja, której inde odpowiadają zadanym stanom (zas. swift)

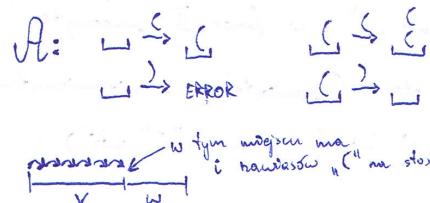
Zobaczymy jak zachowuje się \mathcal{A}^1 jako funkcja.



L - język słów złożonych nawiązów, $L_{\#} = \{w \# v : vw \in L\}$

L[#] jest językem berkantekstowym.

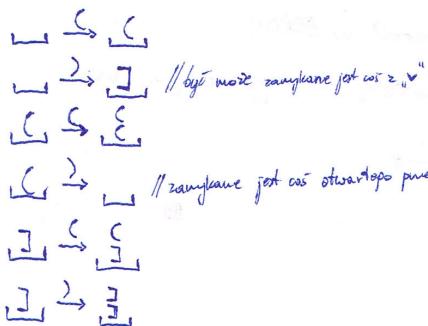
Pokazemy, autonem ze stosem dla L i L^* .



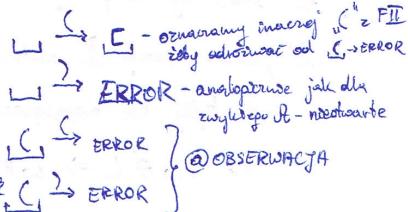
A_# ma dwa fary, i dwa stany, # znowu ten stan



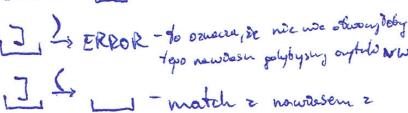
FAZIA I:



FAZAT II



$\Sigma \hookrightarrow \Sigma$



OBSERWACJA: Po faza I jest na stole jest jedwabnik ^{poprzedniej fazy} " to jest źle, bo to znaczy, iż nie został zakupiony nowy, a w oryginalnym stanie wtedy by po prostu nie było, czyli nawiązanie byłoby źle.

$w \# v \in L_{\#} \Leftrightarrow R_{\#} \text{ akzeptiert } w \# v$

\Leftrightarrow $w \# v \in L_{\#} \Rightarrow nw \in L \Rightarrow$ If po wykazane v ma na stanie k symboli

" C ", po wątpliwości w moim pierwotnym stosie, był zauważalny te mówiące
 (miał k. zapisywanym) \Rightarrow $\text{f}_{\#}$ po wątpliwości, w' w fozie I
 będzie miał na stosie k symboli " J ", w fozie II pocięte
 k literki " C " będą wyciągane z stosu. (pamiętaj o niewidzialnych
 \geq f., t.ż. zawsze ważona ważej lub tyle sierw " " C " niż " J ")

$\Rightarrow \mathcal{R}_\#$ accepts $w^\# v$.

1

R_# nie akceptuje w#v

1^o steps by neperusy \Rightarrow zda jest nowasaw $\Rightarrow vw \notin L \Rightarrow w\#v \notin L$

2º podras ferg II automet w result w ERROR

\Rightarrow by the above we see w is orthogonal to

$$\Rightarrow v \omega \notin L \Rightarrow \omega \# v \notin L \#$$

$L_{w \neq v}$ jest CFL ale nie jest DCFL.

Ma to jest DCFL, wtedy jego dopełnienie też jest językiem kontekstowym.

$$\overline{L}_{w \neq v} = \{xx \mid x \in \{0,1\}^*\}$$

Jest to kontekstowy, to zatem musi być o pomiaru.

p-stała = karta. Wtedy słowo $w = 0^p 1 0^p 1 0^p 1$

$w \in \overline{L}_{w \neq v}$ i $1w/3p$, zatem powinno istnieć podział słyxa.

Jakie może być słyxo?

→ może mieć maksymalnie jedynkę jedynkę.

1° $|zy|_1 = 1$, wtedy słyxo ma trzy jedynki, więc nie da się podzielić na dwa same podcięcia

$$2^{\circ} |zy|_1 = 0 \wedge |zy|_0 = i, i \leq p$$

Wtedy słyxo, co się stanie ze ŚRODKIEM słowa?

1° zg bylo w pierwszej połówce:

środek musi być w przedku, ergo

$$w^l = \underbrace{0 \dots 0 1 0 \dots 0 1}_{x_1} \quad \underbrace{0 1 0 \dots 0 1}_{x_2}$$

x_1 który się zerem

x_2 który się jedynką

2° zg bylo w drugiej połówce

→ środek może w lewo

$$w = x_1 x_2$$

$\uparrow \quad \uparrow$
 x_2 ma trzy jedynki
 x_1 ma jedną jedynkę

3°

$$0 \dots 0 1 0 \dots 0 1 0 \dots 0 1$$

$\overbrace{\quad \quad \quad}^{0^i} \quad \overbrace{\quad \quad \quad}^{0^j} \quad \overbrace{\quad \quad \quad}^{0^k}$

$i < j$ lub $i > j \rightarrow$ analogicznie jak w 1° i 2°

$$i=j \text{ wtedy } słyxo = \underbrace{0^p 1 0^p}_{x_1} \underbrace{i 0^p 1 0^p}_{x_2} \quad i \neq 0, \text{ bo } zy \neq \epsilon$$

Składa się z dwóch części, bo x_1 ma pierw przede wszystkim jedynkę a x_2 ma pierw przede wszystkim jedynkę więc $w \notin \overline{L}_{w \neq v}$

Ergo $\overline{L}_{w \neq v}$ nie jest CFL, sprzeciw!

INNE ROZWIĄZANIE:

Załóżmy, że się da, tzn istnieje DPDAT. W jego relacji precyza domena ilość symboli na stosie. Niech n_{\max} to NAJWIĘKSZA wartość symboli jaką domena na stosie.

Po przełączaniu stanów dającym m autamat może być w $1Q/(n_{\max} \cdot m) + 1$ różnych konfiguracjach co jest liniowe względem m.

Różnych słów jest 2^m , co jest wykładnicze.

Dla odpowiednego dającego m będą dwa słowa w, \overline{w} , iż autamat będzie w tej samej konfiguracji: $w \in L$ Po dokonaniu w do końca z nich

$w' \in \overline{L}$ będąc w tej samej, bo jest deterministyczny, i będąc się mylit.

A r.e., czy $\bigcup_{n \in A} \text{Dom}(\varphi_n)$ jest r.e.?

TRW. A jest r.e., więc istnieje program Ψ_A semi-vorstępujący A.

Ψ :

- wczytaj n
- for $k=0$ to ∞
 - i, j, k = bij(k)
 - adpal $\varphi_A(i)$ na j kroków
 - jeśli zwróciło 1, to: // to $i \in A$
 - adpal $\varphi_i(n)$ na k kroków // czy $n \in \text{Dom}(\varphi_i)$?
 - jeśli zwróciło to ret 1;

$$x \in \bigcup_{n \in A} \text{Dom}(\varphi_n) \Leftrightarrow \Psi(x) = 1$$

$x \in \bigcup_{n \in A} \text{Dom}(\varphi_n) \Rightarrow \exists i \in A. x \in \text{Dom}(\varphi_i) \Rightarrow \varphi_i(x)$ staje po k krokach
 \Rightarrow z definicji działania Ψ , kiedyś zwróci 1.

$x \notin \bigcup_{n \in A} \text{Dom}(\varphi_n) \Rightarrow$ nie istnieje $i \in A$ że $\varphi_i(x)$ staje po k krokach
 $\Rightarrow \Psi$ bieżące skończy w nieskończoności.

$$A = \{n \mid \varphi_n \text{ jest } "z \text{ grubią } 1-1"\}$$

$$\overline{K} \subseteq_{\text{rek}} \overline{A}$$

f:

- wczytaj n
- napisz program:
 - * wczytaj m
 - * adpal $\varphi_n(u)$ na m kroków
 - ↳ jeśli zwróciło to zwroć 0
 - ↳ else zwroć m
- zwróć nr TEGO programu.

$$n \in \overline{K} \Rightarrow \varphi_n(u)$$
 nie staje $\Rightarrow \varphi_{f(n)}$ działa jak identyfikator, jeśli jest 1-1 $\Rightarrow f(n) \in A$.

$$n \notin \overline{K} \Rightarrow \varphi_n(u)$$
 staje po $x \Rightarrow \forall j > x \varphi_{f(j)}(j) = 0$, niskowartość indeksu 0^{th} $\Rightarrow f(n) \notin A$.

$\overline{A} =$ istnieje jakieś kątka, it jest ∞ wiele razy w zbożu wartości funkcji.

$$\overline{K} \subseteq_{\text{rek}} \overline{A}$$

f:

- wczytaj n
- napisz program:
 - * wczytaj m
 - * adpal $\varphi_n(u)$ na m kroków
 - ↳ jeśli zwróciło to zwroć m
 - ↳ else zwroć 0.
- zwróć nr TEGO programu

$$n \in \overline{K} \Rightarrow \varphi_n(u)$$
 nie staje $\Rightarrow \varphi_{f(n)}$ staje zawsze 0 $\Rightarrow f(n) \in \overline{A}$

$$n \notin \overline{K} \Rightarrow \varphi_n(u)$$
 staje po $x' \Rightarrow$ do pewnego momentu $0'$ a potem 1-1 $\Rightarrow f(n) \notin \overline{A}$.

$\overline{K} \subseteq_{rel} A$

If we just r.e.

f: -czytaj n

-napisz program:

$$\begin{cases} * \text{czytaj } m \\ * \text{odpal } \varphi_n(u) \\ * \text{zwrć } 1 \end{cases}$$

- x = numer PIERWSZEGO programu

-napisz program:

$$\begin{cases} * \text{czytaj } m \\ * \text{zwrć } 1 \end{cases}$$

- y = numer DRUGIEGO programu

- zwrć $\langle x, y \rangle$ $n \in \overline{K} \Rightarrow \varphi_n(u)$ nie staje $\Rightarrow \varphi_x$ zawsze 1, φ_y zawsze 1 $\Rightarrow f(n) \in A$. $n \notin \overline{K} \Rightarrow \varphi_n(u)$ staje $\Rightarrow \varphi_x$ zawsze 1, φ_y zawsze 1 $\Rightarrow f(n) \notin A$. $L = \{n \mid \varphi_n \text{ staje dla jakaś nieparzystej } \vee \text{ nie staje alle jakaś nieparzystej}\}$ b) $\overline{K} \subseteq L$

f: -czytaj n

-napisz program:

$$\begin{cases} * \text{czytaj } m \\ * \text{if } m \% 2 = 1 \text{ return } 1 // \text{staje dla wszystkich nieparzystych} \\ * \text{odpal } \varphi_n(u) \\ * \text{zwrć } 1 \end{cases}$$

- zwrć nr TEGO programu

 $n \in K \Rightarrow \varphi_n(u)$ staje $\Rightarrow \varphi_{f(n)}$ staje dla parzystych $\Rightarrow f(n) \in L$ $n \notin K \Rightarrow$ dla wszystkich nieparzystych $\varphi_{f(n)}$ staje a dla wszystkich parzystych się zatrzymuje.
 $\Rightarrow f(n) \notin L$ a) $\overline{K} \subseteq_{rel} L$

f: -czytaj n

-napisz program:

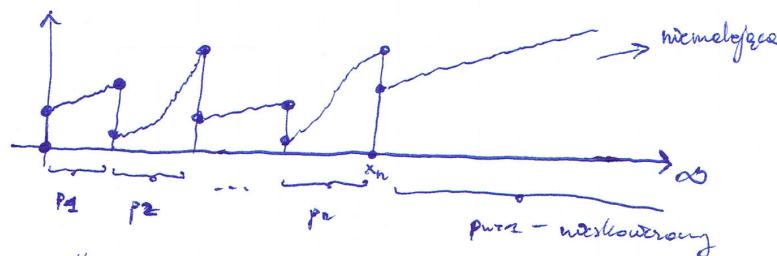
$$\begin{cases} * \text{czytaj } m \\ * \text{odpal } \varphi_n(u) \\ * \text{if } m \% 2 = 1 \text{ ret } 1 // \text{dla } \overline{L} \text{ symetryczne} \\ * \text{else zapisz } \varphi_{f(n)} \\ - \text{zwrć numer TEGO programu} \end{cases}$$
 $n \in \overline{K} \Rightarrow \varphi_n(u)$ nie staje $\Rightarrow \forall x \varphi_{f(n)}(x) = 1 \Rightarrow$ dla nieparzystych 1 $\Rightarrow f(n) \in L$ $n \notin \overline{K} \Rightarrow \varphi_n(u)$ staje \Rightarrow dla nieparzystych zawsze staje
dla parzystych zawsze się zatrzymuje $\Rightarrow f(n) \notin L$.

$f: N \rightarrow N$ with rule, skojarzenie wiele wartości dla $f(n) > f(m)$

$f(N)$ rekurencyjny? TAK

f całk. rule \rightarrow istnieje MT, która ją oblicza (lub program wAlg).

Maszyny zliczają n punktów, że funkcja tam mniej więcej i podobnie na $n+1$ przedziałach



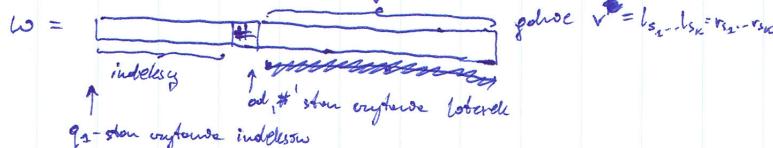
We wszystkich przedziałach $p_1 \dots p_n$ mamy skojarzenie wiele punktów - sprawdzamy, iterując się po nich, czy $f(x) = y$, gdzie pytamy o to, czy $y \in f(N)$.

Jakoże mamy wiele punktów, to ~~możemy~~ f ma wiele punktów. Punkt (x_n, y_n) jest niewielkością. Czyli analogicznie jak w zadaniu mamy dwie rózne.

JFIZO EGZ 2008 II zad 4

NIE istnieje taki algorytm. Gdyby istniał, to rozwiązywał problem PCP.

$\langle L_1, r_1, L_2, r_2, \dots, q_k, v_k \rangle$ Automaty akceptujące same tylko taki:



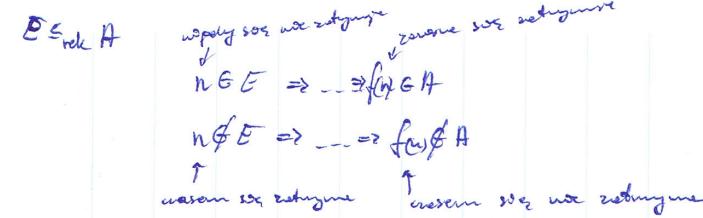
- PDA 1:
 - * jak wskazuje si to daje się na stosie l_i
 - * od # przejdź w stan porządkowy do stanu
 - * jeśli literka s_i nie zgodna, do stanu odrzucającej.
- PDA 2: taki sam tyle r

$L_{\text{PDA1}} \cap L_{\text{PDA2}} \neq \emptyset \Leftrightarrow$ istnieje taki, który jest normowaniem PCP.

(\Rightarrow) Istnieje taki automat akceptujący \Rightarrow istnieje taki, który jest sumą postaci $\langle \text{automat}, \#, \Sigma^* \rangle$, co wynika z konstrukcji automatów i modeli nieuporządkowanych (rozważałyśmy z konstrukcją), więc w chwili hasha' cui jest na stosach PDA1 i PDA2. Stan akceptujący może być taki, gdy stany są puste. Czyli musimy być taki sam literka na tym stosach, czyli mamy normowanie PCP.

(\Leftarrow) Istnieje taki $s \Rightarrow$ oba automaty akceptują $\boxed{s} \# \boxed{v}$, gdzie v to idem pozbawione z cogen s (akceptujące wybrane z konstrukcją)

JFIZO EGZ 2008 II zad 5



Redukcja:

- Wczytaj n
 - Nagospodarz sobie taki program:
 - * Wczytaj m
 - * $\langle a, b \rangle = \text{bez}(m)$
 - * odpet $y_n(a)$ na b sekund
 - * jeśli gie zatrzymał, do s_1 repeat
 - * jeśli s_2 nie zatrzymał to zwroc 1
 - zwroc numer tego programu jako $f(n)$.
- Wasem s_1 zatrzymuje i po pełnym czasie s_2 zatrzymuje wciśnij spustową klawiszową mostkową (parę) } y_K

$n \in E \Rightarrow y_n$ się napisywa nie zatrzymuje $\Rightarrow y_K$ zwroci 1 $\Rightarrow k = f(y_E) \in A$
 bo zwroci s_2 zatrzymuje.

$n \notin E \Rightarrow y_n$ się nie zatrzymał $\Rightarrow y_K$ wtedy się zatrzymał $\Rightarrow f(y_E) \notin A$

Intuicje: $E = \text{EMPT} = \{ \varphi \mid \forall n \forall t. \varphi(n) \text{ nie staje po } t \}$
 $\rightarrow \Pi_1$ czyli co-re

$A = \text{TOTAL} = \{ \varphi \mid \forall n \exists t. \varphi(n) \text{ staje po } t \}$
 $\rightarrow \Pi_2$ czyli trudny nie core.

$A \subseteq_{\text{re}} E$? NIE

Zat, że tak. Wtedy A też jest co-re, bo $\exists \varphi_A$ t.j. $x \notin A \Leftrightarrow \varphi_A(x) = 0$

φ_A : - wczytać n
- oblicz $k = f_{AE}(n)$
- zwroci $\varphi_E(k)$

Napiszemy program Ψ , który rozstygę K .

Ψ : - wczytać n
- napisać program:
 $\begin{cases} * \text{wczytaj } m \\ * \text{odpł } \varphi_E(n) \\ * \text{zwroc } 1 \end{cases}$ } φ_X albo jest A albo jest E (zależy od $\varphi_E(n)$)
 $\begin{cases} 1^{\circ} \text{jed } A \Rightarrow \text{nie jest } E \Rightarrow \varphi_E(x) = 0 \\ 2^{\circ} \text{jed } E \Rightarrow \text{nie jest } A \Rightarrow \varphi_A(x) = 0 \end{cases}$
- $x = \text{numer TEGO programu}$
- odpal rozstygę $\varphi_A(x) \vee \varphi_E(x)$
 \downarrow jeśli φ_A zwroci 0 to return 0
 \downarrow jeśli φ_E zwroci 0 to return 1

$n \in K \Rightarrow \varphi_n(n)$ staje $\Rightarrow \varphi_X$ zawsze staje $\Rightarrow x \in A \Rightarrow x \notin E \Rightarrow \varphi_E(x) = 0 \Rightarrow \Psi$ ret 1

$n \notin K \Rightarrow \varphi_n(n)$ nie staje $\Rightarrow \varphi_X$ nigdy nie staje $\Rightarrow x \in E \Rightarrow x \notin A \Rightarrow \varphi_A(x) = 0 \Rightarrow \Psi$ ret 1

INNE ROZWIAZANIE

Zat, że $A \subseteq_{\text{re}} E$.

Lemat 1) $K \subseteq_{\text{re}} A$

Lemat 2) $E \subseteq_{\text{re}} \bar{K}$

Wtedy $K \subseteq_{\text{re}} A \subseteq_{\text{re}} E \subseteq_{\text{re}} \bar{K}$ czyli $K \subseteq_{\text{re}} \bar{K}$ \Downarrow (także jest, pokazujemy)

D-d L1

f :
- wczytać n
- napisać program:
 $\begin{cases} * \text{wczytaj } m \\ * \text{odpł } \varphi_E(n) \\ * \text{zwroc } 1 \end{cases}$
- zwroci nr TEGO programu.

$n \in K \Rightarrow \varphi_n(n)$ staje $\Rightarrow \varphi_{f(n)} = 1 \Rightarrow f(n) \in A$

$n \notin K \Rightarrow \varphi_n(n) = 0 \Rightarrow \varphi_{f(n)} = 0 \Rightarrow f(n) \notin A$

D-d L2

E e co-r.e

Ψ : - wczytać n
- for $k=0$ to ∞ :
 $i, j = \text{bij}(k)$
if $\varphi_n(i)$ stanęło po j krokach
 \downarrow return 0

\bar{K} jest problemem co-re, trudny, więc $E \subseteq_{\text{re}} \bar{K}$

JF120 EGZ 2009 P II zad 4.

$X = \{ n \mid \text{Dom}(\psi_n) \text{ zawiera "długi odcinek"} \}$
 czyli $\exists k \in \mathbb{N} \forall [k, 2k] \quad \psi_n(i) \in N$

X jest r.e.

Ψ : - wciętej n
 - for $m=0$ to ∞

$$\langle i, j \rangle = \text{boj}(m)$$

adpal $\psi_n(i), \psi_n(i+1), \dots, \psi_n(2i)$ nie krzywia.
 jeśli wcięcie zwoźne to zwróci 1.

Ψ semi-rozstrzygająca X .

$n \in X \Rightarrow \text{Dom}(\psi_n) \text{ zawiera "długi odcinek"} \Rightarrow \exists k \in \mathbb{N} \quad \psi_n(k), \psi_n(k+1), \dots, \psi_n(2k) \text{ stoją po x sekundach}$
 \Rightarrow dla odpowiednich ciągów „m” program zwróci 1.

$n \notin X \Rightarrow \text{Dom}(\psi_n) \text{ nie zwoźne} \Rightarrow$ program wcięty nie zwoźnie.

JF120 EGZ 2008 II zad 4

$\forall B \in \text{r.e. } K$

b) oryginalne fct. K jest re-krzywy
 wcięcie fct. K jest r.e. czyli jest r.e.-zwoźny.

c) skonstruujemy redukcję fct. K $\nsubseteq G \in \text{r.e.} \iff f \in G \in K$

Redukcja: B jest r.e. więc istnieje program φ_B taki $\varphi_B(u) = 1 \iff u \in B$
 $\varphi_B(u) = L \iff u \notin B$

- wciętej n
 - napisz sobie taki program
 * wciętej m //ignorując
 * zwrócić $\varphi_B(n)$ } φ_K
 - zwróć numer tego programu (k) jako f_G

$n \in B \Rightarrow \varphi_B(n) = 1 \Rightarrow \varphi_K \text{ zwróci zwoźne 1 (ignorując wejście)}$

$\Rightarrow \varphi_K(k) = \varphi_{f_G}(f_G) = 1 \Rightarrow f_G \in K \text{ zdefiniowany zwoźnie K.}$

$n \notin B \Rightarrow \varphi_B(n) = L \Rightarrow \varphi_K \text{ zwróci zwoźne } L \Rightarrow \varphi_K(k) = L \Rightarrow \varphi_K \notin K \text{ zdef.}$

MM ≤ 3 Rosnące liczniki

Maszyna Minski'ego ma instrukcje postaci:

$$\langle q_i, \text{zero}^?, \text{zero}^? \rangle \rightarrow \langle q_j, \pm 1, \pm 1 \rangle$$

Zmodyfikowane z 3 rozszczepionego licznika.

$$q_i, c_1, c_2, c_3$$

\uparrow \uparrow \uparrow
 skryb. licznik 1 licznik 2 referencyjny pośredni zero.

$$c_1 - c_3 = \text{licznik 1 z modyfikacją Minski'ego}$$

$$c_2 - c_3 = \text{licznik 2} = \text{--}$$

$$\text{iszero}(\text{licznik 1}) \equiv c_1 \geq c_3 \wedge c_3 \geq c_2$$

$$\text{iszero}(\text{licznik 2}) \equiv c_2 \geq c_3 \wedge c_3 \geq c_2$$

$$\langle q_i, \text{zero}^?, \text{zero}^? \rangle \xrightarrow[T]{N} \langle q_j, +1, +1 \rangle$$

III

$$\langle q_i, c_1 \geq c_3 \wedge c_3 \geq c_2 \wedge \neg(c_2 \geq c_3 \wedge c_3 \geq c_2) \rangle \rightarrow \langle q_j, +1, +1, +0 \rangle$$

$$\langle q_i, \text{zero}^?, \text{zero}^? \rangle \xrightarrow[\text{III}]{N} \langle q_j, +1, -1 \rangle$$

$$\langle q_i, \psi \rangle \rightarrow \langle q_j, +2, +0, +1 \rangle$$

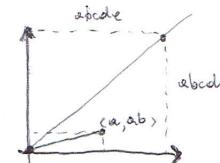
\uparrow \uparrow
 c₂ - c₃ modyfikator + 1.
 c₂ - c₃ modyfikator + 1.

PCP ≤_{rel} PCHKADaje nam liczniki r₁...r_n, stwierdzamy f₁...f_k g₁...g_n i w

$$\exists s \quad l_{s_1} \dots l_{s_{|S|}} = r_{s_1} \dots r_{s_{|S|}} \Leftrightarrow \exists s \quad f_{s_1} \circ \dots \circ f_{s_k}(0) = g_{s_1} \circ \dots \circ g_{s_n}(0)$$

Idea: stwórz będzie kodować liczby w systemie |Σ|-arym.

Każda liczba ma JEDNOZNACZNE kodowanie.



Redukcja: dla liczbego l_i tworzymy f_i
 $f_i(x) = x \cdot |\Sigma|^{|l_i|} + \text{VAL}(l_i)$

dla liczbego r_i tworzymy g_i
 $g_i(x) = x \cdot |\Sigma|^{|r_i|} + \text{VAL}(r_i)$

gdzie $\text{VAL}(w)$ to wartość liczby w zapisanej w systemie |Σ|-arym.PRZYKŁAD: $|\Sigma| = 10$

$$w = a_5a_7 // 57$$

$$l_5 = a_3a_2a_8 // 328$$

$$f_5(57) = 57 \cdot 10^3 + 328 = 57328$$

Istotnie PCP \Leftrightarrow istotnie skończone polity

(\Rightarrow) wciąż ten wyp, tak samo wykonywanie skończonej f_i g_j, obliczając wartości $\text{VAL}(l_{s_1} \dots l_{s_k})$ i $\text{VAL}(r_{s_1} \dots r_{s_n})$, które są różne, to liczby mają jednoznaczne kodowanie.

(\Leftarrow) jest ok tylko mamy do siebie liczby; wciąż jeż zapis w systemie |Σ|-arym. Także stwórz da się kiedyś l_i oraz r_i (zależ, że są zdefiniowane f_i g_j) \Rightarrow PCP ma rozw. \square

$A = \{n \mid \text{Dom}(\varphi_n) \text{ zawiera jakaś } k \text{ oraz pomyjnijącej } k \text{ wartością}$
 $\geq \text{przez } [k, 3k] \}$

1) A jest r.e.

Idea: przedstawienie po kolejci:

1, 1 2 3 na 1s

1, 1 2 3 na 2s

2 2 3 4 5 6 na 1s

1, 1 2 3 na 3s

2, 2 3 4 5 6 na 2s

3, 3 4 5 6 7 8 9 na 1s

⋮

Ψ : - wczytać n

- for m=0 to ∞

$\langle i, j \rangle = b_{ij}(m)$

if $\varphi_n(i)$ stanie się po j krokach:

adpal wszystkie $\varphi_n(k)$ na j kroków (dla $k \in [i, 3i]$)

jeśli pomyjniją "z" nich zwrocić to return 1.

Ψ semiorzędzająca A:

$n \in A \Rightarrow \exists i : \varphi_n(i) \text{ staje się } x \text{ krokach oraz } i, i+1, \dots, ik \in [i, 3i]$

że $\varphi_n(i), \varphi_n(i+1), \dots$ stają się odpowiednio po x_1, x_2, \dots krokach.

\Rightarrow jeśli dojdzie w końcu do m' tzn. $\langle i', j' \rangle = b_{ij}(m')$ oraz

$i' = i$ oraz $j' = \max(x_1, x_2, \dots, x_k)$, a wtedy program

zwroc 1.

$n \notin A \Rightarrow$ program nie powie 1, bo będzie działał
w nieskończoność (bo $\forall i \exists j \dots$)

2) $K \subseteq_{\text{rek}} A$

- wczytać n

- napisz program

$\begin{cases} * \text{ wczytać } m \\ * \text{ edytuj } \varphi_n(u) \\ * \text{ zwroc } 1 \end{cases}$

- zwrot nr TEGO programu

$n \in K \Rightarrow \varphi_n(u) \text{ staje się } \Rightarrow \text{Dom}(\varphi_{f(n)}) = \mathbb{N} \Rightarrow \exists k \dots \Rightarrow f_K \in A$.

$n \notin K \Rightarrow \varphi_n(u) \text{ nie staje się } \Rightarrow \text{Dom}(\varphi_{f(n)}) = \emptyset \Rightarrow \forall k \dots \Rightarrow f_K \notin A$.

Czyli A nie jest rek.

Zatem \bar{A} jest r.e.

wtedy A r.e i \bar{A} r.e \Rightarrow A rek

Spójrzcie, bo A nie jest rek.

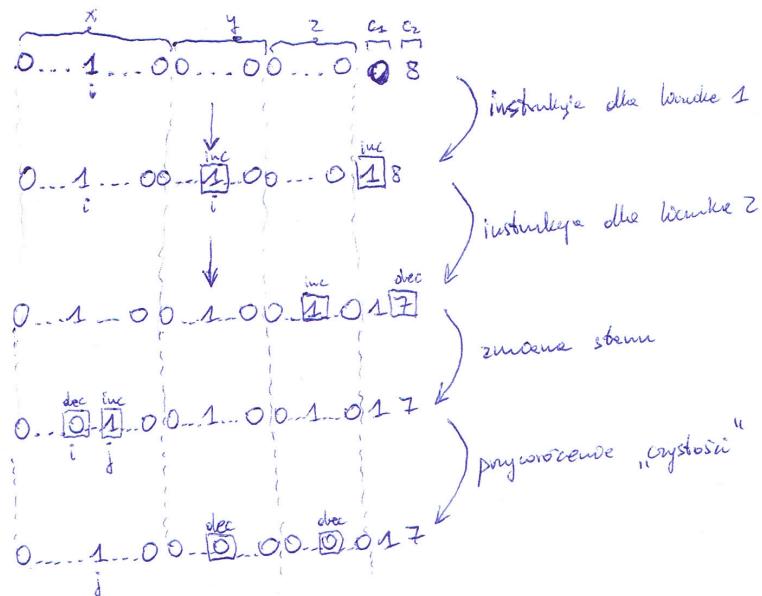
Czyli \bar{A} nie jest r.e.

Maszyna z n^1 komórkami, tylko 2 moze się zmieniać. - M

Plikarzny, i.e. ~~HALT~~ maszyny Moorejego jest wstępniejsze od HALT M.

Działam na maszynie Moorejego. Ma przejścia $\langle q_i, \Sigma, NZ \rangle \rightarrow \langle q_j, inc, dec \rangle$

Plikarzny jek rezultowaczą tątaką przejścia.



X - stan

Y - zakodowane, które instrukcje jsi wykonał.

UWAGA: Źeby to działało, musimy zmienić maszynę Moorejego na taka,

ktora izwra dla komórek kodyc w stanie: $(q_i, isZero(a), read(b)) \equiv q_X$

Maszyna z INSTRUKCJAMI zmieniającą $|Q| \cdot 2 \cdot 2$ instrukcji. $\{T,F\}^2 \times \{T,F\}$

$$c = \underbrace{10 \dots 0}_x \underbrace{00 \dots 00}_y, c' = \underbrace{010 \dots 00 \dots 00}_x \quad c \rightarrow c' \text{ mnożstwopolek.}$$

instrukcje
w roboc-

Minsky'sek TOTAL (NFC)

Stosu jest JEDNE jedyne prawidłowe masyngi Minsky'ego:

$$\text{conf} \# \dots \# q_i \# 2^a \cdot 3^b \# q_j \# 2^c \cdot 3^d \# \dots \# \text{conf} \#$$

conf n conf n+1
 $\underbrace{\qquad\qquad}_{\langle q_i, a, b \rangle}$ $\underbrace{\qquad\qquad}_{\langle q_j, c, d \rangle}$

Czy to prawidłowe jest ok?

w S musi być $(q_i, \text{iszero}(a), \text{iszero}(b)) \rightarrow (q_j, u_1, u_2)$
police $a+u_1=c$ i $b+u_2=d$

Daje nam masyng Minsky'ego, zakończony NFC iż
że M nie zatrzymuje \Leftrightarrow it akceptuje wszystkie słowa.

$$\Sigma = \{q \in Q\} \cup \{\$, \#, 0\}$$

Słowo jest JEDNE polegającym na wszystkich warunkach

1) jest postaci $(Q \# 0^* \$)^*$

$\underbrace{\qquad\qquad}_{\text{stos konfiguracji}}$
 \uparrow konfiguracja

2) ~~zakonieczności~~
dla każdego dwóch kolejnych konfiguracji wynikających z siebie = S MM.

Słowo BRZYDŁE jeśli chociaż jeden warunek nie zatrzyma.

Automat JEDNOZNACZNY wie w jakim położeniu były słowa ± 1 dla nowych warunków, bo MM jest deterministyczny.

Automat ZGADUJE co jest złe. 1) który wykona nową DFA. A co z 2)?

Automat ZGADUJE które konfiguracje są niepoprawne:

$$q_i \# \underbrace{000\dots 0}_{2^a \cdot 3^b} \$ q_j \# \underbrace{000\dots 0}_{2^c \cdot 3^d} \$$$

stany q_i, q_j zapamiętane w stach, $2^a \cdot 3^b$ wraca we stach.

Żeby wpisać w stan akceptujący musi ukończyć, że warunki
co i d są niepoprawne dla tej konfiguracji.

Jakie mogłyby być?

$$2^a 3^b \rightarrow 2^{a+1} 3^{b+1} = 2^a 3^b \cdot 2 \cdot 3 = 2^a 3^b \cdot 6$$

$$2^a 3^b \rightarrow 2^{a+1} 3^b = 2^a 3^b \cdot 2$$

$$2^a 3^b \rightarrow 2^{a+1} 3^{b-1} = 2^a 3^b \cdot 2 \cdot \frac{1}{3} = 2^a 3^b \cdot \frac{2}{3}$$

$$(*) 2^a 3^b \rightarrow 2^{a-1} 3^{b+1} = 2^a 3^b \cdot 3 \quad // 3x więcej \rightarrow więcej co 3 mnoż.$$

$$2^a 3^b \rightarrow 2^{a-1} 3^{b-1} = 2^a 3^b \cdot \frac{2}{3} \quad // 6x mniej \rightarrow mniej po 6 mnoż.$$

Czytając konfigurację conf_n oznacza to sprawdzenie:

* stan

* iszero(a) $\Leftrightarrow 2^a 3^b$ nie ma żadnego sześcianu 2 // mniej niż dwoje sześcianów co dwojki

* iszero(b) $\Leftrightarrow 2^a 3^b$ nie ma żadnego sześcianu 3 // (co dwoje) by nie było sześcianów

Z tego JEDNOZNACZNY wie w jakim położeniu były słowa ± 1 dla nowych warunków, bo MM jest deterministyczny.

Dla ustalenia uwagi $(q_i, \text{iszero}(a), \text{iszero}(b)) \rightarrow (q_j, +0, +1)$
czyli sytuacja (*). Mode zweryfikować, czy jest ok swiązanie
ze stach "0" co tury powyższe "0" (bo nie ich być 3x więcej).

Jest: 1° stos niepusty a wartość "

lub

2° stos pusty a jeszcze co góry wartości "0"

To znaczy NIEPOPRAWNE PRZEJŚCIE KONFIGURACJI i akceptuje słowo.

Zatem MM nie staje \Leftrightarrow NFC akceptuje wszystkie słowa
Totalność jest nierozstrzygająca.

$A_0 = \{ n \mid \text{Dom}(\varphi_n) = \emptyset \}$ - napisy nie stoją

$A_1 = \{ n \mid |\text{Dom}(\varphi_n)| = 1 \}$ - stoją tylko dla jednego argumentu.

$A_0 \subseteq A_1$ TAK

Napiszemy redukcję:

f: - wyrabia n
 - napisz program
 { * wyrabia m
 * jeśli $m=1$ return 1
 * $\langle i, j \rangle = bij(m)$
 * odpal $\varphi_n(i)$ na j kroków
 ↳ jeśli zwróciło to zwrócić 1
 ↳ else zapełnij się
 - zwróci numer TEGO programu.

$n \in A_0 \Rightarrow \varphi_n(i) = \perp \Rightarrow \varphi_{f(n)}$ stanie tylko dla $m=1$

$\Rightarrow \text{Dom}(\varphi_{f(n)}) = \{1\} \Rightarrow f(n) \in A_1$

$n \notin A_0 \Rightarrow$ istnieje jakieś k że $\varphi_n(k)$ staje po x sekundach

\Rightarrow mamy następujące wartości dwóch m , że $\langle i, j \rangle = bij(m)$
 $i = k$ i $j > x$, dla których $\varphi_{f(n)}$ się zatrzyma

$\Rightarrow |\text{Dom}(\varphi_{f(n)})| > 1 \Rightarrow f(n) \notin A_1$

◻

$A_2 \leq_{\text{rek}} A_0$? NIE

Intuicja: $A_0 = \{ n \mid \text{Dom}(\varphi_n) = \emptyset \}$

$= \{ M_n \mid \underbrace{\forall x \forall t}_{} M_n(x) \text{ nie jest po } t \}$

Π_1 (co-re) \rightarrow cykl ta sama "banka" co K

$A_2 = \{ n \mid |\text{Dom}(\varphi_n)| = 1 \}$

$= \{ M_n \mid \underbrace{\exists x \exists t_1 \forall y \forall t_2}_{y \neq x \wedge M_n(x) \text{ stage po } t_1 \wedge M_n(y) \text{ nie stage po } t_2} \}$

$\Sigma_2 \rightarrow$ cykl znajdujący się w K

Lemat: $A_0 \leq_{\text{rek}} K$, $K \leq_{\text{rek}} A_2$

Dowód: $A_2 \leq_{\text{rek}} A_0$. Wtedy $K \leq_{\text{rek}} A_2 \leq_{\text{rek}} A_0 \leq_{\text{rek}} K$

czyli $K \leq_{\text{rek}} K \Leftrightarrow$ (pokazujemy, że taś nie jest)

$A_0 \leq_{\text{rek}} K$:

$\overline{A_0} \in \text{re},$ czyli A_0 co-re, K jest co-re. trwając cykl $\overline{A_0} \in \text{rek}$

$K \leq_{\text{rek}} A_2$:

Redukcja: - wyrabia n
 - napisz program:

{ * wyrabia m
 * odpal $\varphi_n(n)$
 * if $m=0$ return 1
 * else zapełnij się
 - zwróci numer TEGO programu

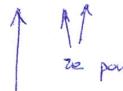
$n \in K \Rightarrow \varphi_n(n)$ staje $\Rightarrow \varphi_{f(n)}$ tylko dla zera zauważ 1 $\Rightarrow |\text{Dom}(\varphi_{f(n)})| = 1 \Rightarrow f(n) \in A_2$

$n \notin K \Rightarrow \varphi_n(n)$ nie staje $\Rightarrow \varphi_{f(n)} = \perp \Rightarrow |\text{Dom}(\varphi_{f(n)})| = 0 \Rightarrow f(n) \notin A_2$

JF120 EGZ 2013 II zad 4.

a) TAK, bo mamy układ równań liniowych i możliwe
to jest rozwiązywanie używając wzoru Gramera.

b) $\text{Th}(N_1^{*,+}, \leq, 0, 1) : \exists x_1 \dots x_n \phi(x) ?$ NIEPODSTAWIONE.



za pomocą tego wygenerowany będą metanequation

za pomocą tego wygenerowane równości: $x \leq y \wedge y \leq x$

$$\exists x_1 x_2 \dots x_n \Phi(x)$$



tu możemy opisać UKŁAD RÓWNAŃ

Czy istnieją też te $x_1 \dots x_n$ naturalne.

Czyta się pytanie skojarzenie układów równań do fabrycznych.

Gdyby było rozwiązywalne, to był problem Hilberta równowaz.

A nie jest, więc NIEPODSTAWIONE.

JF120 EGZ 2013 II zad 5.

$\frac{3}{2}$ PDA jest rozwiązywalny (deterministyczny)

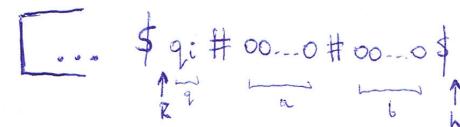
$$MM \leq_{\text{rok}} \frac{3}{2}\text{PDA}$$

Daję nam maszynę Turinga, skonstruującą $\frac{3}{2}\text{PDA}$ st. t.ż.

MM są robiące \Leftrightarrow st. ostateczny stan akceptujący

Idea: - głowica porusza się kolejne konfiguracje maszyny

- głowica czytająca cytry poprzednich konfiguracji.



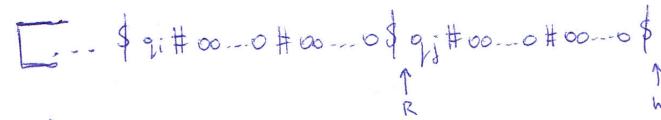
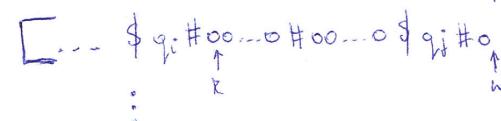
1. Głowica R idzie w prawo i odczytuje:

stan, (zroba), (zroba(b))

dokąd ten sam wie, jaka konfiguracja powinna być nastepna, np.

$(q_5, \text{zero}, \text{not zero}) \rightarrow (q_8, +1, +1)$

2. Głowica R wraca do \$, W wpisuje stan, i zauważa się przepisujące brudów (po jednym „kamyczku” z uzupełnieniem ± 1)



i znowu zaczyna punkt 1.

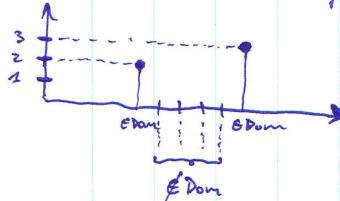
Także R przeczyta „q_f” to automat przechodzi w stan akceptujący i się zatrzymuje.

Dowód wykorzystuje się argumenty.

- f:
- 1) jest "na"
 - 2) jeśli określone we $x \in$ to $f(x) < f(a)$ (czyli nieuję)
- Czy $\text{Dom}(f)$ jest zbiorem pełnozacyplnym? TAK.

$x \in \text{Dom}(f)$ jest zbiorem y t.j. $f(x) = y$

$x \notin \text{Dom}(f)$ jest zbiorem tych $a < x < b$ d.t. $f(a) + 1 = f(b)$
(czyli tyle co, bo f jest "na")



- Y:
- odpalaj na koniec obliczenia dodaję znak
 - rozważ casey kiedy cośś przedstawia przekształcenie:
 - $\rightarrow f(x); f(x+1), f(x), f(xz); f(x-2), f(x), f(x), f(xz), f(xz); \dots$
 - rozważając sobie a - najmniejsze $f(x+c)$ // to leży od x
 - rozważając sobie b - największe $f(x+c)$ // to leży od x

jeśli $a = b - 1$ i argument wej. jest zakończony we x
to return 0
jeśli inak wtedy zwracaj w x do return 1.

Zauważ że zakończenie jest "na" i skończone.

$$\overline{\text{PCP}} \leq \text{TOTALCFG}$$

Dlaż mamy $l_1 \dots l_n r_1 \dots r_n$, zbudujmy gramatykę bezkontekstową G

$$t.d. \exists s \quad l_{s_1} \dots l_{s_L} = r_{s_1} \dots r_{s_L} \Leftrightarrow L(G) = \Sigma^*$$

Stwórz w' jest ŁADNE jeśli kończy się znakiem dla PCP:

$$i_{s_1} l_{s_1} \dots i_{s_L} l_{s_L} \# i_{s'_1} r_{s'_1} \dots i_{s'_L} r_{s'_L}$$

Czyli zakończenie następujące warunki:

$$1) w \in (\Sigma_1 \Sigma_2^*)^* \# (\Sigma_1 \Sigma_2^*)^*$$

2) po każdym i_{s_k} jest odpowiednie l_{s_k} (po $\#$)

po każdym i_{s_k} jest odpowiednie $r_{s'_k}$ (po $\#$)

3) jeśli zmieniająca się leżący w Σ_2 to zostają same indeksy i $\#$ i ma to być palindrom.

4) jeśli zmieniająca się leżący w Σ_1 to zostają same leżące w alfabetie PCP i tworzą palindrom.

$w' \text{ ŁADNE} \Leftrightarrow 1) \wedge 2) \wedge 3) \wedge 4)$

$w' \text{ BRZDZIE} \Leftrightarrow 1) \vee 2) \vee 3) \vee 4)$

Skonstruujmy NPDFA ft t.j. ft akceptuje w $\Leftrightarrow w$ jest BRZDZIE.

ft nedeterministyczne spodobać poważ braków słów. 71 i 72 unik bo unik naszej DFA. 73 i 74 też unik bo nie stac.

Czyli ft praktycznie akceptujący \Leftrightarrow jest jedynie poważ braków słów $\Leftrightarrow w' \text{ BRZDZIE} \Leftrightarrow 73 \text{ powo PCP}$

7 automaty ft tworzące CFG G_{ft}.

Zatem jest istotnie algorytmu TOTALCFG to mamy teraz rozwiązania PCP.

$\Phi_B(n, k) :$ $\varphi_n = \varphi_k \rightarrow \text{TAKE}$
 $\varphi_n \neq \varphi_k \rightarrow \text{NO GO}$
 φ_n lub φ_k nie zdefiniowane \rightarrow zatrzymać

NIE istmoge schw elponythen.

Zat, se obrazce, učebny program, ktery rozšíří K.

Ψ :
- wczytał n
- wąsze programy:

- * wczytaj m
 - * odpel ypn(u) we m sekundach
 - > foto zeg zebowaty do
 - > jelsz we sekundy do

- wstęp programu!
- * weryfikuj 1
- * zwroc 0

→ zwroc Alg (l,m)

$$\left. \begin{array}{l} 1 \\ 2 \\ 3 \end{array} \right\} \Psi_m$$

$n \in \overline{K} \Rightarrow n \notin K \Rightarrow$ φ_n ist keine reelle Zahl \Rightarrow φ_n keine ganze Zahl 0

$\text{Alg}(l_m) \neq 1 \text{ (TAK)} \Rightarrow \psi(n)=1$

$n \notin K \Rightarrow n \in K \Rightarrow \gamma(n)$ zukünftige Ws pos schwindet, ergibt keinen
 zweiten 1 $\Rightarrow \text{Alg}(l, n)$ zweite 0 (NIE) $\Rightarrow \psi(n) = 0$

Ψ rozstupuje zboží K a on uveďte jeho relevantní myšlenky.

FAKT. Rustet i nærpastet alle CFG set nærtypiske.

Alponykin

- * zaznacz wszystkie terminale
 - * postawaj słownik ^{coś} ~~zapisu~~ monoga ~~zapisu~~ zaznaczyłszy:
 - zaznacz nietermal A t.j. reprezentację $A \rightarrow x_1 \dots x_k$
 - i wszystkie $x_1 \dots x_k$ są już zaznaczone
 - * jeśli symbol S jest zaznaczony to
 - return "NIEPUSTY"
 - else return "PUSTY"

a) Nemp NPDIA : ROZSTRZEGALNY

$\text{fb} \rightarrow \text{CFG}$ $G \rightarrow \text{Empty}$ $G \xrightarrow{\text{NFA}} \text{NLE}$

↓
widthening
are widthwise

b) Neues DDDIT: ROZSTRZETGALNFT

- 14 -

c) Total NPDA: Nicrostypus

By the polarons we understand, if $L(G) = \varepsilon^*$ just microstrophes

$f \rightarrow \oplus$ i.e. $L(G) = L_{\oplus}$ zero-stypalve.

d) Total DPPA: $P025TP>ECA/NE$

$\forall A \in \text{DPDA}, L_A \in \text{DCFL} \Leftrightarrow \overline{L_A} \in \text{DCFL}$ (Please remember no dependence)

$A \rightarrow \overline{A} \rightarrow \text{DCFG dla } \overline{A}$, $\overline{A} \in \text{EMPTY} \Leftrightarrow A \in \text{TOTAL}$

if (\emptyset) return 1 // 0 TOTAL
 else return 0 // we take

$\text{PCP} \leq_{\text{rek}} \text{Namp} (\text{PDA Stack Rev})$

Zauważmy, że możliwość obracania stosu daje nam możliwość operowania na dwóch stosach. Musimy mieć jednak specjalny symbol $*$, który będzie oznaczać, że za nim jest już drugi stos.



to, który stos (s_1 czy s_2) jest „na górze” kodujemy w stowach automatów.

Dajemy nam $l_1 \dots l_n r_1 \dots r_n$, skonstruujemy ft t.z.:

$$\exists s. l_{s_1} \dots l_{s_l} = r_{s_1} \dots r_{s_l} \Leftrightarrow \exists w. f \text{ akceptuje } w.$$

Intuicja: w musi być zdaniem dającym rozwojowe PCP.

ŁADNE słowo: $i_{s_1} l_{s_1} \dots i_{s_l} l_{s_l} \# i_{s_1} r_{s_1}^R \dots i_{s_l} r_{s_l}^R$

Słowo jest ŁADNE, jeśli zachodzi następujące warunki (wszystkie):

- 1) jest postacie $(\Sigma_1 \Sigma^*)^* \# (\Sigma_1 \Sigma^*)^*$
- 2) po krescegu $\#$ jest l_{s_m} i przed l_{s_m} // przed $\#$
~~lub -~~ $r_{s_m}^R$ // po $\#$
- 3) patrząc na same indeksy (Σ_2) : $\#$ musi być palindromem
- 4) - - - słowo (Σ_2) : $\#$ musi być palindromem.

Zatem skonstruować ft, który akceptuje słowa spełniające warunki 1, 2, 3 i 4.

1 i 2 da się sprawdzić w stowach (jako DFA); 3 i 4 potrzebują dwóch oddzielnych stosów, a to mamy, bo mamy obracanie stosu.



$$A_0 = \{ n \mid \varphi_n(n) = 0 \}$$

$$A_1 = \{ n \mid \varphi_n(n) = 1 \}$$

Zat. iż $\exists B$ rekurencyjny t.z. $A_0 \subseteq B$ i $A_1 \cap B = \emptyset$

Dowód przekształcający.

B rekurencyjny, więc istnieje ψ_B t.z. $n \in B \Leftrightarrow \psi_B(n) = 1$ i $n \notin B \Leftrightarrow \psi_B(n) = 0$

$$\psi_B = \psi_k \text{ (ma numer „k”)}.$$

$$k \in B \Rightarrow \psi_k(k) = 1 \Rightarrow \varphi_k(k) = 1 \Rightarrow k \in A_1 \Rightarrow k \notin B$$

$$k \notin B \Rightarrow \psi_k(k) = 0 \Rightarrow \varphi_k(k) = 0 \Rightarrow k \in A_0 \Rightarrow k \in B$$

Czyli $k \in B \Leftrightarrow k \notin B$ SPŁECZNOSC.

Zatem nie istnieje taki B rekurencyjny.

PCP \leq_{rek} GrKontekstDaje nam instancje PCP $l_1 \dots l_k \# \dots r_k$. Stwierdzamy $G \stackrel{i.w}{\checkmark} \text{Y.2e}$

$$S \xrightarrow{*} \omega \Leftrightarrow \exists s \quad l_{s_1} \dots l_{s_k} = r_{s_1} \dots r_{s_k}.$$

 $\omega = \epsilon$

$$G_{\Pi}: \quad S \xrightarrow{*} \overbrace{l_i}^m S \xrightarrow{*} \overbrace{r_i}^n \quad | \quad \epsilon \quad \text{gdzie } \overbrace{l_i}^m = L_0 L_1 \dots L_m \quad \text{t.j. } a_0 \dots a_n = l_i \\ S \xrightarrow{*} \overbrace{l_i}^m S \xrightarrow{*} \overbrace{r_i}^n \quad | \quad \epsilon \quad \text{gdzie } \overbrace{r_i}^n = \overbrace{b_0 R b_1 R \dots b_n R}^m \quad \text{t.j. } b_0 \dots b_n = r_i$$

i chcemy jesczniej uzyskac kontekstowe zapisanie:

$$\begin{aligned} L \bar{a} \bar{a} R &\rightarrow L \bar{a} R \quad // \text{tylko jedna litera może} \\ L \bar{a} R &\rightarrow L R \quad \text{zmienić} \\ \bar{a} &\rightarrow \epsilon \\ \bar{R} &\rightarrow \epsilon \end{aligned}$$

$$\Pi: N \times (N \cup T)^*$$

↑ ↑
 1 0, 1, ...

PCP nie rowne $\Rightarrow S \xrightarrow{*} \epsilon$ generujemy rozumianej $S \xrightarrow{*} l_{s_1} \dots l_{s_k} r_{s_1}^n \dots r_{s_k}^n$
a potem kontekstowe zapisujemy i dostajemy ϵ . $S \xrightarrow{*} \epsilon \Rightarrow \text{PCP nie rowne}$

musimy zejsc od $S \xrightarrow{*} S'$ wiedc nie bedzie puste
 i aby bylo puste to musimy zmienic, o moducy zmienic tylko jeden
 sc DOKONCZNIE DWIE LITERKI $L \bar{a} \bar{a} R$. wie du sobie nie zapisujesz
 i rozumowaniem skapej jest jak w dowolne dla kontekstowego
 zapisu - tez w mojosci zmienic $aa \rightarrow \epsilon$.

Zat, ze istnieje program, ktory dla danygo ukladu rownan
 definiujacych z $\{+, -, 1, *, 2\}$ oblicza $*^n$ rozszerzenia, czy
 uklad ma jedno rozszerzenie. Polesci, ze da sie juz japo
 pomocy rozszerzeniem ukladu z $\{+, -, *, 1\}$.

~~(a+b)² = a²+b²+2ab~~ $\Rightarrow a \cdot b = \frac{(a+b)^2 - a^2 - b^2}{2}$

- wynik ukladu $E(+, -, *, 1)$
- dla konkretnego ukladu $a \cdot b$ zapisu na $\frac{(a+b)^2 - a^2 - b^2}{2}$
 rozszerzenie sog nie znam.
- obliczanie ukladu wykonalosc $\Psi(E, +, -, 1, *)$

Dla zadanych. Podejmy, że $\forall f \vee \exists$ obraz f jest nieskończony
 $\wedge f$ jest całą relacją: $K \subseteq \text{dom } Af$.

Daję nam f całą relację o nieskończonym obrazie.

Napiszemy rekurencję F t.j. $n \in K \Leftrightarrow F_n \in Af$.

F : - wczytaj n

- wpisz program:

* > wczytaj m
 * > odpal $\varphi_n(u)$ z komórką
 ↳ jeśli wychodzi max $f(m)$ mniej niż m
 ret 1
 ↳ else zapełnij sug .

- zwrócić numer TEGO programu.

$n \notin K \Rightarrow \varphi_n(u)$ wpisywała sug nie zwracała \Rightarrow bierze resztę w
 nieskończoność i w koncu wychodzi poza $f(m)$ mniej niż

$\Rightarrow \exists m \in \text{dom } \varphi_{F(n)}(m)$ sug zwracała $\Rightarrow F(n) \in Af$.

$n \in K \Rightarrow \varphi_n(u)$ zwracała sug po x komórkach więc zajmuje max

" x " dodatkowej pamięci (unarynny biermek) \Rightarrow skoro obraz f

jest nieskończony to istnieje m t.j. $f(m) > x$ i $\varphi_{F(n)}(m)$

sug zwracała $\Rightarrow F(n) \in Af$

Dwie nowe klasyczne \equiv Dom(φ_n) ko-skierowane \equiv Dom(φ_n) skierowane
 III
 skierowane wiele \perp .

$A = \{n \mid \text{Dom}(\varphi_n) \text{ ma skończone wiele } x \in \mathbb{N} \wedge \text{nie skierowane wiele } \perp\}$.

a) A nie jest rekurencyjny.

$K \subseteq \text{dom } A$

- wczytaj n
 - wpisz program
 * wczytaj m
 * odpal $\varphi_n(u)$ na m komórkach
 ↳ jeśli skierowano zapełnij
 ↳ else ret 1
 - zwróci nr TEGO programu

$n \in K \Rightarrow \varphi_n(u)$ staje po $x \Rightarrow \forall m > x$

$\varphi_{F(n)}(m) = 1 \Rightarrow$ wiele wiele \perp

$\wedge \forall m < x \varphi_{F(n)}(m) = 1 \Rightarrow$ skierowane wiele

$f(u) \in A$

$n \notin K \Rightarrow \varphi_n(u)$ nie staje $\Rightarrow \forall m \varphi_{F(n)}(m) = 1$

$f(u) \notin A$

b) A wiele jest r.e

$\bar{K} \subseteq \text{dom } A$

- wczytaj n
 - wpisz program
 * wczytaj m
 * odpal $\varphi_n(u)$
 * odpal $\varphi_n(u)$ na m komórkach
 ↳ jeśli skierowano zapełnij \perp
 ↳ else ret 1
 - zwróci nr TEGO programu

$n \in \bar{K} \Rightarrow \varphi_n(u)$ wiele $\Rightarrow \varphi_{F(n)}$ wiele

w koncji (*) $\Rightarrow \forall m \varphi_{F(n)}(m) = 1$

$f(u) \in A$

$n \notin \bar{K} \Rightarrow \varphi_n(u)$ staje po x komórkach
 $\Rightarrow \forall m > x \varphi_{F(n)}(m) = 1 \Rightarrow$ nieskończoność
 wiele \perp i skierowane wiele \perp
 $\Rightarrow f(u) \notin A$

Czy istnieje ustalone gramatyka G , t.j. problem: Czy dla CFG H

$L(G) \cap L(H) \neq \emptyset$? jest rozwiązywalny?

TAK, istnieje.

Problem PCP jest rozwiązywalny.

$L_p = i_1 l_1 \dots i_k l_k \# i_k r_k^R \dots i_1 r_1^R$ // słowa kodujące rozwiązań PCP.

$L_p \neq \emptyset$? rozwiązywalne (bo rozwiązywalność pop).

$w \in L_p \Leftrightarrow$ zachodzą warunki 1), 2), 3)

1) $w \in (\Sigma_1 \Sigma_2^*)^* \# (\Sigma_1 \Sigma_2^*)^*$, po ilość $\#$ (pośredni $\#$), po ilość r_i^R (po $\#$)

2) pierwsze i ilekce we wcieleniu w^H to permutacjami τ $\#$ we środku

3) pierwsze i ilekce we wcieleniu w^H to permutacjami τ $\#$ we środku.

$$L_p = L_1 \cap L_2 \cap L_3$$

$\begin{matrix} \uparrow & \uparrow & \uparrow \\ \text{regularny} & \text{cfl} & \text{cfl} \end{matrix}$

$$\underbrace{L_1 \cap L_2}_{\text{cfl bo cfl} \wedge \text{regularny} = \text{cfl}} \cap L_3$$

Ten ustalony język to $L_1 \cap L_2 = L_G$, ustalony gramatyka to G' .

Nierozwiązywalne jest nieskończoność $L_G \cap L_H$, bo polegają bycie, to rozwiązywalnych nieskończoności L_p . §

2MAXSAT jest NP-zupełny.

1. 2SAT_{89%} ∈ NP

→ daje nam 6 i sprawdza my, czy ok (czy 89% ok)

2. 3SAT \leq_p 2SAT_{89%}

Dla każdej klawiszu ($a \vee b \vee c$) tworzymy:

~~a~~ \oplus ~~b~~ \oplus ~~c~~ \oplus ~~d~~

$$(a \vee b) \wedge (\neg b \vee c) \wedge (\neg c \vee \neg a) \\ (a \vee \neg d) \wedge (b \vee \neg d) \wedge (c \vee \neg d)$$

10 klawiszy

abc tutaj symetryczne. Ile modyfikacji spowoduje klawisz?

TTT → max 7	} $\frac{7}{10}$	many rozróżnione sytuacje
TTF → max 7		
TFP → max 7		
FFF → max 6		

Dodajemy jeszcze dla każdej klawiszu ~~290~~ prawidłowych

$$\text{klawisz } (x \vee \neg x) \left(\frac{\frac{7+c}{10+c} = \frac{89}{100}}{\uparrow \text{prawidłowe}} \right).$$

Wtedy dla każdej klawiszu TRUE \Leftrightarrow 89% prawidłowe.

Dowodzimy w obie strony trywialnie.

2CNF₁₀₀ ∈ PTIME

Wiemy, że 2CNF ∈ PTIME (redukacja). Zauważmy, że 80 z pierwszych 100 to statek $c = \binom{100}{80}$ i nie zależy od wojew.

Przetestujemy wszystkie e możliwości.

```

    ∀ możliwe wybory 80 ze 100 klawiszy // ∈
        if spelnione( 80 wybranych, 100...n ) // O(n^2)
            ret true
        ret false
    
```

Z n możliwymi modyfikacjami $\sqrt{4n^2 + 2n + 1}$
wybranych klawiszy. Iterując się i sortując "L". $\frac{1}{(2n)(2n)} \stackrel{\text{max}}{\uparrow} \stackrel{\text{Pvq}}{\uparrow} \stackrel{\text{Pip}}{\uparrow} \stackrel{\text{L}}{\uparrow}$

par-W-chrom \leq_p par-W-chrom

Dajż nam G , skonstruujmy G' t.z. $2 \mid \chi(G) \Leftrightarrow 2 \nmid \chi(G')$.



\checkmark podział na kolor wierzchołka wewnątrz G .

\checkmark musi mieć innego kolor wierzchołka wewnątrz G zatem
 $\chi(G') = \chi(G) + 1$.

$$\Rightarrow 2 \mid \chi(G) \Rightarrow 2 \mid \chi(G) + 1 \Rightarrow 2 \nmid \chi(G')$$

$$\Leftarrow 2 \nmid \chi(G) \Rightarrow 2 \nmid \chi(G) + 1 \Rightarrow 2 \nmid \chi(G')$$

par-W-chrom \leq_p par-W-chrom

\rightarrow redukcja i dowód identyczny.

par-W-chrom jest NP-trudny (zad 9), ale nie amerykańskie należą do NP. dajż nam kolorowane i co?
 a może dla się lepiej...?

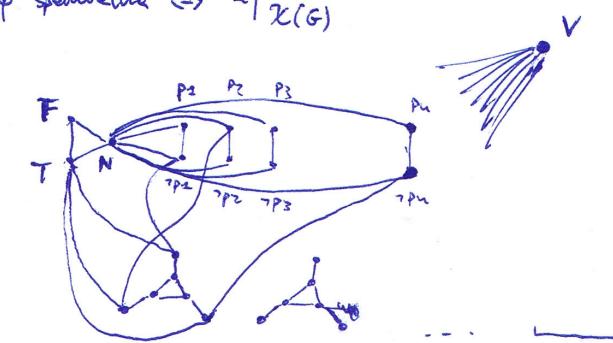
Gdyby par-W-chrom był NP-zupełny, to par-W-chrom jest NP-zupełny
 a to wyprowadzi矛盾 contradiction, bo par-W-chrom jest redukcyjny,
 tzn. tyle mniej jest.

3SAT \leq_p par-W-chrom

redukcja będzie identyczna jak dla 3SAT \leq_p 3COL i zad 7.

Dajż nam formułę φ , stworzącą graf G t.z.

$$\varphi \text{ spełniona} \Leftrightarrow 2 \mid \chi(G)$$



3 fajce (FFF) to są wtedy dla pokolorować, bo zostaje tylko kolory NNN'. V-wierzchołek reprezentuje połączony ze wszystkimi musi mieć innego koloru.

\rightarrow Nie da się pokolorować dwóch kolorów to many K_3 (Δ).

\rightarrow Da się 3+1 lub 4+1 kolorów.

$$\Rightarrow \varphi \text{ spełn} \Rightarrow \text{da się } 3+1 \text{ kolorów} \Rightarrow \chi(G) = 4 \Rightarrow \text{par-W-chrom}$$

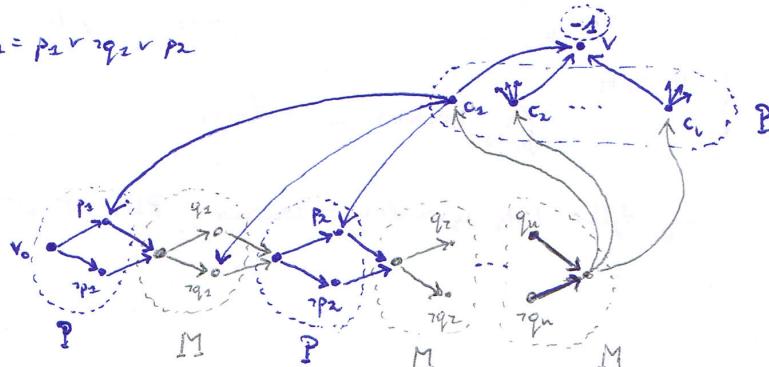
$$\Leftarrow \varphi \text{ niespełn} \Rightarrow \text{da się } 4+1 \text{ kolorów} \Rightarrow \chi(G) = 5 \Rightarrow \text{niedarzelnik}$$

PlusMinus jest PSPACE-zupełny.

1. TQBF \leq_p PlusMinus

Dajmy nam formułę $F = \exists p_1 \forall q_1 \dots \varphi$, zbudujmy $\langle G, v_0, P, M, g \rangle$
t.j. F prawdziwa \Leftrightarrow Plus ma strategię wygrywającą.

$$c_1 = p_1 \vee \neg q_2 \vee p_2$$



$$g : \begin{cases} \forall u \in V \forall v g(u) = 0 \\ v \rightarrow -1 \end{cases}$$

* wszystkie pi wybiera P, wszystkie qj wybiera M.

* pośiadanie karmienia to nadając wartość FALSE.

* gracz M wybiera karmienie, jeśli FFF to gracz P nie ma
wyboru i musi przesunąć karmieniem na V, s=-1 i wygrywa.
jeśli jednak wszystkie karmienie spłytowane, to może postawić
karmieniem jedynie inny niż v (pi lub qj) a stanąć
M lub P nie może się rozróżnić więc s=0 i P wygrywa.

(Redukcja wiadomościowa)

2. PlusMinus ∈ PSPACE

aktualny węzeł jest skonfigurowany
PlusWins(v, s, L)

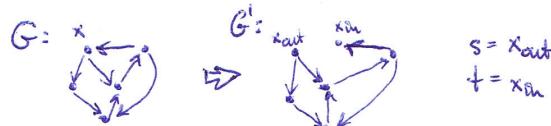
{ IF NO NEIGHBOUR OR EVERY NEIGHBOUR IN "L"
IF S ≥ 0 RETURN TRUE
ELSE RETURN FALSE

ELSE
IF SEP
RETURN $\bigvee_{u \in \text{neigh}} \text{PlusWins}(u, s+g(u), v=L)$ // jeśli nach
P wygrywa
IF SEM
RETURN $\bigwedge_{u \in \text{neigh}} \text{PlusWins}(u, s+g(u), v=L)$ // moździerze odc
M wygrywa

parametry tylko liczące sądowów - londowe, PSPACE ✓

a) $HAM_d \leq_p PATH_d$

Dlaż nam G , staczyj G^i , t.ż. w G istnieje cykl Hamiltona wtedy w G^i istnieje ścieżka Hamiltona z s do t .

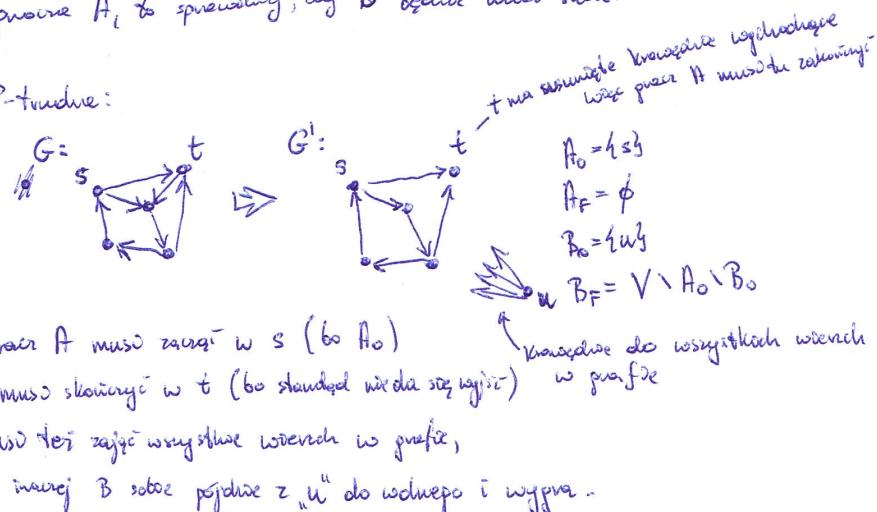


→ wytwarzany dorośły wierzchołek w G i po rozdrożony na x_{in} i x_{out} (tylko krawędzie wejściowe/wyjściowe)

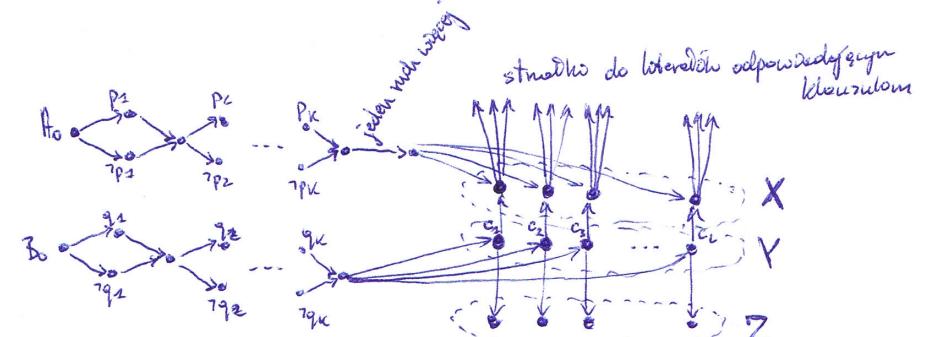
b) $PATH_d \leq_p G7b$

To jest NP-zupełne. $G7b$ GNP, jeli nam dadas strategie dla $G7b$ gracze A_i to sprawdzy, czy B będzie mógł such.

NP-trudne:

 $TQBF \leq_p G8$

Dlaż nam zadanje $\exists p_1 \dots \forall q_k \psi$, robimy instancję $G8$ t.ż.
zadanie prawdziwe \Leftrightarrow gracz A nie staczyj wygrywając.



$$A_F = \emptyset, B_F = X$$

Postawienie piórka oznacza zwartość/zwarcie nor FALSE.

Najpierw gracz A postawiający zadanje, potem B wybiera klawisz (wierzchołek ze zbioru V), a gracz A musi postawić w odpowiadającym wierzchołku z X , bo inaczej B mógłby się wiedzieć tam i by wygrał (bo osiągnąłby wierzchołek ze zbioru B_F). Teraz B ma tylko jedną możliwość: postawić do zbioru Z , jeśli klawisz może chociaż jedno TRUE, to gracz A się tam rusza; kolej gracza B (a on nie ma już dokąd iść z Z) i przypływa; jeśli byłby same FALSE, to A nie wie gdzie iść.

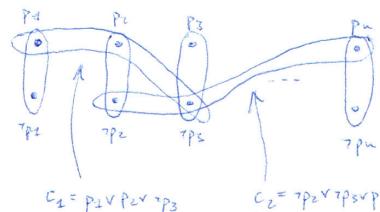
a) KLÓTNIKA jest NP-zupełna // tutaj nie interesuje nas G

1) KLÓTNIKA ∈ NP

→ daje nam „pokolorowane” społeczeństwo na mochery i lewiny;
my sprawdzamy, czy VccC istnieje taki 1 moher i 1 lewina

2) KLÓTNIKA jest NP-trudna.

NAESAT \leq_p KLÓTNIKA



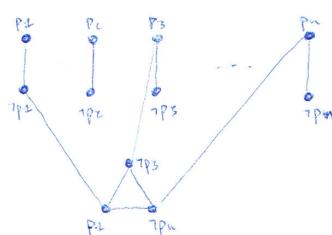
moher \equiv TRUE
lewina \equiv FALSE

1. $p_1 \wedge \neg p_1$ są w zbiorze
wtedy jedno T a drugie F
2. $p_2 \wedge p_3 \wedge \neg p_4$ są w zbiorze
wtedy skrócenie wini pod NAE

b) KLÓTNIKA jest NP-zupełna // tutaj nie interesuje nas G

1) KLÓTNIKA ∈ NP ✓

2) NAESAT \leq_p KLÓTNIKA



$$k = n + 3l + 2l$$

\uparrow zmienné
 \uparrow Krawędzie wewnętrzne klawuli
Krawędzie między klawulami a literami

1. $p_i \wedge \neg p_i$ węzeł w konfliktach
2. od klawuli fagującej z przewinętymi literami, węzeł 3L konfliktów

3. $\begin{array}{c} \bar{1} \\ \diagup \quad \diagdown \\ T \quad T \end{array} \equiv 0$ konfliktów
- $\begin{array}{c} \bar{1} \\ \diagup \quad \diagdown \\ T \quad F \end{array} \equiv 2$ konfliktów
czyli jeśli 2L konfliktów
to każda klawula NAE

To jest problem NP-zupełny.

1° COL4-CYCLE ∈ NP

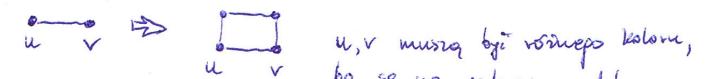
daje nam kolorowanie (maszyna Turinga nie deterministyczna je zatrzymuje) i potem sprawdzamy, czy wszystkie cykle są kolorowane.

2° 3COL \leq_p COL4-CYCLE

Daje nam G, stwierdzamy G' t.j.e ...

Musimy zadbać o dwie rzeczy:

- 1) Jeśli taka krawędź w G, to w G' muszą być innego koloru:

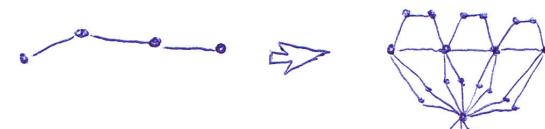


- 2) Każdy wierzchołek $\in G$, musi mieć w G' kolor 1 lub 2 lub 3.



globalne ustawnianie co będzie kolorem „4”, wtedy v nie może być „4” bo byłby we tym samym cyklu.

REDUKCJA:



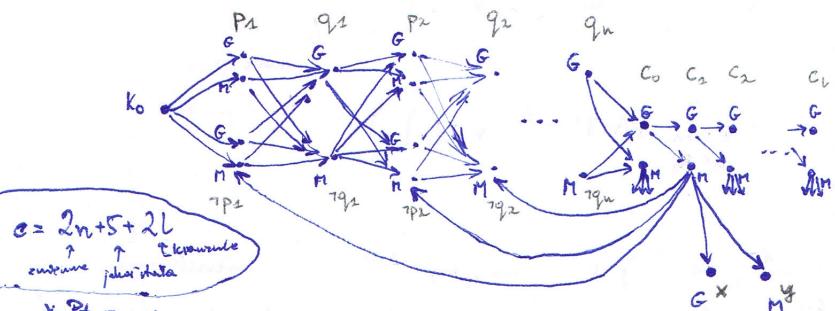
UWAGA: to rozwiązywanie naszego problemu dla G musi w siedzi: \square

bo $\begin{array}{c} 1 \\ \diagup \quad \diagdown \\ 2 \quad 3 \end{array}$ to poprawne kolorowanie, a COL4-CYCLE będzie siedzi, bo 2, który

PSPACE - zupełne.

1. PSPACE - trudne.

$TQBF \leq_p Z7$, Dla φ mamy $\exists p_1 \forall q_1 \dots \varphi$; skonstruujemy planusek i "c" t.j. $\text{Pty}\text{-wygrywane}$ wtedy gdy F prawdziwa.



- * $\text{Pty}\text{-wybiera}$ wartościowane π_i (przez \exists) (Balbina mała & lub M a $\text{Pty}\text{-wygrywa}$)
- * Balbina wybiera wartościowane q_i ($\text{Pty}\text{-ma}$ tylko jedna możliwość) (V)
- * Balbina wybiera klawisz (mowa G jest nowy czas, a potem M).
- * Z kolejnej klawisza są 3 strzałki do loteków i 2 do x, y etykieta na których G, M - $\text{Pty}\text{-nie wygrywa}$, ale nie może ^{jeżeli} do x ani y.
- * Jeśli był w jednym miejscu to to oznacza FALSE.
- * $\text{FFF} \Leftrightarrow \text{Pty}\text{-wygrywa}$ i formuła F jest nieprawdziwa.
(dla π_i strzałki od klawisza do $\pi_i M$ (a nie $\pi_i G$) więc Balbina dla π_i zawsze wybiera $\pi_i M \rightarrow$ mowa "mowa" a $\text{Pty}\text{-wygrywa}$ $\pi_i M$ lub $\neg \pi_i M$)
- * Jeśli chceź jasien literał TRUE to $\text{Pty}\text{-wygrywa}$ bo $\pi_i \rightarrow q_{i+1}$ jest tylko jeden wybór, bo q_{i+1} jest wartościowany, $q_i \rightarrow p_{i+1} \rightarrow q_{i+2}$ tak samo.

2. Z7 G PSPACE

```

PTr5Wins(v, L) ← lista odwiedzonych
  IF NO NEIGHBOURS OR EVERY IS VISITED
    RETURN FALSE // Balbina wygrywa
  FOREACH u | v->u
    IF EVER IS G OR EVER IS M
      RETURN TRUE // Pty - wygrywa
  END
  AND ( V PTr5Wins(u, v:=L) ,
        u,v->u NEG
        V PTr5Wins(u, v:=L) )
        u,v->u NEG M
    ) // niezdobrze, czy Balbina
      // powie "mowa", czy "jedna"
      // to Pty - mode wygrac.
  )
  
```

→ PSPACE, bo cały czas pamiętaj tylko laste wezły do których
berą powtarzanie, czyli klasowe powtarzanie od dawnych wygranych (V)

$\text{TOTAL}_{\text{jump}}$ jest PSPACE-zupełny.

1 $\text{TOTAL}_{\text{jump}} \in \text{PSPACE}$

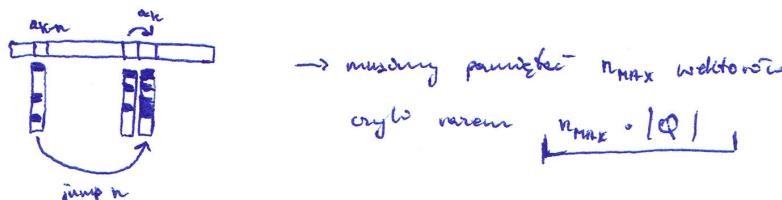
$$\text{TOTAL}_{\text{jump}} \in \text{PSPACE} \Leftrightarrow \overline{\text{TOTAL}_{\text{jump}}} \in \text{PSPACE} \Leftrightarrow \overline{\text{TOTAL}_{\text{jump}}} \in \text{NPSPACE}$$

Pokazując niezdecydowalność algorytmu zamykającego słowo, które jest nieakceptowane przez dany automat.

Dla normalnego automatu mówiącym wektor mówiących stanów.



Teraz może być taki, że po przeczytaniu k znaków mówiących być w jakimś stanie odpowiadającym numer skoku $k-n \rightarrow k$.



* $S \leftarrow \{q_0\}$, $N_{\max} = \max$ po n z instrukcją:

* wykonaj $2^{|Q| \cdot n_{\max}}$ razy

- $a \leftarrow$ zgadnij literkę

- $S' \leftarrow$ wszystkie możliwe stanły z S_1, S_2, \dots, S_n

* - jeśli $S' \cap F = \emptyset$ ret TRUE // znaleziono słowo, które NIE MA próbującego akceptującego

wpp: przesuń wektory (shift o 1)

* ret FALSE

Poprawione → jeśli $> 2^{|Q| \cdot n_{\max}}$ to które konfiguracje stanów się powtarzają.

czyli można wybrać słowo mającym numer N_{\max} zapisanego konsekwentnie, więc OK.

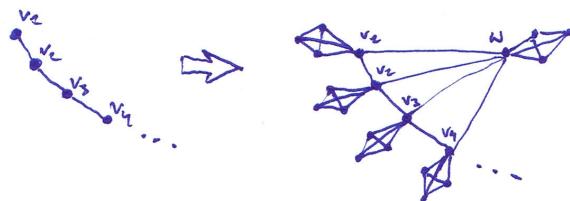
2 $\text{TOTAL} \leq_p \text{TOTAL}_{\text{jump}}$

reprezentacja na NFA i każdej instrukcji zapasującej jako $\langle q, a; q', 1 \rangle$

↑
znak do przedłużonej NFA.

wtedy ten automat ze skończoną ilością reprezentacji może być równoważny reprezentacji total $\Leftrightarrow \text{NFA}_{\text{jump}} \text{ total}$.

$G \in 3COL \Leftrightarrow G'$ da się pokolorować 6 kolorami t.j. że każdy wierzchołek sąsiaduje z wierzchołkami o 3 różnych kolorach.



Lemat: Jesteśmy mamy klocek K_4 to musimy mieć 4 różnych kolorów, aby każdy miał sąsiadów o 3 różnych kolorach.

(\Rightarrow) Jesteś G jest 3-kolorowalny, $\exists w^1$ ma sąsiadów trzech kolorów, a samo w^1 jest czwartego koloru \Rightarrow każdy klocek dla niej dobrze pokolorowac.

(\Leftarrow) G' jest, oto w^1 cyklu $4^1 2^1 3^1$. Skoro taki jest, to wszystkie sąsiadów w^1 , czyli v_i z ogólnego prefiku G , są pokolorowane kolorem 1 lub 2 lub 3.

4 v_i może mieć kolor c_1 , wszystkie sąsiadów wierzchołek w tej kloce muszą być różne, ale skoro w^1 jest v_i , to jeden musi być 4. Porostade dwa z c_2, c_3 gdzie $c_i \in \{1, 2, 3\}$, ale $c_i \neq c_1$ cyklu wierzchołek v_i musi mieć w swojej klate sąsiadów koloru 4, c_2, c_3 , cyklu sąsiadów = ogólnego prefiku mają koloru c_2 lub $c_3 \neq c_1$ cyklu nie może koresponduje, że sąsiad ten jest tego samego koloru $\Rightarrow G$ jest 3-kolorowalny. \square

Maszyny pamiętają n_M znaków do tyłu, gdzie n_M to max n z instrukcji danego automatu. Zapomniane znaków, więc zajmuje to więcej pamięci względem wejścia.

$NEMPTT_{RETRO} \in PSPACE \Leftrightarrow NEMPTT_{RETRO} \in NPSPACE$
t.Tw. Savitcha

Czy istnieje słowo, które automat akceptuje?

Niedeterministyczne je zjadlizmy.

- * $S \leftarrow \{q_0\}$ $\max n$ z instrukcjami // możliwe,że potrzeba $2^{1^{Q1}} \cdot 2^{1^{Q1}} \cdots = 2^{n_M 1^{Q1}}$
 $\rightarrow n_M$ wektorów każdy na $2^{1^{Q1}}$ sposobem
- * wykonuj $2^{1^{Q1}} \cdot n_M$ razy
 - a \leftarrow zjadlizmy literkę
 - $S' \leftarrow$ wszystkie możliwe stanów z S i projektywanymi wyjściami pamięć co było zmianów tennu.
 - jeśli $S' \cap F \neq \emptyset$ ret true.
 - else $S \leftarrow S'$ //wybiór niedeterministyczny tylko jeden stan
- * ret false

$\rightarrow G \in PSPACE$ bo w pamięta many tylko stan i n_{MAX} komórek pamięci, a n_M tylko unikowe.

$\rightarrow 2^{1^{Q1}} \cdot n_M$, bo many $2^{1^{Q1}}$ różnych podzbiorów i n_M mogą wygenerować różne nowe stanów. Jesteś wie mniej to się zapełnia, cyklu false.

1° $k = 2$, COLQ^k & PTIME

Jest $k=2$ to klatek K_2 to krawędzi, czyli krawędź krawędzi ma być pokolorowana różnymi kolorami.

Ustalenie w i oznaczeniu BFS kolejność wierzchołków nie ma znaczenia czarny/biały \rightarrow jeśli brzeg konflikt to FALSE, jeśli nie to TRUE.

2° $k \geq 3$, COLQ^k & NP-zupełny

1 $\text{COLQ}^k G \text{ NP} \rightarrow$ jeśli dany menu pokolorowanie, to sprawdzenie, czy jest dobry (czy krawędź klatek K_k jest dobrze).

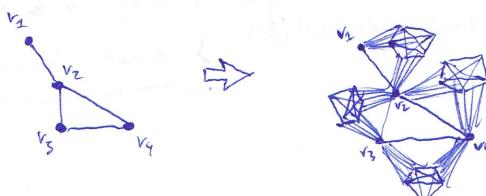
2 $\text{3COL} \leq_p \text{COLQ}^k$

Dla menu graf G , tworzący G' t.o. krawędź klatek K_k dla się pokolorować k kolorami $\Leftrightarrow G$ skolorowalny.

FAKT 1 Jeśli klatek wychodzi w K'_k , to nie da się pokolorować k różnych kolorami.

FAKT 2 Jeśli G skolorowalny to nie zawsze zawsze klatek ≥ 4 , bo by się nie dało go pokolorować.

Rozumieć: dla krawędzi krawędzi $\{u, v\}$ tworzących klatek $K_{k=2}$ i tworzących ją $u \neq v$ tworząc klatek K_k .



Redukcja jest arytmetyczna względem liczby krawędzi (wysz rownolegle i wzajemnie, wiec o wypadek).

(\Rightarrow) G jest 3kolorowalny, czyli $V(G)$ ma i v moga robić kolor

\Rightarrow brzegi w stanie przydzielili k-2 różnych kolorów

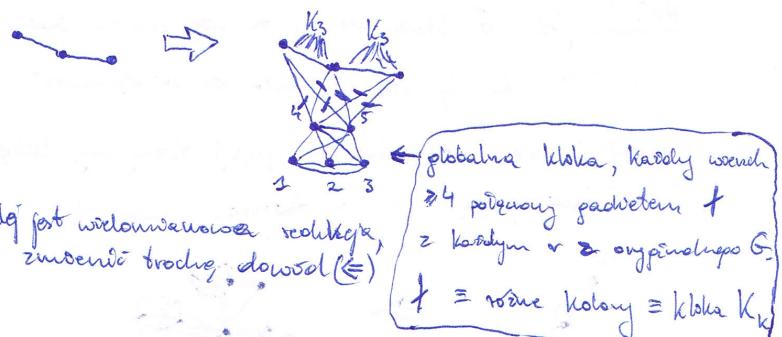
klatek $K_{k=2}$, zatem krawędź klatek K_k będzie ok

(nie będzie problemu z tym, że klatek $K_{k=2}$ się ze sobą łączy i powtarza, co w G nie ma K_k ani nie ma tego skoro jest 3kolorowalny)

(\Leftarrow) Zał, że G nie jest 3kolorowalny, czyli dla krawędzi kolorowanych $c: V \rightarrow \{1, 2, 3\}$ istnieje krawędź $\{u, v\}$, że $c(u) = c(v)$.

Wtedy oznacza, że klatek K_k nie będzie miał pokolorowania k różnych kolorami.

UNAGA: Tu brakuje wymuszonego kolorowania oznaczonych wierszach tylko trzema kolorami !!! Dla $k \geq 4$ musimy to zrobić.



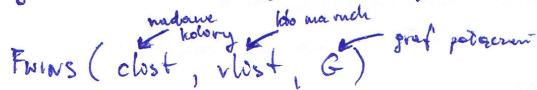
To dalej jest wiedomie, że redukcja jest prawidłowa + z krawędzią $u =$ oznaczenia G .

+ = rowne krawędź = klatek K_k

PSPACE - zupełne

1) Dajwersyty & PSPACE

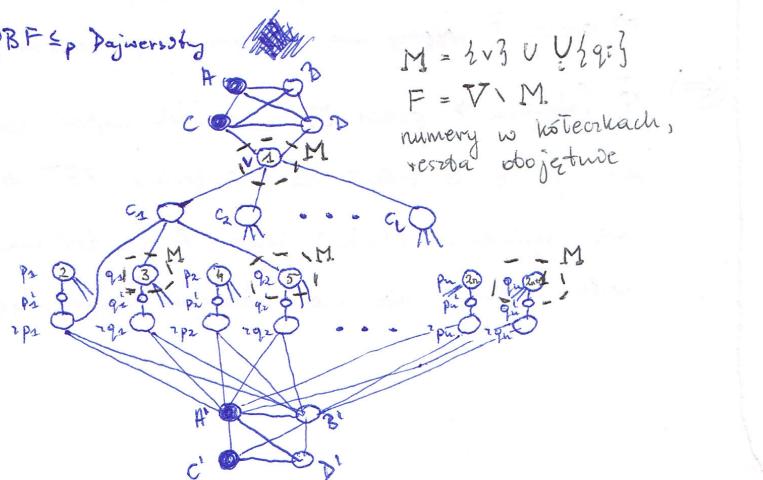
Algorytm w wersji bez kolejnością pionową:

match $vlost$ with| EMPTY → Check Colors BFS ($G, clost$)| $f :: vlost^l \rightarrow$

$\text{FWINS}(0 :: clost, vlost^l, G)$ // F może zwieść gracza
 $\vee \text{FWINS}(1 :: clost, vlost^l, G)$ // ruch do wygrania

| $m :: vlost^l \rightarrow$

$\text{FWINS}(0 :: clost, vlost^l, G)$ // niepotrzebne col ruchu
 $\wedge \text{FWINS}(1 :: clost, vlost^l, G)$ // gracz M , F wygrana

2) $TQBF \leq_p$ Dajwersyty

* Grajemy ABCD i A'B'C'D' dla tego polecam ruchy v oraz wszystkie pi, ~pi, qi, ~qi muszą sąsiadów różnych kolorów.

* pi i ~pi oraz qi i ~qi muszą być różnego koloru, bo inaczej pi lub qi nie będzie mógł sąsiadów różnych kolorów.
 → A WIEC MAMY WARTOŚCIOWANIE
 czarny = T, biały = F (lub na odwrót)

* najpierw gracz M wybiera kolor dla v i bedziemy go traktować jako kolor dla FALSE!

* jedyny problem z sąsiadami może kłaść się - wierzchołek ci i to, coż będzie ok zatem od prawdziwości formuły.

(\Rightarrow) Φ prawdziwa \Rightarrow niedozwolone col wybranego dla v i warzącego zaznaczony q_i w żADNEJ klawiszu nie ma FFF \Rightarrow Kandydat ~~wszystkich~~ klawiszy dla ci ma sąsiadek T i ma też sąsiadek F (wierzch v)
 \Rightarrow gracz F wygrywa na strefie wyprawy wyciągowej.

(\Leftarrow) Φ fałszywa \Rightarrow gracz M może tyle wybrać wartościowe dla q_j, że w którejś klawiszu będące FFF \Rightarrow to ci ma sąsiadów Tylko koloru F (v też ma kolor F) więc gracz F nie ma strategii wyprawy wyciągowej.

a) Zatwierdzić, że istnieje taki algorytm ft.

Pokażemy, że wtedy $P \neq NP$ (bo będącym rozwiązywanym problemem 3COL w wielomianowym czasie, który jest NP -zupełny)

1) Zamień G na $4G$:

$$G \Rightarrow G \text{ } G \text{ } G \text{ } G$$

(4 ramię G nawiązujące ze sobą)

- 2) ~~Znajdi~~ Znajduj kolorowanie $k = \sqrt{2}(4G)$
- 3) Policz skor. koloracji (iteracja po krawędziach)
 $\hookrightarrow x$ = karta koloru dla kolorowania k .
- 4) if $x \in \{0, 1, 2, 3\}$
 return TRUE
 else
 return FALSE

Poprawność:

$$1^{\circ} G \in 3\text{COL}$$

$$\kappa(G) = 0$$

$$\kappa(4G) = 0$$

algorytm zwróci kolorowanie

$$\text{t.j. } \kappa(x) \in \{0, 1, 2, 3\}$$

(Nic mniej 4, bo $\kappa(k) \leq 3 + 0$)

$$2^{\circ} G \notin 3\text{COL}$$

$$\kappa(G) \geq 1$$

$$\kappa(4G) \geq 4$$

algorytm zwróci kolorowanie

$$\kappa(x) \geq 4$$

a) $CK_{2k} = CK_2 \in \text{PTIME}$ 

skoro sąż robiącego o 1
to mówiąc to potakowanie
jako parzyste/nieparzyste

czyli pytanie o dwukolorowość grafu, BFS, PTIME ✓.

b) $CK_{2k+1} \in \text{NP}$

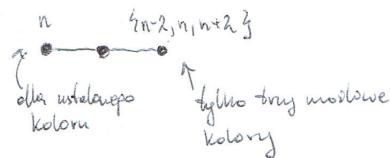
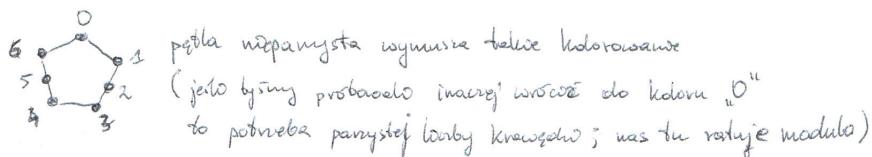
* daje nam kolorowane i sprawdzamy

 $3\text{COL} \leq_p CK_{2k+1}$

* pokazuję to na przykładzie $CK_{\frac{5}{2}}$

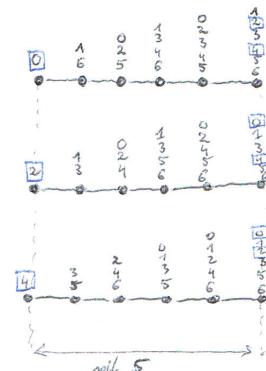
* mówiąc że żadne dwoje sąsiadów:

- 1) wierzch z oryginalnego G są tylko w 3 różnych kolorach
- 2) wierzch podzielone w oryginalnym grafie G są w 5 różnych kolorach.

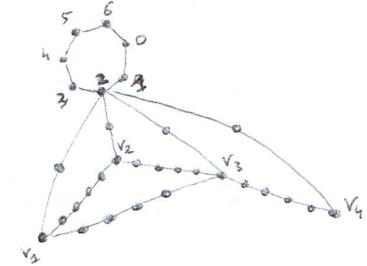
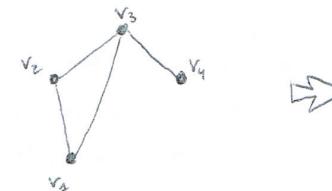


na rysunku po prawej przedstawiono
możliwe kolory wierzchołków począwszy
od 0, 2, 4; widać, że po dodaniu 5
mamy:

$$\begin{aligned} 0 &\rightarrow \{2, 4\} \\ 2 &\rightarrow \{0, 4\} \\ 4 &\rightarrow \{0, 2\} \end{aligned}$$



REDUKCJA:



$$v_1, v_2, v_3, v_4 \in \{0, 2, 4\}$$

oraz ponadto $v_i u, v_j v \in \text{col}(u) + \text{col}(v)$
czyli $G \in \text{3COL} \Leftrightarrow f(G) \in CK_{2k+1}$

CK_{2k+1} jest problemem NP-zupełnym.

Zauważmy, że strategia to funkcja: $\sigma: V^* \rightarrow V$
 $\uparrow \quad \downarrow$
jako się rusza
historia (być może bardziej skrócona)

W tym wypadku mamy ograniczony strategię:

B1	B2	B3	B4	B5	B6
01	02	03	04	05	?

Zauważmy, że to co ma dać Olek w „Ok” zależy od „Bk-1”, „Ok-1”, „Bk”

(Olek chce żeby nie pojawił się niezetyczny kwadrat)

$$\sigma_{OLEK} = k^3 \rightarrow k$$

Czyli Olek musi ustawić tak, aby

B5	B6
05	?

był estetyczny

i żeby

B6	b
?	?

 dla każdego „b” żeby da Bolek mógł postawić „?” tak żeby było estetyczne.

Muszą być jednak ostatnie dwie kolumny i sprawdzić jakieś

$$|R| \cdot |R| \cdot |R| \cdot |R| = |R|^4 \text{ przypadków, czyli wielokrotnie}$$

(Dalsza historia NIE MA wpływu na przestrzeń strategii OLEKA).

Ten problem jest PSPACE-zupełny.

1) OWIN \in PSPACE

$$\text{OWIN-SPACE} \Leftrightarrow \overline{\text{OWIN}} \in \text{PSPACE} \Leftrightarrow \overline{\text{OWIN}} \in \text{NPSPACE}$$

$\forall r_1 r_2$ te kolorowane oki muszą $\exists r_1 r_2$ kolorowane nie oki

zbudujemy NMT akceptującą w $p(|R|)$ pamięci.

→ jeśli istnieje taka r2 to jest krotka nai $2^{|R|}$, bo by się podzielić co.

do $2^{|R|}$ times:

$\epsilon \leftarrow$ neutralny, zadanym kolor
 $S \leftarrow \{ c \in R \mid \begin{array}{|c|c|} \hline a & c \\ \hline b & b \\ \hline \end{array} \text{ jest ok} \} \quad // a, b \in \text{pamięć z poprzedniej iteracji}$
if $S = \emptyset$ return TAK // Olek nie ma ruchu czyli przegra
return NIE // gdy nie trafił w ok, bo zawsze kolorowane oki

2) TOTAL REG \leq_p OWIN

Dajmy nam Σ i autem A , zbudujemy R taki, że A akceptuje wszystkie słowa wtedy Olek ma strategię wygrywającą.

Skoro dowodzę:

$\forall w \exists \pi$ taki, że akceptuje
 \uparrow
ma przebieg akceptujący
dla danej słowa

$\equiv \forall r_1 \exists r_2$ taki, że oki
 \uparrow
to co da Olek
to co da Bolek

A	B	A	A	C	A	...
q0	q5	q11	q2	q3	q5	

\bar{N} - Tadwe kafelki: jeśli w st jest $\langle q_i, a, q_j \rangle$ to

X	a
q_i	q_j

$\in \bar{N}$ dla każdego $x \in \Sigma$.

Deadlock Recovery $\langle G, n \rangle$ - dla sieci uszeregowanych krawędzi i węzłów nie ma żadnego cyklu.

DR jest NP-upędny.

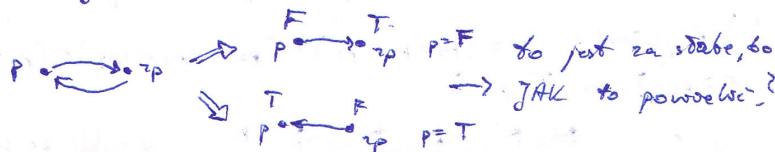
1. DR εNP

→ Mówiąc innym, krawędzie uszeregowane, uszeregowanie i sprowadzenie ich jest ε-up. BFS-em.

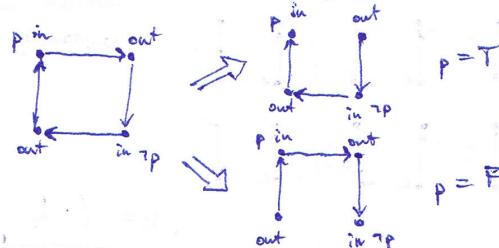
2. DR jest NP-trudny.

ΣSAT \leq_p DR

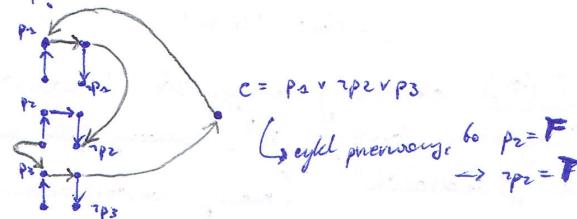
1) Maszyny określone wartościowaniem



bądźmy więc wywiercić:

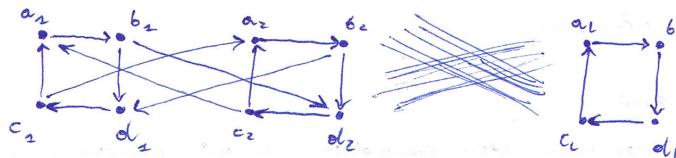


2) Przywołując cykl wtedy gdy przyjmującej jednostce będzie to T.



3) Różne cykle mogą się sobą interferować, więc musimy powodzić wartościowanie, i zadać, aby było tylko same.

GADŻET:

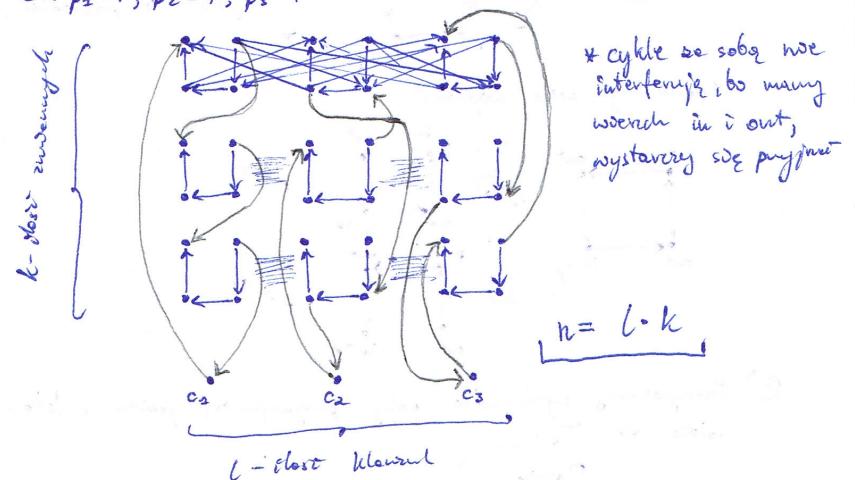


* z każdego ci krawędzi do każdego aj

* z każdego bi krawędzi do każdego dj

→ właściwie jestem mamy „l” krawędzi do usunąć to ALBO usunąć wszystkie $a_i \rightarrow b_i$ ALBO wszystkie $d_i \rightarrow c_i$ aby nie było cyklu, cyku KOPIVYGMF o.

4) Przykład: $y = (p_1 \vee p_2 \vee p_3) \wedge (p_2 \vee \neg p_2 \vee p_3) \wedge (p_3 \vee \neg p_1 \vee p_2)$
 $\sigma: p_1 = T, p_2 = T, p_3 = T$



(\Rightarrow) i (\Leftarrow) dowody określone i wynikające z konstrukcji i wyborem liczby „n”. usunąć tylko jakaś krawędź na „kwadraciku”. (masz wartościowanie).

JF120 EGZ 2021 III zad 8

000...0 \rightsquigarrow 11...1 ?

$\in \text{PSPACE}$ ten pokazany algorytm, który zwraca rozwiązanie
dla duchalkowej parys. (podobnie jak Tw Savitcha)

$\text{PATH}_i(x,y) \equiv$ ory jest swinka $x \rightarrow y$ dla $\leq 2^i$.

\rightarrow Jów istnieje swinka 00...0 \rightsquigarrow 11...1 to jest króciej niż 2^n .

```

 $\text{PATH}_i(x,y)$ 
if  $i=0$ 
    return  $\psi(\vec{x}, \vec{y})$  // bezpośrednia krawędź
else
     $\forall z \in V$ 
        if  $\text{PATH}_{i-1}(x,z)$  and  $\text{PATH}_{i-1}(z,y)$ 
            return true
    return false

```

Poprawność ozywista: $x \xrightarrow{\leq 2^{i-1}} z \xrightarrow{\leq 2^{i-1}} y$

He duchalkowej parys?

$S_i = n + S_{i-1}$ sprawdzony $x \rightarrow z$, potem sprawdzony
 \uparrow i sprawdzony $z \rightarrow y$.
 adres z^n

$$S_n = \frac{n+n+\dots+n}{n} = O(n^2)$$

PSPACE ✓