

# Rozwiązania niektórych z około dwustu łatwych zadań z języków formalnych i złożoności obliczeniowej (i jednego nie aż tak trudnego, jak się o nim mówi)

Numeracja zadań jak w zbiorze z 2017 roku

Wrocław, 28 maja 2017

## 1 Robocze

## 2 Wskazówki

Patrząc perspektywicznie w stronę zbliżających się egzaminów, ta sekcja wydaje mi się ważniejsza i zachęcam do spędzania z nią czasu podczas samodzielnych wieczornych rozmyślań.

### 2.1 Języki i automaty

**Zadanie 63.** *Warto zastanowić się, jak miałby wyglądać niedeterministyczny automat ze stosem rozpoznający taki język. Pomocny może się też okazać pewien lemat.*

#### 2.1.1 Języki rodzinowe

**Zadanie 71.** *Problem sprowadza się do rozpoznawania liter języka  $L^*$ . Gdyby miał istnieć taki język  $L$ , to pewnie musiałby być nieskończony (dlaczego?). Może istnieje jakaś prosta, nieskończona rodzina słów postaci  $a^*ba^*$ , którą automat ze stosem może łatwo rozpoznawać, ale taki bez stosu będzie miał trudniej? Formalnie, jak to bywa, warto używać lematów.*

#### 2.1.2 Konfluentność

**Zadanie 74.** *1) To pytanie jest (nieprzypadkowo) podobne (ale nie identyczne) do pytania: Czy każdy automat deterministyczny można zsynchronizować? (zadania 37-39).*

*2) Warto zauważyć, że w definicji konfluentności jest równoważność. Może*

zatem być tak, że obie jej strony będą zawsze fałszywe, co oznacza, że chcemy “na dobre” zepsuć słowo. Może są języki nieregularne, które łatwo zepsuć?

**Zadanie 75.** To zadanie jest podobne do zadania 38.

**Zadanie 76.** Może chodzić o język, który nie jest tak łatwo zepsuć (ma długie słowa psujące).

## 2.2 Obliczalność

**Zadanie 92.** Jak coś ma nie być r.e. to na pewno chodzi o redukcję z  $\overline{\mathbb{K}}$ . Delikatna modyfikacja 89.

**Zadanie 99.** i) Jak coś ma nie być r.e. to na pewno chodzi o redukcję z  $\overline{\mathbb{K}}$ . Delikatna modyfikacja 89.

ii) Czy  $\Phi_m \neq \Phi_n \iff \exists k \Phi_m(k) \neq \Phi_n(k)$ ?

**Zadanie 100.** To zadanie jest podobne do zadania 102. Zaś z zadania 82 wiemy, czym jest  $\Sigma_1$ .

**Zadanie 101.** Trzeba wprost zdefiniować redukcję z  $B$  do  $A$ .

**Zadanie 102.** a. założenie o nietrywialności zadania b. implikuje odpowiedź w zadaniu a.. Pewna użyteczna redukcja i w tym zadaniu okaże się pomocna. b. formalne sformułowanie tego, co to znaczy że  $n \in B$ , używająca 3 zmiennych, wprost prowadzi do co-r.e. zbioru  $A$ .

**Zadanie 103.** Należałoby się zastanowić, co wspólnego mają te warunki z monotonicznością  $f$ .

**Zadanie 104.** Wskazówka do implikacji  $2 \Rightarrow 1$  podana w treści jest bardzo dobra. Wskazówka do implikacji odwrotnej: to jest podobne do zadania 97. Trzeba powtórzyć tamto rozumowanie, z tym że zamiast 2 zbiorów wziąć przeliczalnie wiele.

**Zadanie 107.** Wskazówka sugeruje, że nie. Dodatkowa byłaby taka, że zbiorów r.e. jest przeliczalnie wiele - bo tyle jest programów, które je rozpoznają. Warto też przypomnieć sobie (jak ktoś nie pamięta), dowód tego, że podzbiorów  $\mathbb{N}$  jest nieprzeliczalnie wiele. Ten argument jest tylko trochę “inteligentniejszy”.

**Zadanie 110.** To jest wgl. ważne zadanie i warto mu poświęcić dłuższą chwilę.

a) Chcemy zakodować Maszynę Turinga (z ustalonym wejściem) jako automat deterministyczny z dwoma stosami.

a<sub>0</sub>) Zamienić MT z ustalonym wejściem na równoważną MT z pustym wejściem.

a<sub>1</sub>) Trzeba się przyjrzeć dokładniej temu, jak Maszyna wygląda, porysować

coś. W szczególności spostrzec, że działa ona bardzo lokalnie. Trochę podobne do zadania 130, może być ono pewną inspiracją.

b) Zamienić stos na licznik.

b<sub>0</sub>) Jak zamienić potencjalnie szeroki wybór sweterków (jak u blondynki) na sweterki tylko dwóch kolorów (jak u blondyna)?

b<sub>1</sub>) Popatrzeć na taki zamieniony stos i zobaczyć, że w rzeczywistości wystarczy nam pamiętać jedną liczbę zamiast jednego stosu.

b<sub>2</sub>) Poświęcić chwilę (!) na względnie dopracowanie wymaganych od stosów operacji w licznikowej implementacji stosów. Przydatne mogą okazać się dwa dodatkowe liczniki.

c) Zachwycić się bogactwem liczb naturalnych - w szczególności można skorzystać ze znanego wszem i wobec kodowania skończonych ciągów liczbowych jako pojedyncze liczby. Np. ciągi 4-elementowe też się da. Zaimplementować wymagane operacje przy użyciu drugiego licznika.

**Zadanie 114.** Mocno podobne do zadania 127 (gramatyk ze znikaniem). Tym razem trzeba zrobić dwie gramatyki, osobną dla słów  $l_i$  i  $r_i$ .

**Zadanie 115.** Warto skorzystać z zadania 114. Udowodnić, że  $(L_G)^c$  dla CFG  $G$  z zad. 114 jest CFL. Porachować, przypomnieć sobie prawa de Morgana.

**Zadanie 116.** Warto skorzystać z zadania 115.

**Zadanie 118.** Czy da się, być może dodając specyficzne zamiany, zasymulować pełny semiproces Thuego, ograniczając się w **istotnych** zamianach tylko do zamiany początku pierwszego słowa na koniec drugiego?

**Zadanie 120.** Warto rozważyć jakiś nierozstrzygalny problem, który można zakodować w kafelkach o skończonej liczbie kolorów, a poprawność kodowania sprawdzać bardzo lokalnie (to będzie odpowiadało poprawnym czwórkom kolorów). Trochę podobne do rozwiązania z Sipsera zadania 115.

**Zadanie 121.** W obu przypadkach redukcja musi zmodyfikować układ równań tak, żeby:

a) zwiększyć liczbę możliwych rozwiązań. Wsk: zapisać dowolną zmienną całkowitą jako wielomian o wsp. całkowitych oraz zmiennych naturalnych.

b) zmniejszyć liczbę rozwiązań. Wsk: tutaj warto skorzystać z wskazówki z treści zadania o zapisie liczby naturalnej jako 4 kwadratów liczb całkowitych.

**Zadanie 122.** Podobne do zadania 129.

## 2.3 Złożoność obliczeniowa.

**Zadanie 143.** Oczywiście trzeba zamienić pojedynczą 5cio literałową klauzulę na równoważną koniunkcję klauzul 3-literaowych (być może używając dodatkowych zmiennych).

**Zadanie 144.** Wskazówka z treści zadania: *To nie jest trudne. Idea jest podobna jak przy dowodzie faktu, że  $3SAT \leq_p 3COL$ , który był na wykładzie. Tylko łatwiej jest dobra.* Dodatkowo, warto zauważyć, że to my ustalamy liczbę agentów. Jaka byłaby naturalna? Kim w formule CNF ma być agent i czym jest to, czego strzeże?

**Zadanie 145.** Wskazówka z treści zadania jest dobra. Oczywiście prościej jest zamienić graf nieskierowany na równoważny graf skierowany.

**Zadanie 148.** Za każde wystąpienie zmiennej  $x$  postawić nową zmienną  $x_i$ . Wykorzystać pozostałe 2 dozwolone wystąpienia zmiennych  $x_i$  na napisanie, że  $x_1 \Leftrightarrow x_2 \Leftrightarrow \dots \Leftrightarrow x_k$  (w postaci CNF).

**Zadanie 150.** Oczywiście redukcja z Hamiltona. Wagi w grafie pełnym powinny wiązać się z tym, które krawędzie są w oryginalnym grafie, a których nie ma.

**Zadanie 151.** Jak w zadaniu 150, ale chcemy, żeby krawędzie niebędące w oryginalnym grafie były bardzo ciężkie. Dlaczego?

**Zadanie 152.** To zadanie ma związek z MST.

**Zadanie 176.** Kluczowe jest to, że pytanie o  $\sigma(G) = 1$  to przecież pytanie o istnienie cyklu Hamiltona.

**Zadanie 177.** Wynika z zadania 178.

**Zadanie 178.** Powinno się udać połączyć  $c(+1?)$  kopii grafu  $G$  w ten sposób, żeby był w tym cykl Hamiltona wtedy i tylko wtedy, gdy w  $G$  jest, a ponadto, żeby  $\sigma(G') > c$  gdy w  $G$  nie ma cyklu Hamiltona.

**Zadanie 192.** Ten zbiór jest oczywiście świadkiem. Najtrudniej pokazać, że nie jest w  $P$  - pokaż, jak obliczyć  $f^{-1}(x)$  w czasie wielomianowym, gdyby jednak był.

**Zadanie 193.** Znajdź zbiór punktów, z których w jednym ruchu może wygrać gracz  $P$ . Czy to jest żmudne?

**Zadanie 194.** Najpierw przypomnij, jak z wyrażenia regularnego buduje się NDFA. Może zamiast sprawdzania totalności tego automatu, łatwiej sprawdzić jego nietotalność (dokładniej, pokaż że jest ona w NPSPACE).

**Zadanie 195.** Trzeba opisać, jak przechodzić wykładnicze drzewo wartościowań mając do dyspozycji pamięć wielomianową. Plemienne liczenie liczb rozwiązań powinno zadziałać (plemię liczy 0/1/2 lub więcej).

Dla wersji  $\{0,1\}$  - czy nie można zamienić formuł arytmetyki modulo 2 na spójniki logiczne? Ponadto, czy łatwo się sprawdza, że rozwiązanie jest jedyne?

**Zadanie 196.** *Po pierwsze: czy to jest problem SAT z trzema kwantyfikatorami uniwersalnymi, dla wszystkich możliwości rozłożenia tych kwantyfikatorów (i dlaczego nie)? A skoro nie, to jak zapisać **zdaniem** problem Patroklesa, jaka jest długość tego zdania i ile występuje w nim kwantyfikatorów uniwersalnych?*

*Po drugie: co to znaczy na nasze, że Mojra może wybierać wszystko, co chce, a formuła i tak będzie prawdziwa (jest mądre słowo na takie formuły).*

**Zadanie 197.** *Spełnienie taiej formuły jest dla Patroklesa co najmniej tak trudne, jak spełnienie odpowiedniego QBF-a - stąd naturalna hipoteza.*

**Zadanie 212.** *Zasymuluj podany na wykładzie dowód PSPACE-zupełności Gry w geografę. Określ  $S$  w naturalny sposób. Wystarczy jeden wierzchołek w zbiorze  $T$ .*

**Zadanie 213.** *Coś jak rozwiązywanie QBF-ów z ograniczoną liczbą kwantyfikatorów uniwersalnych (patrzac z punktu widzenia Złego Przewodnika). Czy Złemu nie wystarczy rozwiązać zadania dla wszystkich możliwości naszych ruchów (których jest wielomianowo ( $n^7$ ) wiele)?*

**Zadanie 214.** *Wystarczy pokazać dla  $|S| = 0, |T| = 1$ , czyli  $k = 1$ . Podobne do zadanie 214'.*

**Zadanie 214'.** *Pokaż, że problem Gry w Wesole Zwiedzanie ( $S = T = V$ ) nie jest w PTIME (zakładamy, że  $NP \neq PTIME$ ). Zwiedzanie jest Wesole, bo Zły Przewodnik nie może nam zaszkodzić.*

### 3 Szkice rozwiązań

#### 3.1 Języki i automaty

##### 3.1.1 Synchronizacja automatów częściowych

**Zadanie 40.**

Odp: NIE.

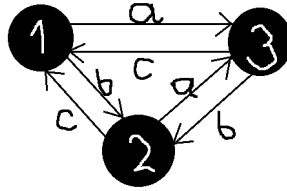
**Zadanie 41.**

W obydwu podpunktach wystarczy zbadać funkcję

$$F : 2^Q \times \Sigma \longrightarrow 2^Q$$

$$F(S, a) = \{\delta(q, a) : q \in S\}.$$

Oczywiście  $s \in \text{csync}(S) \iff |\hat{F}(S, s)| = 1$ , gdy zdefiniujemy  $\hat{F}$  w naturalny sposób. Ponadto  $1 \leq |F(A, a)| \leq |A|$ , zatem  $1 \leq |\hat{F}(S, p)| \leq |S|$  dla



Rysunek 1:  $csync(\{1, 2\}) \supset \{a\}$ ;  $csync(\{1, 3\}) \supset \{b\}$ ;  $csync(\{1, 2\}) \supset \{c\}$ .  
Natomiast  $csync(Q) = \emptyset$ .

dowolnego prefiksu  $p$  słowa  $s$ , czyli  $\hat{F}(S, p)$  może przyjmować co najwyżej  $\sum_{k=1}^{|S|} \binom{|Q|}{k}$  różnych wartości. Oznaczmy tę liczbę jako  $M$ .

Dla  $|s| > M$  istnieją prefiksy  $p_1$  i  $p_2$  słowa  $s = p_1 s_1 = p_2 s_2$ ,  $|p_1| < |p_2|$ , takie że  $\hat{F}(S, s_1) = \hat{F}(S, s_2)$  (Zasada Szufladkowa). Wtedy oczywiście  $\hat{F}(S, s) = \hat{F}(S, p_1 s_2)$ , przy czym  $|p_1 s_2| < |s|$ .

W związku z powyższym  $csync(Q) \neq \emptyset \iff \exists s \in S |s| \leq M$ .

Dokładne odpowiedzi wynikają z tego wprost, po podstawieniu za  $S$  a) dowolnego trzelementowego zbioru stanów b)  $Q$ .

#### Zadanie 42.

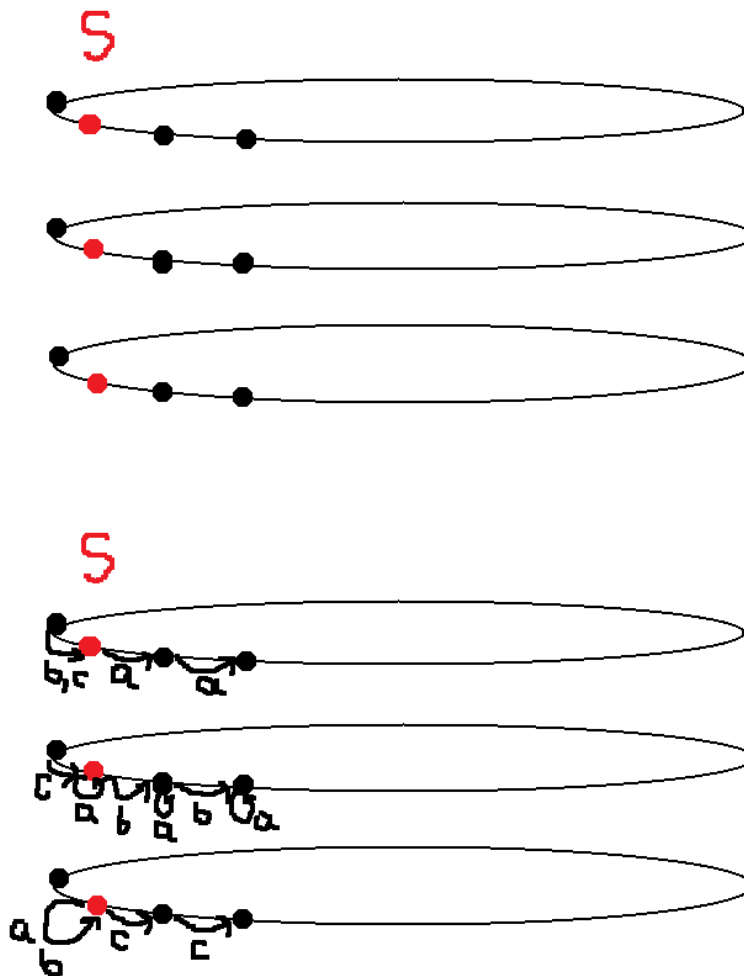
Podpunkt M można rozwiązać w naturalny sposób. W kolejnych podpunktach warto jednak skorzystać ze wskazówki.

Zbudujemy automat (częściowy) z trzech cykli, ułożonych jeden nad drugim, każdy długości  $m$ . Trzy stany, ułożone jeden nad drugim, będą stanowiły nasz początkowy zbiór  $S$

Naszym celem jest, aby co jedną literę zmieniał się stan na górnym cyklu, co  $m$  liter stan na drugim, a co  $m^2$  na trzecim. Zapenimy też, że synchronizacja będzie mogła nastąpić dopiero po przejściu przez dolny stan całego cyklu (czyli  $m^3$  krokach).

Możemy to wymusić w następujący sposób:

przy czym pętelki z literą  $a$  są przy każdym stanie na drugim dysku, a pętelki z literami  $a, b$  są przy każdym stanie na trzecim dysku.

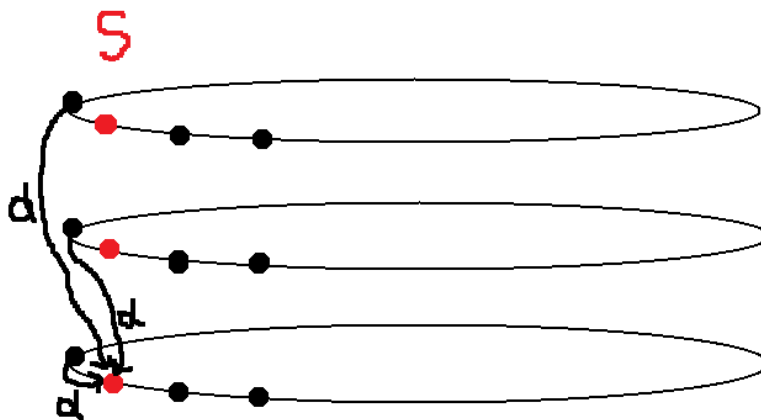


Możliwość synchronizacji zapewniamy przez dodanie przejść z przedostatnich stanów na każdym dysku do pierwszego stanu dolnego dysku. Oznaczenie ich specjalną “literką synchronizacji”, jak  $d$ , zapewni nam, że skorzystać z niej będzie można dopiero gdy dolny stan dojdzie do przedostatniego miejsca na dolnym dysku (co następuje dopiero po  $m^3$  krokach):

Zauważmy teraz, że ten automat (zanim dojedzie do synchronizacji) jednoznacznie wyznacza słowo, dla którego funkcja przejścia jest określona dla wszystkich stanów z  $S$ :

$$s = ((a^{m-1}b)^{m-1}c)^{m-1}d$$

Takie  $s$  synchronizuje  $S$  i nie istnieje żadne krótsze od niego. Nietrudno wyliczyć, że jest ono odpowiednio długie.



### Zadanie 63.

Przypuśćmy nie wprost, że ten język jest bezkontekstowy. Niech  $N$  będzie stałą z lematu o pompowaniu dla tego języka. Wystarczy rozważyć słowo  $0^{2N}1^{2N}0^{2N}1^{2N}$  i pamiętać, że przy podziale z lematu (ozn.  $vxyz$ ) zachodzi  $|wxy| \leq N$ . Odpowiednio cierpliwe rozpatrywanie przypadków (gdzie w oryginalnym słowie łąduje podśłowo  $wxy$ ) prowadzi nas do wniosku, że zawsze można odpowiedni fragment podpompuwać (lub spompować) i uzyskać słowo spoza języka.

### 3.1.2 Języki rodzynekowe

#### Zadanie 71.

Rozważmy  $L = \{a^nba^n : n \in \mathbb{N}\}$ . Łatwo sprawdzić, że nawet deterministyczny automat ze stosem daje sobie radę z językiem  $L^*$ , bo rozpoznawanie liter tego języka (czyli słów języka  $L$ ) jest bardzo łatwe.

Z lematu o pompowaniu bardzo łatwo pokazać, że  $L^*$  nie jest regularny ( $L$  też nie jest).

### 3.1.3 Konfluentność

#### Zadanie 74.

- 1)  $\{0^n : k|n\}$  dla  $k > 1$  jest przykładem na regularny język niekonfluentny.
- 2)  $\{0^n1^n : n \in \mathbb{N}\}$  jest przykładem na nieregularny język konfluentny. Uniwersalnym słowem  $x$  dla tego języka jest 10.

#### Zadanie 75.



Rozwiązanie tego zadania jest identyczne do rozwiązania zadania 38.

#### Zadanie 76.

Język dobrych nawiasowań jest dobrym przykładem. Dowód przebiega nie wprost, a jeśli oznaczymy przez  $c$  stałą wynikającą z założenia nie wprost, to słowa  $s_1 = )^{c+1}$  i  $s_2 = (^{c+1}$  są kontrprzykładami, gdyż trzeba zepsuć  $s_1$  aby je ukonfluentnić, a nie można tego zrobić słowem długości mniejszej niż  $c + 2$ .

#### 3.1.4 Transducery

##### Zadanie 77.

Podpunkt 1: definiujemy  $\sigma_{Mealy} = \sigma_{Moore} \circ \delta$ . Reszta zostaje.

Podpunkt 2: definiujemy (dla transducera Mealy'ego  $\langle \Sigma, \Sigma_1, Q, q_0, \delta, \sigma_{Mealy} \rangle$ )

1.  $Q' = Q \times \Sigma \cup q'_0$ . Stan  $(q, a)$  = stan do którego doszlibyśmy w starym automacie ze stanu  $q$  wczytując literę  $a$ . Stan  $q'_0$  - dodatkowy stan początkowy.
2.  $\delta'((q, a), b) = (\delta(q, a), b)$  dla pary  $(q, a) \in Q \times \Sigma$   
 $\delta'(q'_0, a) = (q_0, a)$
3.  $\sigma_{Moore}((q, a)) = \sigma_{Mealy}(q, a)$   
 $\sigma_{Moore}(q'_0) = \varepsilon$

Otrzymujemy transducer Moore'a  $\langle \Sigma, \Sigma_1, Q', q'_0, \delta', \sigma_{Moore} \rangle$  równoważny z pierwotnym t. Mealy'ego.

Dowód w obu przypadkach zapewne angażuje Zasadę Indukcji Matematycznej względem długości słowa.

##### Zadanie 78.

bso. (77) zajmijmy się transducerem Mealy'ego  $\langle \Sigma, \Sigma_1, Q, q_0, \delta, \sigma_{Mealy} \rangle$ , który świadczy że  $A \leq_{reg} B$ . Niech przy okazji  $A_B = \langle \Sigma_1, Q^B, q_0^B, F^B, \delta^B \rangle$  będzie DFA rozpoznającym  $B$ . Definiujemy  $\delta'(q, a) = \widehat{\delta^B}(q, \sigma_{Mealy}(q, a))$ . Udajemy, że jesteśmy słowem z języka  $B$  i chodzimy po automacie  $A_B$ . Wtedy  $\langle \Sigma, Q^B, q_0^B, F^B, \delta' \rangle$  jest DFA rozpoznającym  $A$  (d-d. indukcyjny względem długości słowa).

##### Zadanie 79.

Definiujemy transducer Mealy'ego  $T_{Mealy}$ :

1.  $\Sigma = \{1, 2, 3, \dots, n\}$

2.  $Q = \Sigma$

3.  $q_0 = 1$

4.  $\delta(q, a) = a$

5.  $\Sigma_1 = Q \times \Sigma$

6.  $\sigma_{Mealy} = Id$

Niech  $T_{Moore} = \langle \Sigma, \Sigma_1, Q', q'_0, \delta', \sigma_{Moore} \rangle$  będzie transducerem Moore'a równoważnym z  $T_{Mealy}$ .

**Obserwacja 1.** Możemy założyć, że każdy stan z  $Q'$  jest osiągany przez DFA stowarzyszony z  $T_{Moore}$ . W przeciwnym razie możemy usunąć te stany, a powstały  $T'_{Moore}$  wciąż będzie równoważny z  $T_{Mealy}$ .

**Obserwacja 2.**  $s \in Im(\sigma_{Mealy}) \Rightarrow |s| = 1$ . Zatem  $s \in Im(\sigma_{Moore}) \Rightarrow |s| = 1$ .

Gdyby było  $|Q'| < n^2$ , to  $Im(\sigma_{Mealy}) \not\subseteq Im(\sigma_{Moore})$ . Niech  $s \in Im(\sigma_{Mealy}) \setminus Im(\sigma_{Moore})$ .  $s = (k, l)$  dla pewnych  $k, l \in \Sigma$ . Rozważmy słowo  $t = kl$ . Wtedy  $f_{T_{Mealy}}(t) = \sigma_{Mealy}(1, k)\sigma_{Mealy}(k, l) = (1, k)(k, l)$ . Załóżmy nie wprost, że  $f_{T_{Moore}}(t) = f_{T_{Mealy}}(t)$ . Jest to równoważne (obs. 2) z tym, że  $\sigma_{Moore}(\delta'(q'_0, k)) = \sigma_{Mealy}(1, k)$  oraz  $\sigma_{Moore}(\widehat{\delta'}(q'_0, kl)) = \sigma_{Mealy}(k, l) = (k, l)$ . Druga równość stoi w jawnej sprzeczności z naszym założeniem, że  $(k, l) \notin Im(\sigma_{Moore})$ .

### Zadanie 80.

Zdaje się, że świadczy o tym następujący transducer Mealy'ego:

1.  $\Sigma = \{ (, ), [, ], \langle, \rangle \}$

2.  $Q = \{q_0\}$

3.  $q_0 = q_0$

4.  $\delta \equiv q_0$

5.  $\Sigma_1 = \{ (, ), [, ] \}$

6.  $\sigma_{Mealy}(q_0, (/)) = ((/))$   
 $\sigma_{Mealy}(q_0, [/]) = [[/]]$   
 $\sigma_{Mealy}(q_0, \langle / \rangle) = [\langle / \rangle]$

Dowód pozostawiamy Czytelnikowi jako ćwiczenie.

### 3.2 Obliczalność

#### Zadanie 89.

Niech  $B = \{n : \text{Dom}(\Phi_n) = \mathbb{N}\}$ . Robimy redukcję  $f_{89}$  z  $\overline{\mathbb{K}}$ .  
Definiujemy  $f_{89}(n)$  jako numer następującego programu:

```
wczytaj k
odpal  $\Phi_n(n)$  na  $k$  kroków
jeżeli się skończył : zapętl się
w p.p. : zwróć 1
```

Sprawdzenie, że zachodzi  $n \in \overline{\mathbb{K}} \iff f_{89}(n) \in B$  pozostawiamy jako proste ćwiczenie.

#### Zadanie 92.

Robimy redukcję  $f$  z  $\overline{\mathbb{K}}$ .  
Definiujemy  $f(n)$  jako numer następującego programu:

```
wczytaj k
jeżeli k jest parzyste : zapętl się
w p.p. :
odpal  $\Phi_n(n)$  na  $k$  kroków
jeżeli się skończył : zapętl się
w p.p. : zwróć 1
```

Oczywiście  $\text{Dom}(\Phi_{f(n)}) = (2\mathbb{N} + \{1\}) \cap \{t \in \mathbb{N} : t < \text{czas wykonania się } \Phi_n(n)\}$ .  
Sprawdzenie, że zachodzi  $n \in \overline{\mathbb{K}} \iff f(n) \in B$  pozostawiamy jako proste ćwiczenie.

#### Zadanie 99.

i) Robimy redukcję  $f$  z  $\overline{\mathbb{K}}$ , pamiętając o redukcji  $f_{89}$ .  
Niech  $c$  będzie numerem następującego programu:

```
wczytaj k
zwróć 1
```

Definiujemy  $f(n) = \langle c, f_{89}(n) \rangle$ . Jest oczywiste (po zrobieniu zadania 89), że zachodzi

$$n \in \overline{\mathbb{K}} \iff f(n) \in T.$$

ii) Nie. Robimy redukcję  $f$  z  $\overline{\mathbb{K}}$ .

Niech  $f_a(n)$  będzie numerem następującego programu:

```
wczytaj k
jeżeli  $k > 1$  : zapętl się
w p.p. :
odpal  $\Phi_n(n)$ 
zwróć 1
```

Niech  $f_b(n)$  będzie numerem następującego programu:

```
wczytaj k
```

jeżeli  $k > 1$  : zapętl się

w p.p. :

zwróć 1

Definiujemy  $f(n) = \langle f_a(n), f_b(n) \rangle$ . Oczywiście zachodzi

$$n \in \overline{\mathbb{K}} \iff f(n) \in \overline{T}.$$

### Zadanie 100.

Oczywiście  $\Pi_0 = \Sigma_0$ . Z zadań 81-82 wynika, że  $\Sigma_1 =$  zbiory rekurencyjnie przeliczalne. Pokażemy, że  $L \notin \Sigma_1$ . Świadczy o tym następująca redukcja  $f$  z  $\overline{\mathbb{K}}$ : definiujemy  $f(n)$  jako numer programu:

wczytaj  $k$

jeżeli  $k < n$  : zwróć 1

w p.p. odpal  $\Phi_n(n)$

zwróć 1

Widzimy, że gdy  $n \in \overline{\mathbb{K}}$  to  $\text{Dom}(\Phi_{f(n)}) = \{1, 2, \dots, n\}$ , w przeciwnym razie  $\text{Dom}(\Phi_{f(n)}) = \mathbb{N}$ .

Następnie pokażemy, że  $L \in \Sigma_2$ . W tym celu zdefiniujemy zbiory

$$B_0 = \{ (n, m) : (\exists l < \pi_1(f^{-1}(m)) \ \Phi_n(l) \text{ zakończy się po } \pi_2(f^{-1}(m)) \text{ krokach}) \wedge \\ (\forall l \geq \pi_1(f^{-1}(m)) \ \Phi_n(l) \text{ nie kończy się}) \}$$

$$B = f(B_0)$$

gdzie  $f$  jest naszą ustaloną, obliczalną bijekcją  $\mathbb{N}^2 \rightarrow \mathbb{N}$ , natomiast  $\pi_i$  jest rzutem na  $i$ -tą współrzędną.

Oczywiście  $L$  jest rzutem zbioru  $B_0$ , zatem żeby uzasadnić, że  $L \in \Sigma_2$ , wystarczy pokazać, że jest on  $B \in \Pi_1$ . Dowód tego faktu pozostawiamy jako ćwiczenie.

### Zadanie 101.

Definiujemy  $f^{-1}$  w następujący sposób:

wczytaj  $n$

Dla  $k = 1, 2, 3, \dots$  :

jeżeli  $f(k) = n$  : zwróć  $k$

Oczywiście  $f^{-1}$  jest całkowita, bo  $f$  jest "na". Łatwo sprawdzić, że zachodzi również  $f \circ f^{-1} = \text{Id}$  (a czy  $f^{-1} \circ f = \text{Id}$ ?).

Pokażemy, że  $f^{-1}$  jest redukcją z  $B$  do  $A$  :

$$n \in B \iff \text{Id}(n) \in B \iff f(f^{-1}(n)) \in B \iff f^{-1}(n) \in A,$$

przy czym ostatnia równoważność wynika z tego, że  $f$  jest redukcją z  $A$  do  $B$ . Oczywiście powyższe oznacza doładnie to, że  $f^{-1}$  jest redukcją z  $B$  do  $A$ .

**Zadanie 102.**

a) Nie. Wystarczy pokazać redukcję z  $\mathbb{K}$  (dlaczego?). O dziwo, jest to dokładnie  $f_{89}$ .

b) Definiujemy  $A = \{(n, k, t) : \Phi_n(i) \text{ kończy się po co najwyżej } t \text{ krokach dla } i = 1, 2, \dots, k \text{ oraz nie kończy się dla } i > k\}$ . Ten zbiór jest dobry, a sprawdzenie pozostawiamy jako ćwiczenie.

**Zadanie 103.**

a) Tak. Bo istnieje takie duże  $N$ , że na zbiorze  $\{N, N+1, \dots\}$   $f$  jest nie-malejąca. Z zadania 86 wynika więc, że  $f(\{N, N+1, \dots\})$  jest rekurencyjny. Zatem  $f(\mathbb{N}) = f(\{1, 2, \dots, N-1\}) \cup f(\{N, N+1, \dots\})$  jest rekurencyjny, jako suma dwóch zbiorów rekurencyjnych.

b) Nie. Niech  $A = \{1\}$ . Niech  $g$  będzie całkowitą funkcją rekurencyjną, taką że  $g(\mathbb{N}) = \mathbb{K}$  (skądinąd wiemy, że taka istnieje - np. 87). Rozważmy następującą funkcję  $f$ :

wczytaj  $n$

jeżeli  $n$  jest parzyste : zwróć  $g(\frac{n}{2})$

w p.p. : zwróć 1

$f$  spełnia warunek z zadania, ale  $f(\mathbb{N}) = \mathbb{K} \cup \{1\}$ , a to nie jest zbiór rekurencyjny.

**Zadanie 104.**

$1 \Rightarrow 2$ :

Niech  $\mathbb{D} = \{A_i : i \in \mathbb{N}\}$ . Niech  $\phi : \mathbb{N}^2 \longrightarrow \mathbb{N}$  będzie dowolną ustaloną obliczalną bijekcją. Definiujemy zbiór  $B$  następująco:

$$\varphi(i, n) \in B \iff n \in A_i$$

Ten zbiór jest dobry, a o tym, że  $A_i \leq_{rek} B$ , świadczy redukcja  $\varphi(i, \cdot)$ .

$2 \Rightarrow 1$ :

Odpowiedź na pytanie ze wskazówki: 1. Jest to zbiór  $f^{-1}(B)$ . Z tego wynika, przy uwzględnieniu faktu, że redukcji (jako programów), jest tylko przeliczalnie wiele, że dla ustalonego zbioru  $B$  istnieje tylko przeliczalnie wiele zbiorów  $A_i$  takich, że  $A_i \leq_{rek} B$ . Zatem, z tego, że  $\mathbb{D} \subseteq \{A_i : i \in \mathbb{N}\}$  wynika, że zbiór  $\mathbb{D}$  musi być przeliczalny.

**Zadanie 105.**

Uwaga:  $N_{emp}$  jest r.e. Semi-rozstrzyga go następujący program  $\Phi_{N_{emp}}$ :

wczytaj  $n$

dla  $k = 1, 2, 3, \dots$  :

odpal  $\Phi_n(i)$  na  $k$  kroków dla  $i = 1, 2, \dots, k$   
 jeżeli na którymś z argumentów  $\Phi_n$  się zatrzymał : zwróć 1

a) Tak. Świadczy o tym następująca redukcja  $f$ :  
 Jako  $f(n)$  przyjmijmy numer następującego programu:  
 wczytaj  $k$   
 odpal  $\Phi_{N_{emp}}(n)$   
 zwróć 1  
 Dowód poprawności pozostawiamy jako ćwiczenie.

b) Nie, bo  $Tot$  nie jest r.e. (zadanie 89), a  $N_{emp}$  jest.

### Zadanie 107.

Niech zbiory  $A_i$  będą wszystkimi nieskończonymi zbiorami r.e.  
 Definiujemy indukcyjnie zbiory  $B$  i  $C$  w następujący sposób:

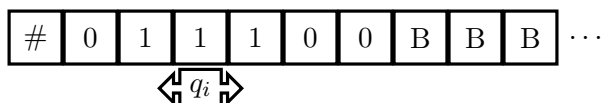
1.  $B_0 = C_0 = \emptyset$
2. Niech  $b_i, c_i \in A_i \setminus C_{i-1}$ ,  $b_i \neq c_i$ . Dla każdego  $j$  zbiór  $C_j$  jest skończony, więc zawsze znajdziemy takie elementy
3.  $B_i = B_{i-1} \cup \{b_i\}$
4.  $C_i = C_{i-1} \cup \{b_i, c_i\}$
5.  $B = \bigcup_i B_i$
6.  $C = \bigcup_i C_i$

Oczywiście zbiór  $B$  jest nieskończony, bo na każdym kroku dodajemy do niego jeden element. Z drugiej strony, dla każdego  $A_i$  istnieje jego element  $c_i$ , taki że  $c_i \notin B$ . W związku z tym, dla każdego  $i$  zachodzi  $A_i \not\subseteq B$ . Zatem  $B$  jest nieskończonym zbiorem bez nieskończonego podzbioru r.e.

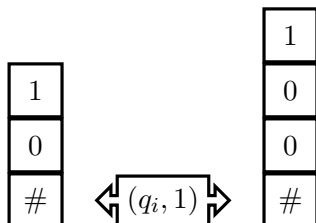
### Zadanie 110.

Bso. przyjmujemy, że mamy Maszynę Turinga z pustą taśmą wejściową i alfabetem złożonym z 0 i 1 oraz symbolu pustej taśmy - B.

a) Bedziemy utrzymawać jako stan automatu ze stosami dwie części aktualnie używanego skrawka taśmy MT: na lewo i na prawo od głowicy (lewy i prawy stos). W stanie będziemy pamiętali aktualny stan MT oraz komórkę na której jest głowica ( $Q' = Q \times \Sigma$ ).



zamieniamy na



Operacje MT implementuje się w prosty sposób.

b) zamienimy stos na licznik: patrzeć na stos jako na liczbę w systemie binarnym (najbardziej znaczące cyfry na dole stosu). Dodatkowo, aby uniknąć kłopotu z zerami wiodącymi, traktujemy symbol dna stosu # jako 1.

Operacje stosowe (zdejmowanie i dodawanie symboli) implementujemy jako odpowiednio dzielenie całkowite oraz mnożenie przez 2 + ewentualne dodanie 1. Do tego wykorzystujemy pozostałe dwa liczniki.

c) konfiguracja 4 liczników jest ciągiem 4 liczb naturalnych  $(a, b, c, d)$ . W związku z tym w prosty sposób można ją zakodować jako jedną liczbę równą  $2^a 3^b 5^c 7^d$ . Drugi licznik jest potrzebny do implementacji wymaganych operacji na 4 licznikach: dodawania i odejmowania ze starego licznika  $\rightarrow$  mnożenia i dzielenia nowego licznika przez odpowiednią liczbę. Ponadto łatwo sprawdzać, czy któryś ze starych liczników jest pusty - wystarczy sprawdzić podzielność przez odpowiednią liczbę.

#### Zadanie 114.

Redukcja z PCP do problemu niepustego przekroju gramatyk:

Dostajemy instancję PCP:  $P = \{\langle l_i, r_i \rangle : i \in \{1, 2, \dots, n\}\}$ . Budujemy następujące gramatyki  $G$  i  $H$ :

1.  $N = \Sigma \cup \{1, 2, \dots, n\}$
2.  $T = \{S\}$
3.  $\Pi_G = \{S \rightarrow l_i S i \mid \varepsilon\}$
4.  $\Pi_H = \{S \rightarrow r_i S i \mid \varepsilon\}$

Oczywiście  $L_G \cap L_H \neq \emptyset (\{\varepsilon\}) \iff$  zadana instancja PCP ma rozwiązanie.

#### Zadanie 115.

Skoro języki  $L_G$  i  $L_H$  z zadania 114 są rozpoznawane w naturalny sposób przez deterministyczne automaty ze stosem, to ich dopełnienia są rozpoznawane przez dopełnicze automaty  $\Rightarrow$  języki  $L_G^c$  i  $L_H^c$  są bezkontekstowe. Z praw de Morgana rachunku zbiorów wynika, że

$$L_G \cap L_H = (L_G^c \cup L_H^c)^c$$

czyli  $L_G \cap L_H = \emptyset \iff (L_G^c \cup L_H^c) = A^* \iff$  zadana instancja PCP nie ma rozwiązania.

Teraz wystarczy pokazać, jak program ma wygenerować gramatykę dla  $L_G^c \cup L_H^c$ , co będzie redukcją z dopełnienia PCP, które jest oczywiście nierozstrzygalne. Być może najłatwiej zbudować deterministyczny automat ze stosem, rozpoznający  $L_G$ , i powiedzieć, że ponieważ jest deterministyczny, to można wywrócić stany akceptujące na lewą stronę i dostaniemy automat rozpoznający dopełnienie. A z niego można wyprodukować gramatykę (ponoć wykład).

### Zadanie 116.

Nie. Redukcja z zadania 115:

Weź gramatykę  $G$  z problemu  $L_G = ?A^*$ . Wyprodukuj gramatykę  $G_{all}$ , taką że  $L_{G_{all}} = A^*$ . Zwróć parę gramatyk  $G$  i  $G_{all}$ .

Oczywiście zachodzi  $L_G = A^* \iff L_G = L_{G_{all}}$ , co kończy dowód poprawności.

### Zadanie 118.

Redukcja z semiThuego:

Dostajemy zbiór produkcji  $P$  oraz słowa  $v$  i  $w$ . Idea: zapętlić słowo (skleić początek z końcem), dodając przy tym specjalny znak początku słowa ( $\#$ ). Formalnie:

1.  $\Sigma' = \Sigma \cup \{\#\}$
2.  $v' = \#v$
3.  $w' = \#w$
4.  $P' = P \cup \{\langle a, a \rangle : a \in \Sigma'\}$

Implikacja w prawo jest, jak zwykle, dość oczywista. Do implikacji w lewo trzeba uzasadnić przekonująco, że jeżeli  $v' \xrightarrow{P'}^* w'$ , to można naśladować potrzebne produkcje i z  $v$  otrzymać  $w$  w zwykłym semiThu.

### Zadanie 143.

Każdą klauzulę o 5ciu literałach ( $l_1 \vee l_2 \vee l_3 \vee l_4 \vee l_5$ ) zamieniamy na następującą koniunkcję klauzul 3-literałowych:

$$(l_1 \vee l_2 \vee y_1) \wedge (l_3 \vee \neg y_1 \vee \neg y_2) \wedge (l_4 \vee l_5 \vee y_2)$$

Dla każdej klauzuli tworzymy nowe zmienne  $y$ . Oczywiście wartościowania spełniające powstałe formuły istnieją (lub nie) równocześnie.