Języki formalne i złożoność obliczeniowa oswojone

rozwiązania opracował
Arkadiusz Janicki <arek@wywrota.pl>
wersja z dnia 28. grudnia 2009 r.
Instytut Informatyki, Uniwersytet Wrocławski

Contents

Wstęp

Niniejszy zbiór powstał 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. Dołączyłem także rozwiązanie jednego ciekawego zadania z książki M. Sipsera, oraz szkice kilku rozwiązań zadań z ćwiczeń.

Decyzja o publikacji tego zbioru była dla mnie sporym problemem moralnym. 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 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.

Zauważmy, że żaden z tych punktów nie przewiduje zaglądania do gotowych rozwiązań. Nie zdradzę jednak żadnej tajemnicy, jeśli powiem, że takie praktyki zawsze miały miejsce, tak samo jak studenci od zawsze ściągali na egzaminach. Trzeba te działania oczywiście uznawać za naganne. Osobiście odradzam studentom zaglądania do tych rozwiązań przed rzetelnym przejściem przez powyższe trzy punkty. Aby nie uśpić waszej czujności ostrzegam, że jedno z zadań jest rozwiązane nieprawidłowo. Postanowiłem pozostawić je w takiej formie i to traktuję jako swoje rozgrzeszenie.

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)

Zadania z egzaminów

2003.B.4 $f, g, h : \{0, 1, ..., p\} \to \{0, 1, ..., p\}$ [1]

Dane: $f, g, h : \{0, 1, ..., p\} \to \{0, 1, ..., p\}$ oraz skończone ciagi $(b_1, b_2, ..., b_k)$ i $(c_1, c_2, ..., c_k)$. Czy istnieje liczba iteracji n taka, że $F^n_{f,g,h}(b_1, b_2, ..., b_k, 0, 0, ...) = (c_1, c_2, ..., c_l, 0, 0, ...)$?

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.
- Wybieżmy 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	 ...	

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. ↯. □

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

2005.B.6 programy działające w czasie kwadratowym

Niech A będzie zdefiniowanym w zadaniu zbiorem numerów maszyn Turinga. Załóżmy, 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. $\lightning$ $\square$

2006.A.4 Automat machający dwiema nogami

Automat niedeterministycznie machający dwiema nogami (2LEGS)

Czy problem niepustości automatu z dwiema nogami jest rozstrzygalny?

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 $2LEGS = \langle \Sigma, Q, q_0, F, \delta \rangle$

- $Q = \{1, 2, ..., k\} \times \left\{ 0, 1, ..., \underset{1 \leq i \leq k}{max} |w_i| \right\} \times \left\{ 0, 1, ..., \underset{1 \leq i \leq k}{max} |v_i| \right\} \cup \{q_0, q_{ok}, q_{accept}, q_{reject}\}$
  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.[2]
  - $\big\langle \langle i, 0, 1 \rangle$, $*$, $b_{|w_i|}$, $q \in \{q_{ok}, q_{accept}\}$, 0, $1\big\rangle$ - 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.

□

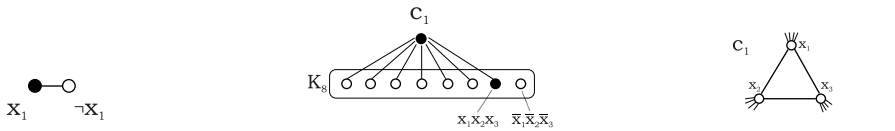
[2] Na potrzeby przykładu zakładamy, że słowa w_i, v_i są niepuste. Oczywiście bez tego założenia również możemy skonstruować redukcję.

2007.A.8 Problem P_{z8} [3]

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 $x_1 = x_2 = x_3 = 1$)

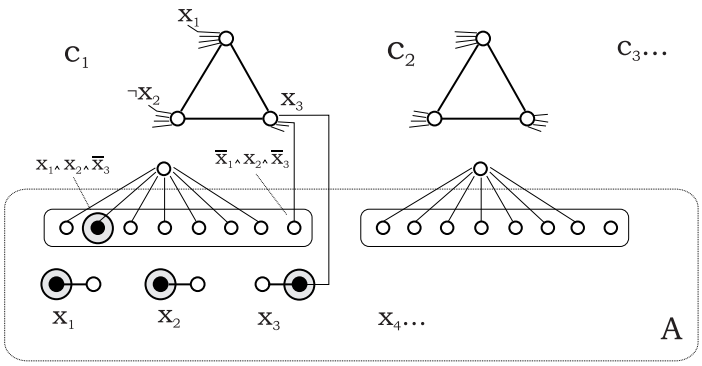
c) *trójkąt* - gadżet dla klauzuli

Dla każdego elementu formuły twożymy 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}

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

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

1. Jeżeli formuła była spełnialna, to istnieje możliwośc 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 taże odpowiedni wierzhołek 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.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 odnaleźc 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

Problem stopu dla automatów skończonych spacerujących po S

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
   $\pi\left(q, P(T^2) \setminus \{[x,y] : x = l_1 = -1 \vee y = l_2 = -1\}\right) = (q, [0,0])$

Z powyższej kontrukcji 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

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 pierweszej 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. □

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 $\overline{K}$[4], 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*[5] 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.
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 $\overline{K}$ ↯. □

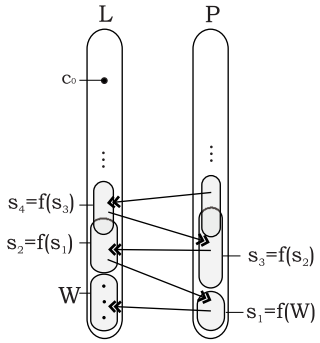
[4] $\overline{K}$ - czyli zbiór numerów tych programów, które uruchomione dla argumentu będącego numerem uruchamianego programu zapętlają się.

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

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.



Zakłóżmy, 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. □

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*.[6]

[6] 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.

Zadania z ćwiczeń

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

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

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

1. $L(G) \subseteq L$
   (a) $S \to X \mid Y \mid Z$ - 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 pojedyńczego nietermianala
   (b) |w| - parzysta $w = x_1 0\, x_2 y_1 1\, y_2$- możemy wypr z 2 nieterminali

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

1. $L = \{w \in \{0,1,2\}^* : \exists x \in \{0,1,2\}^*\ w = xx\}$
   $L \cap L_{\underline{0^*10^*1}\,\underline{0^*10^*1}}$- nie jest regularny

Zadanie 100. $CLIQUE_{n/2}$

1. Pokazujemy redukcję $3SAT \leq_{rek} CLIQUE_n$ [7]
   - 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'. [8]
   - 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 □

Zadanie 102. Spełnialność 9/10 klauzul

Pokażemy reduckję $SAT \leq_P SAT_{9/10}$, 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:

[7] Sipser, tw. 7.11 (str. 306)
[8] Sipser, zad 7.22 (str. 336)

– 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 117. $KOAL_2$

1. $INDEPENDENT - SET_{n/4} \leq_P INDEPENDENT - SET_{n/2}$

   Redukcja $f : \langle G, n/4\rangle \to \langle G', n'/2\rangle$ polega na dodaniu do grafu G $\frac{n}{2}$ izolowanych wierzchołków.

   - Jeśli w G był zbiór niezależny o wielkości $\frac{n}{4}$, to w G' również on się pojawi, powiększony o $\frac{n}{2}$ nowych wierzchołków. Rozmiar tego zbioru wyniesie $\frac{3}{4}n$, a więc $\frac{1}{2}n'$.
   - Jeśli w skonstruowanym grafie G' istnieje zbiór niezależny o wielkości przynajmniej $\frac{3n}{4}$, oznacza to, że w podgrafie nie zawierającym dodanych wierzchołków musiał istnieć podzbiór o $\frac{n}{4}$ wierzchołkach. √

2. $INDEPENDENT - SET_{n/2} \leq_P KOAL_2$

   Konstruujemy redukcję $f' : \langle G, n/2\rangle \to \langle Partie, Stanowiska\rangle$. Zbiorem $Partie$ będzie zbiór wierzchołków G, zbiorem $Stanowiska$ zaś zbiór krawędzi. Jeśli krawędź istnieje pomiędzy dwoma wierzchołkami oznacza to, że odpowiadające im partie ubiegają się o to samo stanowisko. Dodatkowo jeśli liczba wierzchołków G jest parzysta dodajemy jedną partię, nieubiegającą się o żadne stanowisko. (Przyjmujemy, że każda partia dysponuje jednakową liczbą mandatów, równą np. 2).

   - Jeśli w G był zbiór niezależny $\frac{n}{2}$ oznacza to, że wybierając partie odpowiadające wierzchołkom tego zbioru (i ewentualnie 1 wirtualną) uda się stworzyć większościową koalicję.
   - Jeśli istnieje koalicja, to w odpowiadającym modelu naturalnie istnieje także zbiór niezależny wielkości $\frac{n}{2}$. □

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. □

Zadania z książki Michaela Sipsera

Zadanie 3.20

Pokaż, że jednostronna TM, która nie może zapisywać na fragmencie taśmy zajmowanym przez słowo wejściowe rozpoznaje tylko języki regularne.

Zauważmy, że zachowanie, a więc i wynik zwrócony przez maszynę są zdeterminowane jedynie przez to, jak zachowuje się ona na słowie wejściowym, a dokładniej przez następujące wartości (tworzące swoistą "funkcję przejścia" na słowie wejściowym):

- stan w którym po raz pierwszy opuszcza słowo wejściowe
- dla każdego stanu w którym ponownie wchodzi od prawej strony na słowo wejściowe: stan w którym je potem opuszcza (lub ewentualnie informacja, że akceptuje mając głowicę gdzieś w środku)

Różnych zachowań maszyny jest więc $O\left(|Q|^2\right)$, tj. skończona liczba - zatem jest tylko skończona liczba klas słów na których maszyna różnie się zachowuje, zaś w obrębie klasy (modulo to ile dokładnie chodzi po wejściu nim je opuści za każdym razem) maszyna zachowuje się dokładnie tak samo.

Powyższe nie oznacza jeszcze, że rozpoznawany język jest regularny - te klasy muszą mieć odpowiednie własności. Ale można zauważyć, że dwukierunkowy automat skończony jest w stanie symulując maszynę "odtworzyć" "funkcję przejścia" opisaną w punkcie (1), tj. stwierdzić do której klasy należy słowo na wejściu. Nie wiemy co prawda dla każdej z klas czy maszyna akceptuje słowa z tej klasy czy nie (może się w dziwny sposób zapętlać), ale różnych możliwości jest skończona liczba $\left(2^{\#klas}\right)$ - dla każdej z nich umiemy stworzyć odpowiedni dwukierunkowy automat.

Dwukierunkowe automaty są równoważne zwykłymi, dowód jest gdzieś w książce Hopcrofta i Ullmanna[9]. □

pomoc przy rozwiązaniu: Artur i Łukasz Jeż

[9] lub tu: Shepherdson, J., "The Reduction of Two-Way Automata to One-Way Automata," IBM Journal of Research and Development, 3 (1959), 198–200.