Języki formalne i złożoność obliczeniowa oswojone

rozwiązania opracował
Arkadiusz Janicki <arek@wywrota.pl>
Instytut Informatyki, Uniwersytet Wrocławski
wersja z dnia 8 września 2010

Spis treści

Wstęp **3**
Treści zadań . . . 4
Literatura . . . 4

Część I - Automaty i języki **5**
2009.A.1 Czy istnieje NFA rozpoznający L_p, że $|Q| < p + 3$ . . . 5
2009.B.2 M_n– wszystkie litery oprócz być może jednej . . . 6
2009.B.3 M_n– wszystkie litery oprócz być może jednej i rodziny . . . 7

Część II – Teoria obliczalności **8**
2003.B.4 $F(a_i) = f(a_{i-1}) + g(a_{i-1}) + h(a_{i-1})$ . . . 8
2005.B.6 programy działające w czasie kwadratowym . . . 9
2006.A.4 Automat machający dwiema nogami . . . 10
2007.B.4 zatrzymuje się $\varphi_n(k)$ lub $\varphi_m(k)$ . . . 11
2007.B.5 Automat chodzący po kolorowej ćwiartce . . . 11
2008.A.5 Programy zatrzymujące się zawsze lub nie zatrzymujące nigdy $E \leq_{rek} A$ . . . 13
2008.A.6 Programy zatrzymujące się zawsze lub nie zatrzymujące nigdy $A \leq_{rek} E$ . . . 13
2008.B.4 Funkcja całkowita rekurencyjna, czasami malejąca . . . 14
2008.B.5 Automat skończony spacerujących po ćwiartce . . . 14
2008.B.6 Automat skończony spacerujących po połówce . . . 16
2009.A.4 R.E. trudność . . . 16
2010.A.5 Redukcja, która jest„na” . . . 17

Część III – Teoria złożoności **18**
2007.A.8 Problem P_{z8} . . . 18
2008.A.8 Gra w kompromis . . . 20

Zadania z ćwiczeń **21**
Zadanie 20. $w^n = v$ . . . 21
Zadanie 22. $L_{/2}$ . . . 21
Zadanie 29. $2n|w|_0 \leq |w|_1 \leq (2n+1)\,|w|_0$ . . . 21
Zadanie 32. $\neg\exists x : w = xx$ . . . 22
Zadanie 33. $\exists x : w = xx$ . . . 22
Zadanie 34. $L_{n3w} = \{w \in \Sigma^* \setminus \{www : w \in \Sigma^*\}\}$, $\Sigma = \{0,1\}$ . . . 22
Zadanie 37. $L_{nwvw} = \{w \in \Sigma^* \setminus \{wvw : w, v \in \Sigma^*, |w| = |v|\}\}$, $\Sigma = \{0,1\}$ . . . 23

Zadanie 100. $CLIQUE_{n/2}$ 23
Zadanie 102. Spełnialność 9/10 klauzul 24
Zadanie 105. Problem smutnych strażników 24
Zadanie 118. $TOTAL_{RE} \in PSPACE$ 25

„Wartość zajmowania się poważnie przez pewien czas nauką ścisłą nie polega właściwie na wynikach: bowiem one w porównaniu z morzem tego, co wiedzieć warto, będą nieskończenie małą kroplą. Wynika jednak stąd przyrost energii, zdolności wnioskowania, ciągłości w wytrwaniu; nauczono się osiągać cel środkami, zastosowanymi do celu. W tem przeto znaczeniu jest rzeczą bardzo cenną, ze względu na wszystko, czem człowiek później się zajmuje, żeby przedtem był człowiekiem naukowym."

Fryderyk Nietzsche — „Ludzkie, Arcyludzkie"

Wstęp

Niniejszy zbiór zaczął powstawać latem 2009 roku podczas moich przygotowań do egzaminu poprawkowego z języków formalnych. Czułem potrzebę opracowania kilku zadań egzaminacyjnych w sposób wzorcowy, tzn. po pierwsze, bez żadnych uchybień rachunkowych, a po drugie, zapisując rozwiązania w sposób jak najbardziej czytelny — bez żadnych skrótów myślowych ani, z drugiej strony, bez niepotrzebnego rozpisywania się — tak, aby oddając tak rozwiązane zadania na egzaminie otrzymać maksymalną ilość punktów. Mam nadzieję, że przynajmniej w niektórych miejscach mi się to udało.

Kiedyś zastanawiałem się, czy upublicznienie rozwiązań nie będzie działaniem uznawanym za niepedagogiczne. Właściwy proces rozwiązywania zadań, zaproponowany przez jednego z wykładowców w naszym instytucie, składa się wszak z trzech etapów:

1. Samodzielnych prób rozwiązania zadania z wykorzystaniem jedynie własnej wiedzy.
2. Gdy to nie przynosi zamierzonych efektów — sięgnięcia do zewnętrznych źródeł wiedzy takich jak notatki z wykładu czy książki. Być może tam uda się odnaleźć informacje, które naprowadzą nas na rozwiązanie.
3. Gdy i to zawodzi — zalecana jest konsultacja z kolegą (lub koleżanką) z grupy; wspólna wymiana myśli często bywa owocna. Przy trudnych zadaniach udajemy się na konsultacje do osoby prowadzącej ćwiczenia lub do wykładowcy.

Żaden z powyższych punktów nie przewiduje zaglądania do gotowych rozwiązań. Taka praktyka w wielu przypadkach, zamiast pomagać – tak naprawdę szkodzi. Nauczenie się rozwiązania na pamięć i przerysowanie odpowiednich symboli na tablicy nie oznacza wcale opanowania materiału. Uważam jednak, że przedmiot wykładu jest na tyle trudny, iż wszelkie pomoce naukowe które mogą pomóc w zrozumieniu materiał wyładu powinny być odstępne. Ostatecznie zakwalifikowałem ten zbiór do pomocy naukowych wymienionych w punkcie drugim, rozwiązując tym samym mój dylemat moralny. Odradzam jednak studentom zaglądania do rozwiązań przed rzetelnym przejściem przez powyższe trzy punkty.

Dziękuję Sebastianowi Bali i Łukaszowi Jeżowi za pomoc i weryfikację niektórych rozwiązań, a winę za pozostałe błędy biorę na siebie.

Arkadiusz Janicki
arek@wywrota.pl

Treści zadań

Treści zadań pochodzą z egzaminów z wykładu prowadzonego przez Jerzego Marcinkowskiego w Instytucie Informatyki Uniwersytetu Wrocławskiego oraz ze zbioru *„Sto łatwych zadań z języków formalnych i złożoności obliczeniowej”* opracowanego przez wykładowcę.

- `http://aerjotl.net/jfizo/`
  zebrane przeze mnie materiały do przedmiotu: treści egzaminów oraz listy zadań
- `http://www.ii.uni.wroc.pl/~jma/kurs/`
  strona internetowa wykładu

Literatura

1. *Wprowadzenie do teorii obliczeń* - Michael Sipser; tłum. Przemysława Kanarek (WNT 2009)
2. *Złożoność obliczeniowa* - Christos H. Papadimitriou; tłum. Przemysława Kanarek, Krzysztof Loryś (WNT 2007)
3. *Wprowadzenie do teorii automatów, języków i obliczeń* - John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman; tłum: Beata Konikowska (PWN 2005)

Część I - Automaty i języki

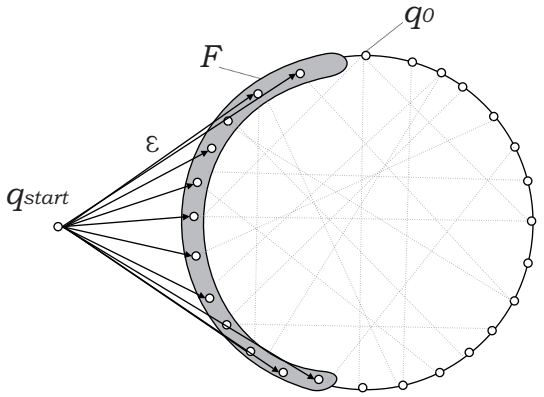
2009.A.1 Czy istnieje NFA rozpoznający L_p, że $|Q| < p + 3$

Niech $p > 5$ będzie pewną liczbą pierwszą, a L_p językiem tych słów nad $\{0, 1\}$ które czytane jako liczba w zapisie binarnym dają, jako resztę z dzielenia przez p, jedną z liczb $\left\{1, 2, ..., \frac{p-1}{2}\right\}$, przy czym liczby czytamy „od prawej", czyli od najmniej znaczącego bitu (to znaczy pierwszy znak słowa jest ostatnią cyfrą liczby). Czy istnieje niedeterministyczny automat skończony o mniej niż $p + 3$ stanach rozpoznający język L_p?

Pokażemy jak skonstruować taki automat. Na początku załóżmy, że słowo jest wczytywane od najbardziej znaczącego bitu. W takim wypadku NFA $A_{\rightarrow}$ rozpoznający L_p język będzie składał się z p stanów ponumerowanych $0, 1, ..., p-1$; stanem startowym będzie q_0, natomiast stany $\left\{q_1, q_2, ...q_{\frac{p-1}{2}}\right\}$ będą stanami akceptującymi. Ponieważ w pierścieniu $mod\, p$, operacja mnożenia zachowuje własności podzielności, możemy zdefiniować funkcję przejścia:

$$\begin{aligned} \bigvee_{0 \leqslant i < n} \langle q_i, q_{2i \bmod p}, 0 \rangle &\in \delta \\ \bigvee_{0 \leqslant i < n} \langle q_i, q_{2i+1 \bmod p}, 1 \rangle &\in \delta \end{aligned}$$

Poprawność tego automatu wydaje się być oczywista. Kolejnym krokiem będzie zmiana sposobu wczytywania słów, tak, aby znaki były wczytywane od najmniej znaczącego bitu liczby — tak jak w definicji zadania. Aby tego dokonać zastosujemy operację „odwracania kota ogonem"[1] aby otrzymać $A_{\leftarrow} = \langle \Sigma, Q', q_{start}, F', \delta' \rangle$:



1. odwrócimy groty wszystkich strzałek w grafie przejścia automatu
2. dodamy jeden nowy stan — q_{start} i połączymy go z każdym $q \in F$ ϵ-przejściem
3. jako nowy zbiór stanów akceptujących zaznaczymy $F' = q_0$

[1] Po dowód poprawności operacji odwracania odsyłam do ćwiczenia 19/2009.

Funkcja przejścia δ' będzie wyglądać następująco:

$$\begin{aligned}
\bigvee_{0\leqslant i<n} \langle q_{2i \bmod p}, q_i, 0\rangle &\in \delta' \\
\bigvee_{0\leqslant i<n} \langle q_{2i+1 \bmod p}, q_i, 1\rangle &\in \delta' \\
\bigvee_{1\leqslant i\leqslant (p-1)/2} \langle q_{start}, q_i, \epsilon\rangle &\in \delta'
\end{aligned}$$

Otrzymaliśmy zatem DFA o $q+1$ stanach. □

2009.B.2 M_n– wszystkie litery oprócz być może jednej

W kolejnych dwóch zadaniach niech M_n będzie językiem tych słów nad alfabetem $\{1,2,...,n\}$ (gdzie n jest pewną liczbą parzystą) w których występują wszystkie litery alfabetu oprócz być może jednej. Przez $\overline{M_n}$ rozumiemy dopełnienie języka M_n do zbioru $\{1,2,...,n\}^*$.

a) Jaką minimalną liczbę stanów musi mieć deterministyczny automat skończony rozpoznający $\overline{M_n}$?

b) Jaką minimalną liczbę stanów musi mieć niedeterministyczny automat skończony rozpoznający $\overline{M_n}$?

---

punkt a)

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

$M_n = \left\{w \,:\, w \in \Sigma^* \wedge \exists_{a\in\Sigma\cup\{\epsilon\}} \forall_{i\in\Sigma\setminus\{a\}}\, i \in LITERY(w)\right\}$

Słowo należy do $\overline{M_n}$ jeśli nie zawiera przynajmniej dwóch liter z alfabetu. Rozpatrzymy klasy relacji należenia do $\overline{M_n}$ i skorzystamy z twierdzenia o indeksie aby odnaleźć liczbę stanów minimalnego DFA.

1	$w = \epsilon$	— klasa słowa pustego
n	$w \in a*$	— klasa słów zawierających jedną literę z alfabetu
$\binom{n}{2}$	$w \in \{a,b\}*$	— klasa słów zawierających dwie litery z alfabetu
$\vdots$		
$\binom{n}{n-2}$	$\cdots$	—w zawiera $n-2$ litery z alfabetu
1	$w \in M_n$	— wypadamy poza język

$$\begin{aligned}
|Q_{DFA}| &= \binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{n-2} + 1 = \\
&= \sum_{k\leqslant n} \binom{n}{k} - \binom{n}{n-1} - \binom{n}{n} + 1 = \\
&= 2^n - n - 1 + 1 = \underline{2^n - n}
\end{aligned}$$

punkt b)

$|Q_{NFA}| = \binom{n}{2} + 1$ — odpowiada to ilości sposobów, na które można wybrać dwie litery z n-elementowego alfabetu. (???)

2009.B.3 M_n– wszystkie litery oprócz być może jednej i rodziny

Udowodnij, że każdy niedeterministyczny automat skończony rozpoznający język M_n musi mieć więcej niż $2^{\frac{n}{2}-1}$ stanów.

Wskazówka. Dla liczby naturalnej k, takiej, że $1 \leq k \leq n/2$ nazwijmy parę liczb $\{2k-1, 2k\}$ *rodziną*. Powiemy że słowo $x \in \{1, 2, ...n\}^*$ *nie rozdziela rodzin*, jeśli zawsze wtedy, gdy jedna z liter z jakiejś rodziny występuje w słowie x, w słowie tym występuje również druga z tych liczb. Powiemy że słowo x jest *rosnące*, gdy każda jego kolejna litera jest liczbą większą niż poprzednia. Ile jest słów nie należących do M_n, które są rosnące i nie rozdzielają rodzin?

$NOSPLIT = \{w \in \{1, ..., n\}^* : w \notin M_n,\ w - rosnące,\ w\, nie\, rozdziela\, rodzin\}$

Liczba możliwości wyboru rodzin to $2^{\frac{n}{2}}$ i tylko jedna spośród tych możliwości należy do M_n (oczywiście ta, w której bierzemy wszystkie rodziny), zatem $|NOSPLIT| = 2^{\frac{n}{2}} - 1$.

Załóżmy nie wprost, że istnieje NFA B rozpoznający M_n i $Q_B \leq 2^{\frac{n}{2}-1}$.

Rozważmy dwa słowa $w, v \in NOSPLIT$, $w \neq v$ takie, że automat B czytając je zatrzymuje się w tym samym stanie q. Takie słowa istnieją, bo słów rozróżnialnych pod względem rodzin jest $2^{\frac{n}{2}} - 1$, a dla $n \geq 4$ liczba stanów jest mniejsza ($2^{\frac{n}{2}-1} < 2^{\frac{n}{2}} - 1$).

Rozważmy przypadki:

1. $|w| = |v|$
   Istnieje wtedy rodzina i która należy do v, ale nie należy do w. Wynika z tego, że słowo $w\bar{v}$[2] nie zawiera rodziny i, a więc słowo $w\bar{v} \notin M_n$ a ma w automacie B ścieżkę akceptującą. ↯
2. $|w| < |v|$ (bez straty ogólności)
   Załóżmy, że w zawiera m rodzin, a v — k rodzin i $m < k$. Wtedy słowo $w\bar{v}$ zawiera $m + \frac{n}{2} - k$ rodzin, a $m + \frac{n}{2} - k < \frac{n}{2}$. Aby należeć do M_n słowo musi zawierać $\frac{n}{2}$ rodzin, więc $w\bar{v} \notin M_n$ a ma w B ścieżkę akceptującą. ↯ □

[2] Słowo $\bar{v}$ jest „dopełnieniem" v — czyli takim słowem, w którym występują wszystkie te i tylko te litery alfabetu, których nie ma w v. $\forall_{a \in \Sigma} (a \in LITERY(v) \vee a \in LITERY(\bar{v})) \wedge \neg (a \in LITERY(v) \wedge a \in LITERY(\bar{v}))$.

Część II – Teoria obliczalności

2003.B.4 $F(a_i) = f(a_{i-1}) + g(a_{i-1}) + h(a_{i-1})$ [3]

Dla danych funkcji$f, g, h : \{0, 1, ..., p\} \to \{0, 1, ..., p\}$ i danego nieskończonego ciągu liczb naturalnych $(a_1, a_2, a_3, ...)$, niech $F_{f,g,h}(a_1, a_2, ...)$ będzie ciągiem liczb naturalnych którego i-ty element jest równy $f(a_{i-1}) +_p g(a_{i-1}) +_p h(a_{i-1})$, gdzie $+_p$ oznacza dodawanie modulo p (przyjmujemy, że $a_0 = 0$). Udowodnij, że problem:

> dane funkcje f, g, h oraz skończone ciągi $(b_1, b_2, ..., b_k)$ i $(c_1, c_2, ..., c_l)$. Czy istnieje liczba iteracji n taka, że $F^n_{f,g,h}(b_1, b_2, ..., b_k, 0, 0, ...) = (c_1, c_2, ..., c_l, 0, 0, ...)$

jest nierozstrzygalny.

Pokażemy, że problem jest nierozstrzygalny poprzez redukcję do problemu stopu. Redukcja będzie przyjmować jako argument Maszynę Turinga $MT = \langle \Sigma, Q, q_0, F, \delta \rangle$ oraz słowo w, a będzie zwracać f, g, h oraz ciągi $B = (b_1, b_2, ..., b_k)$ i $C = (c_1, c_2, ..., c_k)$. Będzie możliwe wykonanie n operacji przekształcających jeden ciąg w drugi wtedy i tylko wtedy, gdy MT zatrzymuje się dla słowa w po n krokach.

- W jednej liczbie będącej elementem ciągu zakodujemy informację o symbolu $a \in \Sigma$, o $q \in Q \cup \{\phi\}$ oraz $k \in \{L, R, \phi\}$ - informację skąd przyszliśmy w ostatnim kroku.
- Wybierzmy dowolny sposób kodowania dla tych trzech symboli, możemy np. zindeksować tablicę wszystkich możliwych kombinacji, wtedy otrzymamy $p = |\Sigma| \cdot (|Q| + 1) \cdot 3$
- Za ciąg B przyjmijmy zakodowane słowo wejściowe w - każdy symbol na osobnej pozycji, z dodatkowymi informacjami q_0, L w pierwszej komórce.
- Przekształćmy MT tak, aby po zakończeniu obliczeń czyściła taśmę roboczą i zatrzymywała się na pierwszej pozycji w specjalnym stanie i tą konfigurację zakodujmy jako ciąg C.
- Niech $\delta(a, q) = \langle b, q', m \rangle$. Funkcje f, g, h skonstruujemy w ten sposób, aby g zwracała symbol z alfabetu, jaki MT zapisuje do komórki, a f i h - stan i kierunek przejścia.
  - $g'(\langle a, q, k \rangle) = \begin{cases} \langle b, \phi, \phi \rangle & gdy\, k \neq \phi \\ \langle a, \phi, \phi \rangle & gdy\, k = \phi \end{cases}$
  - $f'(\langle a, q, k \rangle) = \begin{cases} \langle \phi, q', m \rangle & gdy\, k = L \\ \langle \phi, \phi, \phi \rangle & w\, p.p. \end{cases}$
  - $h'(\langle a, q, k \rangle) = \begin{cases} \langle \phi, p, m \rangle & gdy\, k = R \\ \langle \phi, \phi, \phi \rangle & w\, p.p. \end{cases}$

1	2	3	4	5	
q_o, L a_1	a_2	a_3	a_4	...	$\delta(a_1, q_0) = \langle b, q_1, L \rangle$
 b	q_1, L a_2	a_3	a_4	...	$\delta(a_2, q_1) = \langle c, q_2, R \rangle$
q_2, R b	c	a_3	a_4	...	

[3] Znane także jako zadanie nr 77 z listy 2009.

Widzimy, że powyższy model obliczeń umożliwia nam symulowanie maszyny Turinga i gdyby był rozstrzygalny, oznaczałoby to też, że problem stopu dla maszyny Turinga jest rozstrzygalny, a nie jest. $\lightning$. □

2005.B.6 programy działające w czasie kwadratowym

Udowodnij, że problem czy dany algorytm działa w czasie kwadratowym jest nierozstrzygalny. Dokładniej mówiąc, wykaż, że zbiór numerów tych maszyn Turinga, które zatrzymują się dla każdego słowa wejściowego, i dla których istnieją liczby a, b, c takie, że zakończenie działania maszyny następuje zawsze prędzej niż po $ax^2 + bx + c$. krokach, gdzie x jest długością słowa wejściowego, jest nierekurencyjny.

Niech A będzie zdefiniowanym w zadaniu zbiorem numerów maszyn Turinga. Załóżmy nie wprost, że A jest rekurencyjny, to znaczy, że istnieje $\varphi_A$rozstrzygający przynależność do A. Pokażemy, że $K \leq_{rek} A \Rightarrow K$ jest rekurencyjny, co naturalnie jest nieprawdą.

Rozważmy program Ψ:

- wczytaj n
- skonstruuj program φ_t:
  - wczytaj m
  - $d = log_2 m$ *długość danych*
  - uruchom $\varphi_n(n)$ na co najwyżej d kroków
  - jeśli otrzymałeś wynik - zapętl się
  - wpp. zwróć 1
- zwróć $\varphi_A(t)$

1. $n \in K$, zatem istnieje takie m od którego φ_t będzie się zapętlał. To oznacza, że $t \notin A$, więc jako wynik $\Psi(n)$ otrzymamy 0.
2. $n \in \overline{K}$, więc φ_t zawsze będzie zwracał jedynkę po czasie równym długości danych + stała c. Z tego wynika, że $t \in A$, a $\Psi(n)$ zwróci 1.

Pokazaliśmy, że K jest rekurencyjny. $\lightning$ □

2006.A.4 Automat machający dwiema nogami

Automat z dwiema nogami zdefiniujemy sobie jako piątkę $\langle \Sigma, Q, q_0, F, \delta \rangle$, gdzie Σ jest skończonym alfabetem, Q skończonym zbiorem stanów, $q_o \in Q$ jest stanem początkowym. $F \subseteq Q$ jest zbiorem stanów akceptujących a $\delta : Q \times \Sigma \times \Sigma \to Q \times \{0,1\} \times \{0,1\}$ jest funkcją przejścia. Funkcja przejścia jest rozumiana następująco: jeśli $\delta(q, a_1, a_2) = (q', b_1, b_2)$, to gdy automat jest w stanie q, lewą nogę ma na symbolu a_1 a prawą nogę na symbolu a_2, to ma przejść do stanu q', przesunąć lewą nogę o b_1 symboli w słowie wejściowym w prawo i przesunąć prawą nogę o b_2 symboli w prawo. Na początku działania automat jest w stanie q_0 i ma obie nogi na pierwszej literze słowa. Automat akceptuje słowo, jeśli znajdzie się obiema nogami na jego ostatniej literze i będzie wtedy w stanie $q \in F$. Czy problem niepustości automatu z dwiema nogami jest rozstrzygalny? Przez problem niepustości rozumiemy tu problem istnienia jakiegokolwiek słowa akceptowanego przez dany automat.

Rozwiązanie

Problem jest nierozstrzygalny - pokażemy redukcję $PCP \leq_{rek} EMPTY_{2LEGS}$.

Idea jest taka, że na początku automat niedeterministycznie zgaduje od której pary $\langle v_i, w_i \rangle$ należy zacząć, po czym przesuwa obie głowice w prawo sprawdzając, czy na taśmie znajdują się słowa v_i, w_i. Gdy dojdzie do końca każdego ze słów nie stwierdzając błędów, przechodzi w stan q_{ok} i zgaduje kolejną parę, lub - jeśli głowice się spotkały - przechodzi w stan q_{accept} . PCP ma rozwiązanie wtedy i tylko wtedy gdy głowice zejdą się w stanie akceptującym dla pewnego ciągu par.

Szczegóły konstrukcji

Dla danej instancji problemu $PCP = \left\{ \{\langle v_i, w_i \rangle\}_{i=1}^{k} \; w_i, v_i \in \Sigma^* \right\}$

konstruujemy automat z dwiema nogami $2LEGS = \langle \Sigma, Q, q_0, F, \delta \rangle$

- $Q = \left(\{1, 2, ..., k\} \times \left\{ 0, 1, ..., \underset{1 \leq i \leq k}{max} |w_i| \right\} \times \left\{ 0, 1, ..., \underset{1 \leq i \leq k}{max} |v_i| \right\} \right) \cup \{q_0, q_{ok}, q_{accept}, q_{reject}\}$
  Stan indeksowany krotką $\langle i, j, k \rangle$ mówi nam o tym, że automat sprawdza parę $\langle v_i, w_i \rangle$ oraz, że zostało mu do przeczytania j liter słowa v_i i k liter słowa w_i.
- $F = \{q_{accept}\}$
- $\delta : \underbrace{Q \times \Sigma \times \Sigma}_{input} \times \underbrace{Q \times \{0,1\} \times \{0,1\}}_{output}$

  Dla każdej pary $\langle v_i, w_i \rangle$, $v_i = a_1 a_2 ... a_{|v_i|}$, $w_i = b_1 b_2 ... b_{|w_i|}$ w relacji δ mamy:
  - $\langle q \in \{q_0, q_{ok}\}, a_1, b_1, \langle i, |v_i| - 1, |w_i| - 1 \rangle, 1, 1 \rangle$ - jeśli aktualnie nie sprawdzałeś żadnej pary słów i widzisz pierwsze litery z rozważanej pary, to zmień stan na odpowiedni i przejdź obiema nogami o jedno pole dalej.[4]
  - $\left\langle \langle i, 0, 1 \rangle, *, b_{|w_i|}, q \in \{q_{ok}, q_{accept}\}, 0, 1 \right\rangle$ - jeżeli sprawdzałeś słowa o indeksie i oraz pozostała ci do przeczytania jedna litera słowa w_i, i widzisz tą literę, to przesuń drugą nogę o jedno pole i przejdź w stan q_{ok} - jeśli nogi automatu nie są razem, lub w q_{accept} - jeśli nogi się zeszły.

[4] Na potrzeby przykładu zakładamy, że słowa w_i, v_i są niepuste. Oczywiście bez tego założenia również możemy skonstruować redukcję.

Jeśli istnieje słowo akceptowane przez ten automat, to jest ono równocześnie rozwiązaniem danej instancji problemu PCP. Jeśli natomiast skonstruowany automat nie akceptuje żadnego słowa, oznacza to, że także dana instancja PCP nie ma rozwiązania. Jeżeli zatem problem $EMPTY_{2LEGS}$ byłby rozstrzygalny, to rozstrzygalny byłby także problem PCP, a oczywiście nie jest. □

2007.B.4 zatrzymuje się $\varphi_n(k)$ lub $\varphi_m(k)$

Niech A będzie zbiorem takich par numerów programów (n, m), że dla każdej liczby naturalnej k albo program o numerze n uruchomiony dla danej k się zatrzymuje, a program o numerze m uruchomiony dla danej k się nie zatrzymuje, albo program o numerze m uruchomiony dla danej k się zatrzymuje, a program o numerze n uruchomiony dla danej k się nie zatrzymuje.
Czy A jest rekurencyjnie przeliczalny?

$A = \{\langle n, m\rangle : \forall_{k\in\mathbb{N}}\, zatrzymuje\, się\, \varphi_n(k)\ lub\ \varphi_m(k)\}$

Załóżmy nie wprost, że A jest rekurencyjnie przeliczalny. Oznacza to, że istnieje program φ_A rozpoznający go.
Rozważmy następujący program Ψ:

- wczytaj n
- $x \leftarrow$numer programu zawsze zwracającego 1
- dla $i = 1..\infty$
  - uruchom $\varphi_n(n)$ na i kroków
    jeśli się zatrzymał zwróć 1
  - uruchom $\varphi_A(\langle x, n\rangle)$ na i kroków
    jeśli się zatrzymał zwróć 0

Program Ψ rozstrzyga zbiór K. ↯ □

2007.B.5 Automat chodzący po kolorowej ćwiartce

Automat chodzący po pierwszej ćwiartce będzie opisany przez skończony zbiór instrukcji będący podzbiorem $Q \times K \times \{T, F\} \times \{T, F\} \times Q \times \{n, s, e, w\}$, gdzie Q jest skończonym zbiorem *stanów* zaś K jest skończonym zbiorem *kolorów*, i będący funkcją (częściową) ze względu na cztery pierwsze argumenty. Wejściem automatu jest ćwiartka płaszczyzny: nieskończona, w kierunku wschodnim i północnym, szachownica, której każde pole pokolorowane jest jednym z kolorów ze zbioru K. Na początku swojego działania automat umieszczany jest w (jedynym) rogu szachownicy, a następnie w każdym ruchu, w zależności od tego w jakim jest stanie, na polu jakiego koloru stoi, oraz czy jest prawdą że stoi na zachodnim skraju szchownicy, i czy jest prawdą że stoi na południowym skraju szchownicy, zmienia stan i przechodzi o jedno pole na północ, południe, wschód lub zachód. Instrukcje są tak zbudowane, że nie może opuścić szachownicy, to znaczy np. że gdy stoi na południowym skraju, to instrukcja nie może mu kazać iść na południe. Jeśli automat nie ma instrukcji mówiącej mu jak się zachować w jakiejś sytuacji, to zatrzymuje się i akceptuje wejście. Automat jest totalny jeśli akceptuje każde wejście.

Udowodnij, że problem totalności jest dla automatów chodzących po pierwszej ćwiartce nierozstrzygalny. Gdybyś przy okazji dowodu konstruował jakiś automat, to nie musisz wypisywać wszystkich jego instrukcji, natomiast koniecznie powinieneś precyzyjnie opisać ideę jego działania.

Pokażemy redukcję problemu stopu automatu z dwoma licznikami do ograniczonej wersji problemu z zadania. Ograniczenie będzie polegało na tym, że użyjemy tylko jednego koloru do pokolorowania pól szachownicy. Problem totalności będzie w tej sytuacji równoznaczny z problemem akceptacji przez automat jednokolorowej ćwiartki.

Redukcja: Dla każdego automatu z dwoma licznikami $2CM = \langle Q, l_1, l_2, \delta_{cm}, q_0, q_f \rangle$ zbudujemy automat chodzący po pierwszej ćwiartce $A = \langle Q', K, \delta, q_0' \rangle$. Konstruowany automat A będzie akceptował wejście wtw. gdy działanie $2CM$ kończy się akceptacją. Idea konstrukcji automatu polega na tym, aby symulować działanie maszyny z dwoma licznikami i wykorzystać współrzędne na szachownicy jako wartości liczników, a osie współrzędnych jako wartości zerowe. Zakładamy, że $2CM$ startuje z pustymi licznikami - odpowiada to położeniu automatu A w rogu szachownicy.

- $Q' = (Q \times \{0,1\}) \cup \{q_x'\}$
  - stany automatu będą indeksowane krotką $\langle q, b \rangle$, gdzie $q \in Q$, $b \in \{0,1\}$
  - q_x' jest dodatkowym wyróżnionym stanem „nieakceptującym"
  - $q_0' = \langle q_0, 0 \rangle$ – stan początkowy A odpowiada stanowi początkowemu $2CM$,
  - $q_f' = \langle q_f, 0 \rangle$ – definicja automatu chodzącego po pierwszej ćwiartce nie przewiduje explicite stanu akceptującego, ale możemy wyróżnić taki stan podczas konstrukcji. Zakładamy, że stan końcowy w $2CM$ jest tylko jeden i że nie ma z niego żadnego przejścia. Także w konstruowanym automacie z q_f' nie będzie żadnego przejścia, zatem po wejściu w ten stan automat zatrzyma się i zaakceptuje wejście.
- K – tylko jeden kolor (np. różowy)
- δ – dla każdej instrukcji z δ_{cm} w formacie:

  $\langle q_1, c_1, c_2; q_2, i_1, i_2 \rangle$ $\qquad$ $c_1, c_2 \in \{0,1\}$, $i_1, i_2 \in \{-1, 0, 1\}$

  swórz dwie instrukcje (ignorujemy informacje o kolorze):

  $\langle \langle q_1, 0 \rangle, c_1, c_2; \langle q_2, 1 \rangle, d_1 \rangle$

  $\langle \langle q_2, 1 \rangle, *, *; \langle q_2, 0 \rangle, d_2 \rangle$

  Dla ułatwienia rozszerzymy zbiór kierunków o symbol „$\circ$", który będzie oznaczał, że nie poruszamy się w danej płaszczyźnie.

  d_1 jest kierunkiem przejścia w poziomie zależnym od i_1, natomiast d_2 kierunkiem przejścia w pionie zależnym od i_2.
  Wartościom i_1 równym $-1, 0, 1$ będą odpowiadały przejścia d_1 odpowiednio $w, \circ, e$
  Wartościom i_2 równym $-1, 0, 1$ będą odpowiadały przejścia d_2 odpowiednio $s, \circ, n$
  - dodatkowo w δ znajdą się instrukcje obsługujące sytuacje, w których $2CM$ nie wie co zrobić i zatrzymuje się odrzucając wyjście. W takim przypadku automat A powinien przejść do stanu „nieakceptującego" q_x', w którym wpadnie w nieskończoną pętlę: $\langle q_x', *, *, q_x', \circ \rangle$.

Widzimy, że powyższy automat symuluje działanie maszyny z licznikami chodząc po szachownicy i zatrzymując się wyłącznie wtedy, gdy $2CM$ akceptuje wejście. Gdyby problem zdefiniowany w zadaniu był rozstrzygalny, to rozstrzygalny byłby również $STOP_{2CM}$, a oczywiście nie jest. $\square$

2008.A.5 Programy zatrzymujące się zawsze lub nie zatrzymujące nigdy $E \leq_{rek} A$

Udowodnij, że $E \leq_{rek} A$.

Pokażemy redukcję f taką, że $x \in E \Leftrightarrow f(x) \in A$:

- wczytaj n
- zbuduj program Ψ:
  - wczytaj k
  - $\langle a, b\rangle \leftarrow bij(k)$ *bij() jest bijekcją* $\mathbb{N} \to \mathbb{N} \times \mathbb{N}$
  - uruchom $\varphi_n(a)$ na b kroków
  - jeśli się zatrzyma - zapętl się
  - wpp. zwróć 1
- zwróć $\langle\Psi\rangle$

1. $x \in E$ więc $\varphi_n(a)$ nigdy się nie zatrzyma, więc program Ψ który zbudujemy nigdy zawsze będzie zwracał jedynkę. $f(x) \in A$. $\surd$
2. $x \notin E$ zatem istnieje taki argument a, dla którego $\varphi_n(a)$ się zatrzyma. W tym przypadku program Ψ się zapętli, więc jego numer $f(x) \notin A$. $\square$

2008.A.6 Programy zatrzymujące się zawsze lub nie zatrzymujące nigdy $A \leq_{rek} E$

Czy $A \leq_{rek} E$?

Odpowiedź: Nie. Gdyby istniała taka redukcja znaczyłoby to, że zbiór $\overline{K}$[5], jest rekurencyjnie przeliczalny, a jak wiemy nie jest.

Załóżmy, że $A \leq_{rek} E$, to znaczy, że istnieje redukcja $f : A \to E$ taka, że $x \in A \Leftrightarrow f(x) \in E$. Pokażemy program Ψ, który *rozpoznaje*[6] przynależność do $\overline{K}$:

- wczytaj n
- $z \leftarrow \langle\varphi_n(n)\rangle$ *numer programu zwracającego* $\varphi_n(n)$ *niezależnie od argumentu*
- $y \leftarrow f\,(z)$ *numer programu zwracanego przez redukcję* f *na* z
- for $i = 1\, to \,\infty$
  - $\langle a, b\rangle \leftarrow bij(i)$ *bij() jest bijekcją* $\mathbb{N} \to \mathbb{N} \times \mathbb{N}$
  - uruchom $\varphi_y(a)$ na b jednostek czasu
  - jeśli otrzymałeś wynik zwróć 1

1. $n \in K$ - Uruchamiając program Ψ z argumentem $n \in K$ redukcja f zwróci nam numer programu, który się zapętla, więc Ψ też się zapętli.

[5] $\overline{K}$ - czyli zbiór numerów tych programów, które uruchomione dla argumentu będącego numerem uruchamianego programu zapętlają się.

[6] Pojęcie *rozpoznawalności* rozumiem tak, jak zostało zdefiniowane w książce M. Sipsera. Inna nazwa to *semi-rozstrzygalność*.

2. $n \in \overline{K}$ - Gdy z kolei uruchomimy Ψ z argumentem $n \in \overline{K}$ redukcja f zwróci numer programu, który zatrzymuje się dla *jakiegoś* argumentu. Następnie przeszukujemy zbiór wartości φ_y metodą przekątniową i w końcu na pewno znajdziemy ten argument. Wtedy program zwróci wartość 1.

   W ten sposób otrzymaliśmy program *rozpoznający* przynależność do $\overline{K}$ ↯. □

2008.B.4 Funkcja całkowita rekurencyjna, czasami malejąca

Funkcja $f : \mathbb{N} \to \mathbb{N}$ jest całkowita rekurencyjna i taka, że istnieje tylko skończenie wiele liczb naturalnych, dla których $f(n) > f(n+1)$. Czy wynika z tego, że zbiór $f(\mathbb{N})$ jest rekurencyjny?

Odpowiedź: tak. Skoro f jest całkowitą funkcją rekurencyjną to istnieje pewna maszyna Turinga M, która ją rozpoznaje. Możemy zatem znaleźć numer tej maszyny, przeanalizować ją i znaleźć ostatni argument, dla którego zachodzi $f(n) > f(n+1)$ (nazwijmy go k). Następnie sprawdzamy argumenty $0..k$ aby odnaleźć poszukiwaną wartość. Jeśli jej nie znajdziemy, to mamy do czynienia ze znaną nam sytuacją, ponieważ pozostałe argumenty są opisane funkcją rekurencyjną która może być stała lub rosnąca. W obu przypadkach zbiory wartości są rekurencyjne.

2008.B.5 Automat skończony spacerujących po ćwiartce

Niech $S \subseteq \mathbb{Z} \times \mathbb{Z}$ będzie pewnym podzbiorem zbioru punktów kratowych płaszczyzny, takim, że $[0,0] \in S$. Przez T oznaczmy zbiór $\{—1, 0, 1\}$. Dla punktu $[a,b] \in S$ niech $s([a,b]) = \{[x,y] \in T^2 : [a+x, b+y] \in S\}$ (oczywiście zawsze zachodzi $[0,0] \in s[a,b]$, zaś $s[a,b]$ można rozumieć jako „zbiór kierunków w jakich można zrobić krok z $[a,b]$ pozostając w zbiorze S").

W kolejnych dwóch zadaniach rozpatrujemy *automaty skończone spacerujące po zbiorze* S. Automat taki dany jest przez skończony zbiór stanów Q, ze stanem początkowym $q_0 \in Q$, oraz przez zbiór instrukcji π, będący funkcją o dziedzinie zawartej w $Q \times \mathcal{P}(T^2)$ i o zbiorze wartości zawartym w $Q \times T^2$ taką, że (*) jeśli $\pi(q, W) = [q', w]$, to $w \in W$.

Jeśli teraz automat znajduje się w stanie q w punkcie $[a,b] \in S$ i jeśli $\pi(q, s([a,b])) = [q', [x,y]]$, to w kolejnym kroku przechodzi do stanu q' i do punktu $[a+x, b+y]$. Potocznie mówiąc: automat "wie" w jakim jest stanie, i które z ośmiu sąsiednich punktów należą do S, i w zależności od tej swojej wiedzy zmienia stan i przechodzi do jednego z sąsiednich punktów. Zwróć uwagę, że z (*) wynika, że automat nigdy nie opuści zbioru S.

Jeśli automat znajduje się w stanie q w punkcie $[a,b] \in S$ i jeśli$\pi(q, s([a,b]))$ nie jest określone, to automat się zatrzymuje.

Dla ustalonego zbioru S przez *problem stopu dla automatów skończonych spacerujących po* S rozumiemy problem ustalenia, dla danego Q, q_0, π, czy opisany powyżej automat, uruchomiony w stanie q_0 w punkcie $[0,0]$, kiedykolwiek się zatrzyma.

a. (od -1 do 7 punktów). Czy problem stopu dla automatów skończonych spacerujących po S jest rozstrzygalny, jeśli $S = \{[a,b] : a, b \geq 0\}$?

b. (od -1 do 3 punktów). Czy odpowiedź na pytanie z punktu a. zmieni się, jeśli przyjmiemy $S = \{[a,b] : b \geq 0, a \geq -b\}$?

---

rozwiązanie a)

Problem $STOP_{WALK-5a}$ dla automatów spacerujących po $S_{5a} = \{[a,b] : a,b \geq 0\}$jest *nierozstrzygalny.*

Pokażemy redukcję $STOP_{2CM} \leq_{rek} STOP_{WALK-5a}$, tzn. udowodnimy, że gdyby $STOP_{WALK-5a}$ był rozstrzygalny to potrafilibyśmy rozwiązać problem stopu dla maszyn z dwoma licznikami, a tym samym dla maszyn Turinga, który oczywiście jest nierozstrzygalny. Idea rozwiązania polega na przekształceniu maszyny w automat spacerujący i pokazaniu, że możliwości automatu wystarczają, aby zasymulować 2CM.

Maszyna z dwoma licznikami składa się z dwukierunkowej głowicy czytającej oraz dwóch liczników. Można pozbyć się taśmy wejściowej konwertując słowo w do postaci unarnej i zapisując je na pierwszym z liczników oraz przerabiając odpowiednio funkcję przejścia. Formalnie 2CM=$\langle Q_{2CM}, q_o, l_1, l_2, F, \delta\rangle$, $\delta : Q \times \{\phi, \bullet\}^2 \to Q \times \{+,-,0\}^2$.

1. Zbiór stanów automatu spacerującego będzie się składał się ze stanów Q_{2CM} oraz dodatkowych n stanów (n jest długością słowa w w postaci unarnej) służących do ustawienia automatu w odpowiedniej pozycji początkowej.
2. Wartości liczników l_1, l_2 będą odpowiadać współrzędnym $[x,y]$ automatu na płaszczyźnie.
3. Przekształcamy instrukcje $\delta(q, l_1, l_2) = (q', i_1, i_2)$ w instrukcje π:
   a) Dodaj odpowiedniki instrukcji wszystkim przejściom, z wyjątkiem tych w których $q \in Q_F$. Sprawdzenie niepustości pierwszego licznika odpowiada sprawdzeniu czy $[-1,0] \in s([x,y])$, analogicznie drugiego $[0,-1] \in s([x,y])$
   b) Dodaj puste przejścia dla każdej instrukcji nie kończącej się akceptacją:
   dla instrukcji δ w których $q \notin Q_F$ dodaj przejścia
   $\pi\left(q, P(T^2)\backslash\{[x,y] : x = l_1 = -1 \vee y = l_2 = -1\}\right) = (q, [0,0])$

Z powyższej konstrukcji wynika, że automat spacerujący zatrzyma się dokładnie wtedy, gdy odpowiadający mu 2CM przejdzie w stan $q \in Q_F$. □

rozwiązanie b)

Problem $STOP_{WALK-5b}$ dla automatów spacerujących po $S_{5b} = \{[a,b] : b \geq 0,\ a \geq -b\}$ jest *nierozstrzygalny.*

Możemy zasymulować automat spacerujący po S_{5a} za pomocą automatu spacerującego po S_{5b}.

1. Zbiory stanów pozostaną identyczne.
2. Współrzędne: $x_a = x_b$, $y_a = y_b + x_b$
3. Stworzymy bijekcję z funkcji π_b na funkcję π_a:
   a) Każdemu elementowi $s_a[x,y] \in P(T^2)$ przypiszemy jego odpowiednik,
   np. $s_a \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} = s_b \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$, $s_a \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} = s_b \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$, ...
   b) Podobnie wyznaczymy odpowiednik dla każdej wartości funkcji
   π_a : $\rightarrow$, $\searrow$, $\downarrow$, $\swarrow$, $\leftarrow$, $\nwarrow$, $\uparrow$, $\nearrow$
   π_b : $\searrow$, $\downarrow$, $\swarrow$, $\leftarrow$, $\nwarrow$, $\uparrow$, $\nearrow$, $\rightarrow$. □

2008.B.6 Automat skończony spacerujących po połówce

Czy problem stopu dla automatów skończonych spacerujących po S jest rozstrzygalny, jeśli $S = \{[a, b] : b \geq 0\}$?

Problem $STOP_{WALK-6}$ dla automatów spacerujących po $S_6 = \{[a, b] : b \geq 0\}$ jest *rozstrzygalny.*

Pierwszą rzeczą, którą możemy zauważyć jest fakt, że funkcja przejścia π nie zależy od pierwszej współrzędnej, dlatego możemy zredukować problem do automatu spacerującego po prostej $\pi'(q, b) = (p, x)$. Aby rozpoznać, że automat się nie zatrzyma musimy umieć sprawdzić:

- Czy automat osiągnie wysokość $|Q| + 2$?
  oznaczać to będzie, że nie zdoła już powrócić do poziomu zerowego.
- Czy spacerując po prostej nie wpadnie w powtarzalną sekwencję ruchów (czy nie zapętli się)?

Możemy rozstrzygnąć te warunki konstruując program symulujący działanie automatu i zapamiętujący, dla każdego stanu z Q wysokość (współrzędną b), na jakiej ten stan ostatnio odwiedziliśmy. Jeśli aktualna wysokość dla bieżącego stanu $\geq$ poprzednia - informujemy o tym, że automat nie zakończy obliczeń. Program ma złożoność $O\left(|Q|^2\right)$, ponieważ wartość dla każdego stanu zmienimy nie więcej niż $|Q|$ razy. □

2009.A.4 R.E. trudność

a) Udowodnij, że dla każdego zbioru rekurencyjnie przeliczalnego B zachodzi $B \leq_{rek} K$.

b) Jak nazywa się własność zbioru K opisana w punkcie **a.**?

rozwiązanie

a) Język B jest nie trudniejszy od K w sensie redukcji obliczalnych, jeśli istnieje redukcja $f : \mathbb{N} \to \mathbb{N}$ taka, że $\forall w \in B \iff f(w) \in K$.

Pokażemy taką redukcję dla każdego B:

- wczytaj w
- $b \leftarrow \langle \varphi_B(w) \rangle$
  (przypisz zmiennej b numer programu *rozpoznającego* przynależność w do B, ignorując swój własny argument)
- zwróć b

Skoro zbiór B jest rekurencyjnie przeliczalny to dla każdego $w \in B$ dostaniemy numer programu kończącego obliczenia, a więc należącego do K. Dla $w \notin B$ dostaniemy numer programu zapętlającego się, a więc nie należącego do K. □

b) K jest trudny w klasie języków rekurencyjnie przeliczalnych ze względu na redukcje będące funkcjami rekurencyjnymi, albo w skrócie K jest *r.e.-trudny.*[7]

[7] Akceptowaną odpowiedzią była także *"K jest r.e.-zupełny"*. W punkcie a) mowa jest o r.e.-trudności, natomiast r.e.-zupełność bierze się z punktu b) i tego, że K jest r.e.

2010.A.5 Redukcja, która jest„na"

Niech A, B będą podzbiorami zbioru liczb naturalnych. Załóżmy, że f jest redukcją świadczącą o tym, że $A \leq_{rek} B$. Załóżmy, że f jest „na" (tzn. jej obrazem jest cały zbiór liczb naturalnych). Pokaż, że w takim razie zachodzi również $B \leq_{rek} A$.

Funkcja f jest redukcją, tzn $\forall_{a \in \mathbb{N}}\, a \in A \Leftrightarrow f(a) \in B$. Funkcja f jest „na", ale nie wiemy, czy jest różnowartościowa, nie możemy zatem użyć f^{-1} jako redukcji do przeprowadzenia dowodu. Potrafimy za to skonstruować program znajdujący najmniejszy argument, dla którego funkcja f przyjmuje podaną wartość. Oto program obliczający funkcję g:

- wczytaj a
- $i \leftarrow 0$
- powtarzaj:
  - $i \leftarrow i + 1$
  - $b \leftarrow f(i)$
    dopóki $b \neq a$
- zwróć i

Funkcja g jest poprawną redukcją, ponieważ jest całkowita i obliczalna w sensie Turinga (rekurencyjna). Zauważmy, że program zawsze się zatrzyma, ponieważ w końcu zawsze znajdzie odpowiednią wartość f. Możemy zatem napisać, że $\forall_{a \in \mathbb{N}}\, a \in B \Leftrightarrow g(a) \in A$, a zatem $B \leq_{rek} A$ □

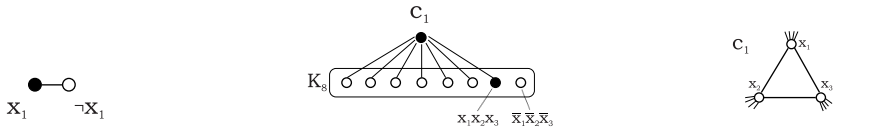
Część III – Teoria złożoności

2007.A.8 Problem P_{z8} [8]

Dane: $G = \langle V, E \rangle$, $A \subseteq V$. **Czy można wybrać** $B \subseteq A$ **tak aby:**

1. B był dominujący w G
2. B był niezależny
3. $\forall_{x \in V}$ istnieje co najwyżej jeden $y \in B$, że $E(x, y)$

Pokażemy, że $3SAT \leq_{rek} P_{z8}$, czyli dla danej formuły φ zbudujemy graf będący instancją problemu, który będzie spełniał warunki zadania wtedy i tylko wtedy, gdy φ jest spełnialna. Do konstrukcji użyjemy następujących gadżetów:



a) gadżet dla zmiennej x_1 (zwartościowanej na 1)

b) *kontroler* - gadżet dla klauzuli ($c_1 = x_1 \vee x_2 \vee x_3$, wartościowanie $x_1 = x_2 = x_3 = 1$)

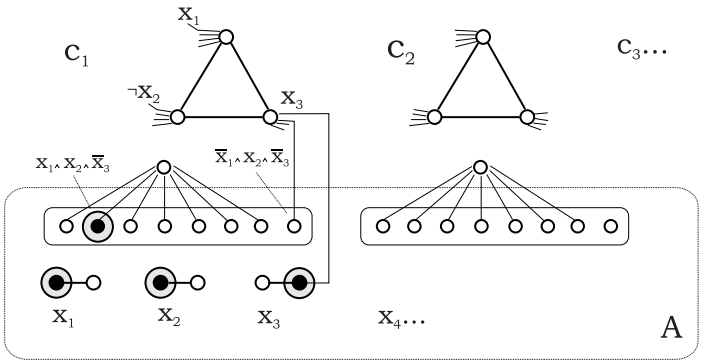
c) *trójkąt* - gadżet dla klauzuli

Dla każdego elementu formuły tworzymy instancje gadżetów - dla każdej zmiennej tworzymy po dwa połączone wierzchołki, a dla każdej klauzuli dwa gadżety: *kontroler* i *trójkąt*.

Kontroler jest kliką o rozmiarze dziewięć bez jednej krawędzi. Odpowiada on za sprawdzanie warunków prawdziwości klauzuli. Osiem wierzchołków indeksowanych jest wystąpieniami zmiennych ich negacjami (2^3) i interpretujemy je, jakby zmienne były połączone spójnikiem AND. Łączymy każdy taki wierzchołek z dziewiątym - c_i, z wyjątkiem jednego - tego, którego zmienne mają odwrotne znaki niż w klauzuli.

Trójkąt odpowiada oczywiście trzem literałom występującym w klauzuli. Każdy wierzchołek (indeksowany l_j) łączymy z literałem o przeciwnym znaku, a także z trzema dodatkowymi w klice kontrolera. W indeksach tych trzech wierzchołków literał l_j jest zanegowany, natomiast przynajmniej jeden z pośród dwóch pozostałych indeksów ma taki sam znak jak w rozpatrywanej klauzuli.

Jako zbiór A wybieramy wszystkie wierzchołki gadżetów zmiennych, a także po osiem wierzchołków każdego kontrolera połączonych w klikę.



Konstrukcja redukcji dla problemu P_{z8}

$\varphi = c_1 \wedge c_2 \wedge \cdots$, $c_1 = x_1 \vee \neg x_2 \vee x_3$, $x_1 = 1$, $x_2 = 1$, $x_3 = 0$

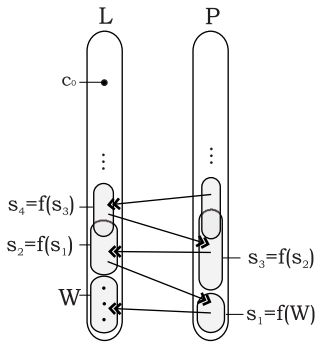
[8] Na liście z roku 2009 zadanie to znajduje się pod numerem 115.

1. Jeżeli formuła była spełnialna, to istnieje możliwość wybrania zbioru B prawidłowo. Do B wybieramy po jednym wierzchołku dla każdej zmiennej zgodnie z wartościowaniem, oraz po jednym wierzchołku dla każdego *kontrolera*. Takim, który po zwartościowaniu przyjmuje wartość true. Rozpatrując wierzchołki w A widzimy, że spełniają warunki zadania. Gdy przyjrzymy się *trójkątom*, także możemy zauważyć, że każdy wierzchołek sąsiaduje dokładnie z jednym z wierzchołków B. $\surd$
2. Jeśli udało się wybrać zbiór B prawidłowo, to możemy odczytać wartościowanie formuły φ. Do zbioru musiał trafić dokładnie jeden wierzchołek z gadżetu każdej zmiennej, a także odpowiedni wierzchołek z każdego kontrolera. Przy innym wyborze naruszylibyśmy warunki zadania. Musimy jeszcze zapewnić, że wartościowanie, które powstało spełnia formułę φ. Załóżmy, że tak nie jest. Oznaczałoby to, że przynajmniej jedna klauzula jest zwartościowana na 0, tzn. że wszystkie wierzchołki trójkąta są dominowane przez wierzchołki z gadżetów zmiennych. W takim wypadku w kontrolerze musiał być wybrany wierzchołek z etykietą, w której wszystkie literały są mają przeciwne znaki niż w klauzuli, lecz jest to jedyny wierzchołek, który nie ma połączenia z dziewiątym wierzchołkiem gadżetu, a więc zbiór B nie spełniał własności zadania. $\lightning$ $\square$

2008.A.8 Gra w kompromis

$G = \langle V, E \rangle$, $V = L \cup P$, $deg(G) \geq 2$

Pokażemy, że złożoność gry w kompromis jest w P poprzez pokazanie wielomianowego algorytmu odpowiadającego, czy istnieje strategia dojścia z wierzchołka c_o do W w ciągu $2|V|$ ruchów.



Załóżmy, że gracz L ma strategię wygrywającą i przyjrzyjmy się jak musiał wyglądać poprzedni ruch. Gracz P musiał trafić do W z pewnego skończonego zbioru wierzchołków, nazwijmy go s_1. Podobnie, idąc wstecz możemy wyznaczyć zbiór wierzchołków s_2, z których mógł przyjść gracz L.

Widzimy już, że możemy napisać funkcję $f(S) = \{v : \exists_{w \in V} (w, v) \in E(G)\}$ wyznaczającą dla danego zbioru wierzchołków zbiór leżący w drugiej części składowej składający się z tych wierzchołków, które mają krawędź skierowaną prowadzącą do jakiegoś $v \in S$. Funkcję możemy napisać tak, aby działała w czasie $O(m \cdot n)$. Jeżeli wyznaczając rekurencyjnie kolejne podzbiory, po wykonaniu $2|V|$ ruchów nie napotkamy wierzchołka c_0, oznacza to, że gracz P ma strategię wygrywającą. Jeżeli napotkaliśmy c_0 to oczywiście strategię wygrywającą ma gracz L. □

Zadania z ćwiczeń

Lemat o pompowaniu dla języków regularnych:
$\forall_{L-reg}\, \exists_{n\in N}\, \forall_{w\in L,\, |w|\geq n}\, \exists_{x,y,z\in\Sigma^*: w=xyz,\, |y|>0,\, |xy|\leq n}\, \forall_{k}\, xy^kz\in L$

Lemat o pompowaniu dla języków bezkontekstowych:
$\forall_{L\in CFL}\, \exists_{m=2^{|N|}}\, \forall_{w\in L,\, |w|\geq m}\, \exists_{s,t,v,x,y\in\Sigma^*: w=stvxy,\, |tvx|\leq m,\, |tx|>0}\, \forall_{k}\, st^kvx^ky\in L$

Zadanie 20. $w^n = v$

$L = \{w\, :\, \exists n\in\mathbb{N}, v\in R_{-reg}\, w^n = v\,\}$
$\mathbf{Q'} = Q^n \qquad \mathbf{q_0'} = \langle q_0, q_1, ..., q_n\rangle$ — startujemy „w każdym z możliwych stanów"
$\delta'\, : Q^n\times\Sigma\to Q^n = \langle \delta(r_0,a),\, \delta(r_1,a),\, ...,\, \delta(r_n,a)\rangle$
$\mathbf{F'} = \left\{\langle r_0, r_1, ..., r_n\rangle\, :\, \exists b_0, b_1, ..., b_c < n\, :\, b_0 = 0,\, \forall_{i<c}\, r_{b_i} = q_{b_{i+1}},\, q_{b_c}\in F\right\}$
A' akceptuje jeśli istnieje ciąg stanów $\langle r_{b_0}, r_{b_1}, ..., r_{b_c}\rangle$ tworzących „łańcuszek" przejścia od q_0 do $q\in F$
$\langle r_{b_0}, r_{b_1}, ..., r_{b_c}\rangle\, :\, r_{b_0} = \hat{\delta}(q_0, w),\, r_{b_i} = \hat{\delta}(q_{i-1}, w)$

Zadanie 22. $L_{/2}$

$L_{/2} = \{w\, :\, \exists v\in\Sigma^*, |w| = |v|\ vw\in L\,\}$
$Q\text{`} = Q^n\times 2^Q$ start w każdym ze stanów, pamiętamy też zbiór stanów osiągalnych w k krokach
$\delta\text{`}(\langle R, A\rangle, a) = \left\langle \langle \forall_{r\in R}\delta(r,a)\rangle\, ,\, \bigcup_{q\in A}\forall_{x\in\Sigma}\delta(q,x)\right\rangle$
$F\text{`} = \{\langle R, A\rangle\, :\, \exists_k\, r_k\in F\wedge q_k\in A\}$

Zadanie 29. $2n|w|_0 \leq |w|_1 \leq (2n+1)\,|w|_0$

Załóżmy nie wprost, że język $L = \{w\in\{0,1\}^*\, :\, \exists n\in\mathbb{N}\, 2n|w|_0\leq |w|_1\leq (2n+1)\,|w|_0\}$ jest bezkontekstowy. Istnieje zatem m — stała z lematu o pompowaniu dla CFL. Rozpatrzmy słowo $w = 1^{(2m+1)(2m+1)}0^{(2m+1)}$. Słowo to należy do L, istnieje zatem podział $w = stvxy$.

1. $tvx\in 1^*$ — po zastosowaniu operacji pompowania dla $k>1$ słowo wychodzi poza język na drugiej nierówności
2. $tvx\in 0^*$ — dla pewnego $k>1$ przestaje zachodzić pierwsza nierówność
3. $tvx\in 1^+0^+$— w tym przypadku za pomocą pompowania również możemy wyprowadzić słowo poza język.

   Niech $a = |tvx|_0$, $b = |tvx|_1$, niech $p\geq -1$ — ilość pompowań

   $2n(2m+1+pa)\ \leq\ (2m+1)(2m+1)+pb\ \leq\ (2n+1)(2m+1+pa)$

   $2n(2m+1)+2npa\ \leq\ (2m+1)^2+pb$

   $2npa - pb\ \leq\ (2m+1)^2 - 2n(2m+1)$

   $p\ \leq\ \frac{(2m+1)^2-2n(2m+1)}{2na-b}$ — widzimy, że p musi być ograniczone podaną wartością, ale oczywiście możemy pompować w nieskończoność, a więc sprzeczność ↯. □

Zadanie 32. $\neg\exists x \,:\, w = xx$

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

$S \to X \,|\, Y \,|\, Z \,|\, XY \,|\, XZ \,|\, YX \,|\, YZ \,|\, ZX \,|\, ZY$
$X \to 0 \,|\, x_1 X\, x_2 \quad \langle x_1, x_2\rangle \in \{0,1,2\} \times \{0,1,2\}$
$Y \to 1 \,|\, y_1 Y\, y_2$
$Z \to 2 \,|\, z_1 Z\, z_2$

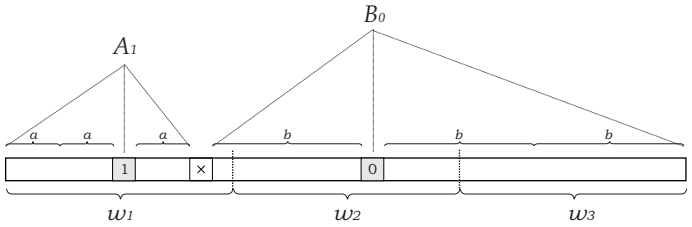
1. $L(G) \subseteq L$
   a) $S \to X \,|\, Y \,|\, Z$ — długość jest nieparzysta, więc ok
   b) $S \to XY \quad w = x_1 0\, x_2 y_1 1\, y_2 \qquad |x_1| = |x_2| = i,\ |y_1| = |y_2| = j$
   przy podziale na 2 podsłowa równej długości słowa
   zawsze będą się różnić na wyróżnionej pozycji
2. $L \subseteq L(G)$
   weźmy najkrótsze słowo niewyprowadzalne z G
   a) $|w|$ — nieparzysta — wyprowadzamy z pojedynczego nietermianala
   b) $|w|$ — parzysta $w = x_1 0\, x_2 y_1 1\, y_2$- możemy wyprowadzić z dwóch nieterminali

Zadanie 33. $\exists x \,:\, w = xx$

$L = \{w \in \{0,1,2\}^* :\ \exists x \in \{0,1,2\}^*\ \ w = xx\}$

$L \cap L_{\underline{0^*10^*1}\,\underline{0^*10^*1}}$ — nie jest regularny

Zadanie 34. $L_{n3w} = \{w \in \Sigma^* \setminus \{www \,:\, w \in \Sigma^*\}\},\ \Sigma = \{0,1\}$



Do języka L_{n3w} należą wszystkie słowa z Σ^* z wyjątkiem tych, które można przedstawić jako konkatenację trzech jednakowych podsłów. Rozważmy słowa o długości podzielnej przez 3, bo pozostałe należą do L_{n3w}. Aby w należało do języka dwa sąsiadujące podsłowa muszą się różnić na przynajmniej jednej pozycji.

Gramatyka G:

$S \to N \,|\, A_0 X B_1 \,|\, A_1 X B_0 \qquad X \to 0|1$

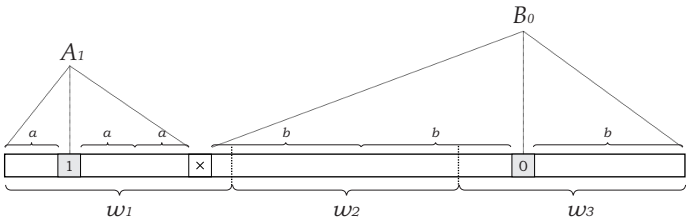
$N \to XXXN \,|\, XX \,|\, X \qquad$ — *$|w|$ nie dzieli się przez 3*

$A_0 \to XXA_0X \,|0 \qquad A_1 \to XXA_1X \,|1$

$B_0 \to XB_0XX \,|0 \qquad B_1 \to XB_1XX \,|1$

1. Każde słowo wyprowadzone z G należy do L_{n3w}. Załóżmy nie wprost, że tak nie jest, zatem wyprowadziliśmy $w = w_1w_2w_3$ takie, że $w_1 = w_2 = w_3$. Wyprowadzając słowo musieliśmy skorzystać z jednego z wyprowadzeń zawierających A_x i B_y, $x \neq y$, zatem na pozycji $2a+1$ będzie symbol x, a na pozycji $3a+b+3$ symbol y, zatem $w_1 \neq w_2$ lub $w_2 \neq w_3$. ↯
2. Z G można wyprowadzić wszystkie słowa z L_{n3w}. Załóżmy nie wprost, że istnieje słowo $w \in L_{n3w}$, którego nie da się wyprowadzić. Szybko dochodzimy do sprzeczności, gdyż potrafimy pokazać wyprowadzenie dla każdego słowa, jeśli tylko $3 \nmid |w|$ lub jeśli podsłowa różnią się na odpowiedniej pozycji. □

Zadanie 37. $L_{nwvw} = \{w \in \Sigma^* \setminus \{wvw : w, v \in \Sigma^*, |w| = |v|\}\}$, $\Sigma = \{0, 1\}$



$S \to N \mid A_0XB_1 \mid A_1XB_0 \qquad X \to 0|1$

$N \to XXXN \mid XX \mid X \qquad$ — $|w|$ nie dzieli się przez 3

$A_0 \to XXA_0X \mid 0 \qquad A_1 \to XXA_1X \mid 1$

$B_0 \to XB_0XX \mid 0 \qquad B_1 \to XB_1XX \mid 1$

Dowód analogicznie jak w zadaniu 34.

Zadanie 100. $CLIQUE_{n/2}$

1. Pokazujemy redukcję $3SAT \leq_{rek} CLIQUE_n$ [9]
   - Dla formuły Φ o k klauzulach i l zmiennych tworzymy graf o liczbie wierzchołków $= 3 \cdot k$ i o krawędziach pomiędzy wszystkimi wierzchołkami, z wyjątkiem a) wierzchołków z jednej *trójki*, b) wierzchołków odpowiadających przeciwnym literałom, np. x i $\neg x$.
   - Klika o k wierzchołkach istnieje wtw. gdy istnieje poprawne wartościowanie formuły Φ.
2. Pokazujemy redukcję $CLIQUE_n \leq_{rek} CLIQUE_{n/2}$. Dla $\langle G, k\rangle$ tworzymy G'. [10]
   - jeśli $k = m/2$ to $G' = G$
   - jeśli $k < m/2$ to $G' = G \cup X$, $X =$ zbiór $m - 2k$ wierzchołków połączonych ze wszystkimi pozostałymi
   - jeśli $k > m/2$ to $G' = G \cup X$, $X =$ zbiór $2k - m$ izolowanych wierzchołków □

[9] Sipser, tw. 7.11 (str. 306)
[10] Sipser, zad 7.22 (str. 336)

Zadanie 102. Spełnialność 9/10 klauzul

Pokażemy redukcję $SAT \leq_P SAT_{9/10}$, to znaczy funkcję $f : \langle \varphi \in CNF \rangle \to \langle \phi \in CNF \rangle$

- n - liczba klauzul w φ
- $\phi = \varphi \wedge y_1 \wedge \neg y_1 \wedge y_2 \wedge \neg y_2 \wedge ... \wedge y_k \wedge \neg y_k$
- $\underbrace{n+k}_{\text{liczba spełnionych klauzul}} = 9/10 \cdot (n + 2k) + 1$
- $k = \lceil \frac{n-10}{8} \rceil$

1. Jeśli φ jest spełnialna, to ϕ jest spełnialna w $SAT_{9/10}$. Wynika to z konstrukcji ϕ - będą w niej spełnione wszystkie klauzule z φ, oraz dokładnie połowa z dodatkowych zmiennych.
2. Jeśli ϕ jest spełnialna w $SAT_{9/10}$ to φ jest spełnialna. Jeśli $9/10$ klauzul z ϕ przyjmuje wartość 1 oznacza to, że wszystkie klauzule z φ muszą być spełnione, bo jedynie połowa dodatkowych zmiennych przyjmie wartość 1. □

Zadanie 105. Problem smutnych strażników

$S = \{s_1, s_2, ..., s_l\}$ - strażnicy

$A = \{a_1, a_2, ..., a_k\}$ - obiekty

$Z_{E_i}, Z_{F_i\, i \in \{1..l\}} \subseteq A$ - obiekty obserwowane na ekranach

Pokażemy redukcję $SAT \leq_P GUARD$, to znaczy funkcję $f : \langle \varphi \in CNF \rangle \to \langle S, A, Z_E, Z_F \rangle$

- zbiorem S będą zmienne występujące w φ
- zbiorem A będą klauzule występujące w φ
- dla każdego literału w formule φ:
  - jeśli zmienna x_i występuje w klauzuli j-tej w postaci niezanegowanej, to dołączamy a_j do zbioru Z_{E_i},
  - jeśli natomiast występuje w postaci zanegowanej to do Z_{F_i}.

W ten sposób otrzymaliśmy instancję problemu $GUARD$, która ma rozwiązanie wtedy i tylko wtedy, gdy φ jest spełnialna. Dowód oczywisty. □

Zadanie 118. $TOTAL_{RE} \in PSPACE$

Zbudujemy niedeterministyczną maszynę Turinga M:

1. czytaj r będące wyrażeniem regularnym i zamień je na NFA A
2. na taśmie zapisz stany A z zaznaczonym stanem początkowym
3. powtarzaj $2^{|Q|}$ razy: (niedeterministycznie zgadujemy słowo w, którego A nie zaakceptuje)
   - zgadnij kolejny symbol słowa w
   - zmień wartości stanów $Q(A)$ na taśmie zgodnie z funkcją przejścia A
   - jeśli wśród stanów $Q(A)$ nie ma żadnego stanu akceptującego *odrzuć*
4. nie znaleźliśmy kontrprzykładu, więc *zaakceptuj*

Jeśli istnieje jakieś słowo, którego A nie zaakceptuje, to istnieje także słowo o długości $\leq 2^{|Q|}$ - wynika to z lematu o pompowaniu. M działa w czasie wykładniczym, ale w pamięci liniowej niedeterministycznej, więc $TOTAL_{RE} \in NSPACE$. Z twierdzenia Savitcha wiemy jednak, że $NSPACE\left(f(n)\right) \subseteq PSPACE\left(f^2(n)\right)$, to znaczy, że deterministycznie problem da się rozstrzygnąć w pamięci kwadratowej. □