1 Wykład 1 (28/02/2008)

1.1 Deterministyczne automaty skończone

Definicja Deterministyczny automat skończony (DFA) to:

- $M = \langle Q, \Sigma, \delta, q_0, F \rangle$;
- Q: skończony zbiór stanów;
- Σ: skończony alfabet wejściowy;
- $q_0 \in Q$: stan początkowy;
- $F \subseteq Q$: zbiór stanów końcowych (akceptujących);
- $\delta : Q \times \Sigma \to Q$: funkcja przejścia.

Co to znaczy, że słowo w jest akceptowane przez automat M? W przypadku słów jednoliterowych ($w \in \Sigma$) jest tak gdy $\delta(q0, w) \in F$.

Zdefiniujmy $\hat{\delta} : Q \times \Sigma^* \to Q$ jako:

- $\hat{\delta}(q, \varepsilon) = q$;
- $\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$.

Słowo w jest akceptowane przez M gdy $\hat{\delta}(q_0, w) \in F$.

Definicja Zbiór słów akceptowanych przez M (język nad M):

- $L(M) = \left\{ w : \hat{\delta}(q, w) \in F \right\}$

1.2 Lemat o pompowaniu

Dla każdego języka regularnego L istnieje taka stała $n \in \mathbb{N}$ (zwana "stałą z lematu o pompowaniu"), że dla każdego takiego słowa $z \in L$, że $|z| \geqslant n$, istnieje taki podział $z = uvw$, że $|uv| \leqslant n, |v| \geqslant 1$ i dla każdego $k \in \mathbb{N}$ słowo $uv^k w \in L$.

Język L jest regularny jeśli istnieje akceptujący go DFA.

Przykład Czy język L słów, w których liczba 0 jest równa liczbie 1, jest regularny?

Weźmy $z = 0^n 1^n$. Dla $|uv| \leqslant n$ w dowolnym podziale $z = uvw$ słowo uv składa się z samych 0, zatem v również składa się z samych 0, więc uv^2w zawiera więcej 0 niż 1, więc nie należy do L. Zatem L nie jest regularny.

Przykład Czy $L = \left\{ (0^{i^2}) : i \in \mathbb{N} \right\}$ jest regularny?

Załóżmy, że jest. W takim razie bierzemy n z lematu o pompowaniu. Weźmy $z = 0^{n^2}, z \in L$. Podzielmy $z = uvw$ tak, żeby $|uv| \leqslant n$. Skoro $|uv| \leqslant n$, to $|v| \leqslant n$, więc $\left|uv^2w\right| = |uvw| + |v| \leqslant n^2 + n < (n+1)^2$. Jednocześnie $n^2 < \left|uv^2w\right|$. Zatem L nie jest regularny.

1.3 Wyrażenia regularne

Definicja $L_1 L_2 = \{uv : u \in L_1, v \in L_2\}$

- $L^0 = \{\varepsilon\}$;
- $L^1 = L$;
- $L^i = LL^{i-1}$, dla $i \geqslant 2$.
- $L^* = \bigcup_{i=0}^{\infty} L^i$

- $\varnothing$ oznacza język pusty;
- ε oznacza język $\{\varepsilon\}$;
- a oznacza język $\{a\}, a \in \Sigma$;
- jeśli r, s są wyrażeniami regularnymi określającymi języki R, S, to $r + s$ oznacza $R \cup S$, rs oznacza RS, r^* oznacza R^*.

Przykład Weźmy $L = \{00, 000, 111\}$. Odpowiada mu wyrażenie $00 + 000 + 111$.

Przykład Język, w którym występują gdzieś trzy zera:

- $(1 + 0)^*000(1 + 0)^*$

Przykład Język składający się ze słów, których długość jest podzielna przez 3 lub 5:

- $(000)^* + (00000)^*$.

Przykład Język składający się ze słów, w których liczba 0 jest podzielna przez 3:

- $(1^*01^*01^*01^*)^* + 1^*$.

2 Wykład 2 (4/03/2008)

Wykład zaczął się pod złymi auspicjami.

JMA: Słychać mnie?

sala: Nie.

Plan wykładu:

1. Teoria języków formalnych:
   - języki regularne
   - automaty skończone (deterministyczne lub nie)
   - wyrażenia regularne
   - języki bezkontekstowe
   - automaty ze stosem (deterministyczne lub nie)
   - niedeterminizm (centralne pojęcie teorii informatyki)
2. Elementy teorii rekursji:
   - zbiór rekurencyjny/rozstrzygalny/obliczalny
   - zbiór rekurencyjnie przeliczalny (recursively enumerable)
   - funkcja rekurencyjna
3. Teoria złożoności obliczeniowej:
   - PTIME
   - NP (nondeterministic polynomial)
   - PSPACE
   - EXPTIME
   - nieelementarne
   - nierekurencyjne
   - nie r.e.

Literatura:

- Hopcroft & Ullman
- Kozen

- Papadimtriou

Definicja Język:

- podzbiór $\{0,1\}^*$, innymi słowy podzbiór $\mathbb{N}$;
- albo podzbiór A^* dla jakiegoś skończonego alfabetu A.

JMA: To nie może być nic trudnego, skoro wymyślono to w latach trzydziestych dwudziestego wieku. Ludzkość nie znała wtedy nawet odciekacza do sałaty.

Definicja Deterministyczny automat skończony (DFA) to krotka $\langle \Sigma, Q, q_0, F, \delta \rangle$.

Definicja Wyrażenie regularne:

- $\varnothing$ jest w.r., $L_\varnothing = \varnothing$;
- ε jest w.r., $L_\varepsilon = \{\varepsilon\}$;
- jeśli $a \in \Sigma$, to a jest w.r., $L_a = \{a\}$;
- jeśli φ, ψ są w.r., to $\varphi\psi, \varphi + \psi$ są w.r., $L_{\varphi}\psi = L_\varphi L_\psi, L_\varphi + \psi = L_\varphi \cup L_\psi$;
- jeśli L jest językiem, to L^* to najmniejszy język taki, że $\Sigma \in L^*, L \subseteq L_*$, oraz jeśli $w, v \in L^*$, to $wv \in L^*$;
- jeśli φ jest w.r., to φ^* jest w.r. oraz $L_{\varphi^*} = L_\varphi^*$.

Definicja Język regularny to taki, że istnieje automat skończony, który go rozpoznaje.

Twierdzenie Dla każdego języka regularnego L istnieje wyrażenie regularne φ takie, że $L = L_\varphi$.

Dowód Skoro L jest regularny, to istnieje taki automat $M = \langle \Sigma, Q, q_0, F, \delta \rangle$, że $L = L(M)$. Niech $Q = \{q_0, q_1, ..., q_{N-1}\}$.

(obrazek 2.1)

Dla każdej trójki liczb $0 \leqslant i, j, k \in N$ napiszemy wyrażenie $\psi_{i,j,k}$ oznaczające zbiory tych $w \in \Sigma^*$, że:

1. $\hat{\delta} q_i, w = q_j$;
2. dla każdego niepustego, właściwego prefiksu v słowa w jeśli $\hat{\delta}(q_i, v) = q_m$, to $m < k$.

Najpierw napiszemy:

- $\psi_{i,j,0} = \sum_{a \in \Sigma \cup \{\varepsilon\}, \hat{\delta} q_i, a = q_j} a$

Skoro nie może być żadnych pośrednich stanów, to mogą wystąpić tylko jednoliterowe słowa.

A teraz napiszemy:

- $\psi_{i,j,k+1} = \psi_{i,j,k} + \psi_{i,k,k}\psi^*_{k,k,k}\psi_{k,j,k}$

Są to słowa, które albo nie wchodzą do q_k, albo wchodzą, krążą poniżej q_k, a następnie wracają i idą do q_j.

(obrazek 2.2)

$\varphi = \sum_{q_f \in F} \psi_{0,f,N}$

3 Wykład 3 (6/03/2008)

3.1 Niedeterminizm

Definicja Niedeterministyczny automat skończony (NDFA) to krotka $\langle \Sigma, Q, q_0, F, \delta \rangle$, gdzie δ jest relacją przejścia.

W DFA $\delta : Q \times \Sigma \to Q$, zaś w NDFA $\delta \subseteq Q \times \Sigma \times Q$.

Przykład Rozważmy $L = \{w \in 0^* : 35 \not| \, |w|\}$. Jest to język regularny.

(obrazek 2.3)

(obrazek 2.4)

Dla słowa składającego się z 14 zer, doradca każe iść w prawo. Dla słowa składającego się z 15 zer, doradca każe iść w lewo. Dla słów o długości dzielącej się przez 35, doradca nie ma możliwości oszukania robaczka.

Przykład Słowa, które na 20 pozycji od końca mają 0.

(obrazek 2.5)

Robaczkowi głos z kratki podpowiada, które to jest to właściwe zero. Jednakże robaczek jest zabezpieczony również przed podłymi głosami z kratki.

Czy niedeterminizm coś zmienia? To zależy, jakim się jest zwierzątkiem.

JMA: Fusia proponuje, że od pewnego momentu naszego nieskończonego życia jesteśmy szczęśliwi. Właściwie to ona powinna mieć na imię Nirvanka.

Dygresja Automaty skończone działające na nieskończonych słowach ma przebieg akceptujący wtedy, kiedy nigdy nie jest w stanie akceptujący ostatni raz.

Przykład $L = \left\{w \in \{0,1\}^{\mathbb{N}} : \text{liczba jedynek w } w \text{ jest skończona}\right\}$ Nie ma deterministycznego automatu, który rozpoznałby ten język.

(obrazek 2.6)

JMA: Jeśli jedynek jest skończenie wiele, to zer musi być nieskończenie wiele, czyli w szczególności przynajmniej jedno.

Twierdzenie Każdy język rozpoznawany przez pewien NDFA jest regularny.

Dowód Niech $M = \langle \Sigma, Q, q_0, F, \delta \rangle$ będzie NDFA. Zbudujemy taki DFA $M' = \langle \Sigma, Q', q_0', F', \delta' \rangle$, że $L(M') = L(M)$. Zbudujemy go z M:

- $Q' = 2^Q$ (zbiór potęgowy)
- $q_0' = \{q_0\}$
- $\delta' : Q' \times \Sigma \to Q', q \subset Q$
- $\delta'(q', a) = \{q : \exists_{q'' \in q'} \, \delta(q'', a, q)\}$
- $F' = q' : q' \cap F \neq \varnothing$

Twierdzenie Niech $L \subseteq A^*$. Następujące 3 warunki są równoważne:

1. istnieje DFA M taki, że $L(M) = L$;
2. istnieje NDFA M taki, że $L(M) = L$;
3. istnieje wyrażenie regularne φ taki, że $L_\varphi = L$.

Udowodniliśmy, że (1) $\Leftrightarrow$ (2) i (1) $\Rightarrow$ (3). Pokażemy, że (3) $\Rightarrow$ (2).

Definicja NDFA z ε-przejściami rozpoznają tę samą klasę języków, co NDFA.

(obrazek 2.7)

Dla wyrażenie φ zdefiniujemy NDFA M_φ taki, że $L_\varphi = L(M_\varphi)$. Konstrukcja indukcyjna po strukturze wyrażenia φ.

(obrazek 2.8)

1. $\varnothing$
2. ε
3. a

4. $\varphi = \varphi_1\varphi_2$
5. $\varphi = \varphi_1 + \varphi_2$
6. $\varphi = \varphi_1^*$

4 Wykład 4 (11/03/2008)

4.1 Gramatyki bezkontekstowe

Definicja Gramatyka bezkontekstowa to krotka $\langle \Sigma, N, S, \Pi \rangle$, gdzie:

- Σ to alfabet symboli terminalnych;
- N to alfabet symboli nieterminalnych, $\Sigma \cap N = \varnothing$;
- S to symbol początkowy, $S \in N$;
- Π to skończony zbiór produkcji, czyli $\Pi \subseteq N \times (N \cup \Sigma)^*$.

Wyobrażamy sobie teraz, że $\Pi \subseteq \Omega^* \times \Omega^*$, gdzie $\Omega = N \cup \Sigma$.

Niech $w, v \in \Omega^*$. Wtedy $w \to_\Pi v$ jeśli istnieją takie słowa $w_1, w_2 \in \Omega^*, \langle v_1, v_2 \rangle \in \Pi$, że $w = w_1 v_1 w_2, v = w_1 v_2 w_2$.

(obrazek 4.1)

Mówimy, że w przepisuje się w 1 kroku do v przy pomocy Π.

Relacja $\to_\Pi^*$ jest tranzytywnym domknięciem relacji $\to_\Pi$.

Definicja Niech $G = \langle \Sigma, N, S, \Pi \rangle$ to gramatyka bezkontekstowa (CFG). Zbiór słów definiowanych tą gramatyką to $L(G) = \{w \in \Sigma^* : S \to_\Pi^* w\}$.

Definicja (na potrzeby wewnętrzne) Zbiór wszystkich słów zawierających terminale i nieterminale to $L'(G) = \{w \in (\Sigma \cup S)^* : s \to_\Pi^* w\}$.

Definicja L nazywamy językiem bezkontekstowym (CFL), jeśli istnieje gramatyka bezkontekstowa G taka, że $L = L(G)$.

Uwaga Równość języków bezkontekstowych interesuje nas z dokładnością do słowa pustego.

Przykład (bardzo łatwy) Język palindromów:

- $L = \left\{w \in \{0,1\}^* : w = w^R\right\}$

Jest on bezkontekstowy, ale nie regularny, co łatwo dowieść z lematu o pompowaniu przy użyciu słowa postaci $000\ldots1\ldots000$.

Gramatyka:

- $S \to 0S0 \mid 1S1 \mid 0 \mid 1 \mid \varepsilon$

JMA: To było na programowaniu? I były takie wykresy? To co, on nie ma czego uczyć tam?

Definicja Zapis $|w|_0$ oznacza liczbę zer w słowie w:

- $|\varepsilon|_0 = 0$;
- $|0w|_0 = 1 + |w|_0$;
- $|1w|_0 = |w|_0$.

Przykład (łatwy) Język słów o tej samej liczbie zer i jedynek:

- $L = \left\{w \in \{0,1\}^* : |w|_0 = |w|_1, 2 \mid |w|_0\right\}$

Gramatyka:

- $S \to SS \mid TT \mid 1T0 \mid 0T1 \mid \varepsilon$
- $T \to ST \mid TS \mid 1S0 \mid 0S1$

Chcemy pokazać, że $L = L(G)$. Pokażemy, że (1) $L(G) \subseteq L$ i (2) $L \subseteq L(G)$.

Dowód (1) Niech $L' = \{w \in \{0,1,S,T\}* : 2 \mid |w|_0 + |w|_T, |w|_0 = |w|_1\}$.

Wystarczy, że pokażemy, że $L'(G) \subseteq L'$, bo $L = L' \cap \{0,1\}^*$, zaś $L(G) = L'(G) \cap \{0,1\}^*$. Wobec tego wystarczy, że dla każdego $k \in \mathbb{N}$ jeśli $S \to_\Pi^k w$, to $w \in L'$.

Niech $k = 0$. Wtedy $w = S$, zaś $S \in L'$. Załóżmy, że teza jest prawdziwa dla pewnego k.

(obrazek 4.2)

Dowód (2) Chcemy pokazać, że $L \subseteq L(G)$. Niech $L' = \{w \in 0, 1^* : |w|_0 = |w|_1, 2 \not| \, |w|_0\}$. Niech G' będzie taka sama, jak G, tylko że symbolem początkowym będzie T.

Wystarczy, że pokażemy, że $L \in L(G)$ i $L' \in L(G')$. Załóżmy, że tak nie jest, czyli że zbiór $K = (L \setminus L(G)) \cup (L' \setminus L(G)')$ jest niepusty. Niech zatem w będzie najkrótszym słowem w K. Rozważmy dwa przypadki:

JMA: Teraz pracowicie chowam królika do kapelusza... i potem w ułamku sekundy go wyciągnę i powiem "ach!"

1. $w \in L, w \notin L(G)$.
   - 1A. w ma nietrywialny niepusty prefiks v taki, że $|v|_0 = |v|_1$. Niech $w = vv'$.
     - 1Aa. $|v|_0$ jest parzysta. Wtedy $S \to_\Pi^* v$. Również $S \to_\Pi^* v'$. Zatem $S \to_\Pi^* SS \to_\Pi^* vv' = w$. Sprzeczność!
     - 1Ab. $|v|_0$ jest nieparzysta. Wtedy $T \to_\Pi^* v$. Również $T \to_\Pi^* v'$. Sprzeczność!
   - 1B. w nie ma takiego prefiksu.
     - 1Ba. $w = 0w'1$. Wtedy $T \to_\Pi^* w'$, ale $S \to_\Pi 0T1 \to_\Pi^* 0w^1 = w$. Sprzeczność!
     - 1Bb. Analogicznie.
2. $w \in L', w \notin L(G')$. Analogicznie.

5 Wykład 5 (13/03/2008)

5.1 Lemat o pompowaniu dla języków regularnych

Przypomnienie Dla każdego języka regularnego L istnieje taka stała $n \in \mathbb{N}$ (zwana "stałą z lematu o pompowaniu"), że dla każdego takiego słowa $w \in L$, że $|w| \geqslant n$, istnieje taki podział $w = stv$, że $|st| \leqslant n, |t| \geqslant 1$ i dla każdego $k \in \mathbb{N}$ słowo $st^k v \in L$.

5.2 Postać normalna Chomsky'ego gramatyki bezkontekstowej

Definicja Gramatyka $G = \langle \Sigma, N, S, \Pi \rangle$ jest w postaci normalnej Chomsky'ego (ChNF) jeśli każda produkcja z Π jest postaci $A \to a$, dla $A \in N, a \in \Sigma$ lub postaci $A \to BC$, dla pewnych $A, B, C \in N$.

Twierdzenie (nudne) Dla każdej CFG G istnieje taka CFG G' w ChNF, że $L(G) = L(G')$ (z dokładnością do słowa pustego).

Wyobraźmy sobie wyprowadzenie z gramatyki w ChNF.

(obrazek 5.1)

To drzewo bierze się z pewnej takiej gramatyki G w ChNF, że $|N| = n$. Niech d oznacza głębokość drzewa. Są dwie możliwości:

1. $d \leqslant n$, wtedy $|w| < 2^n$, co jest nudne.
2. $d > n$:

(obrazek 5.2)

Gdzieś na ścieżce od korzenia drzewa do liścia powtórzą się dwa nieterminale w odległości mniejszej od m. Wobec tego, możemy napompować sobie drzewo.

5.3 Lemat o pompowaniu dla języków bezkontekstowych

Twierdzenie Dla każdego języka bezkontekstowego L istnieje taka stała $m \in \mathbb{N}$ (zwana "stałą z lematu o pompowaniu" i równa $2^{|N|}$, gdzie N jest zbiorem nieterminali z takiej gramatyki G w ChNF, że $L = L(G)$), że dla każdego słowa $w \in L$, jeśli $|w| > m$, to istnieje taki podział $w = stvxy$, że $|tvx| \leqslant m, |tx| \geqslant 1$ i dla każdego $k \in \mathbb{N}$ słowo $st^k vx^k y \in L$.

Przykład Niech $L = \left\{w \in \{a,b,c\}^* : |w|_a = |w|_b = |w|_c\right\}$. Pokażemy, że L nie jest CFL.

Załóżmy, że L jest CFL. Niech n będzie stałą z lematu o pompowaniu. Rozważmy słowo $w = a^{2n}b^{2n}c^{2n}$. Niech $w = stvxy$ będzie takim podziałem słowa w, jak w lemacie o pompowaniu. W takim razie $|tvx| \leqslant n$. Zatem słowo tvx zawiera literki co najwyżej dwóch rodzajów. Słowo tx jest niepuste. Zatem:

- $|w|_a \neq |svy|_a$ lub
- $|w|_b \neq |svy|_b$ lub
- $|w|_c \neq |svy|_c$, ale
- $|w|_a = |svy|_a$ lub
- $|w|_b = |svy|_b$ lub
- $|w|_c = |svy|_c$.

Zatem nie jest prawdą, że $|svy|_a = |svy|_b = |svy|_c$, czyli $svy \notin L$. Sprzeczność!

Przykład Niech $L_1 = \left\{w \in \{a,bc\}^* : |w|_a = |w|_b\right\}$. Język ten jest CFL, bo wystarczy do przykładu pozbawionego c przekształcić produkcję $\cdots \to a$ na $\cdots \to A$ i $\cdots \to b$ na $\cdots \to B$, oraz dodać produkcje $A \to a \mid cA \mid Ac$ i $B \to b \mid cB \mid Bc$.

Niech $L_2 = \left\{w \in \{a,b,c\}^* : |w|_b = |w|_c\right\}$. A co to jest $L_1 \cap L_2$? Nie jest to CFL, jak pokazaliśmy w poprzednim przykładzie.

JMA: Widzimy to wszyscy? No ja nie wiem, czy wszyscy to widzą. Proszę to sobie wyobrazić w tramwaju, czy coś. Tylko żeby was nie okradli...

6 Wykład 6 (20/03/2008)

6.1 Automaty ze stosem

Definicja Niedeterministyczny automat ze stosem (NDPDA) to krotka $\langle \Sigma, N, Q, q_0, \ldots, \delta \rangle$, gdzie:

- Σ to alfabet taśmowy;
- N to alfabet stosowy, w tym specjalny symbol dna stosu $Z \in N$;
- Q to skończony zbiór stanów;
- stan początkowy $q_0 \in Q$;
- coś akceptującego;
- $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times N \times Q \times N^*$ to relacja przejścia.

Uwaga Ostrożnie z tym N^* — pilnujmy porządku w sprawie dna stosu. Mogą też się pojawić ε-przejścia.

Kiedy automat ze stosem akceptuje? Można się umawiać na 2 sposoby:

1. kiedy po zakończeniu wczytywania słowa może znaleźć się w pewnym stanie $q_f \in F$;
2. kiedy po zakończeniu wczytywania słowa ma pusty stos.

Okazuje się, że to na jedno wychodzi.

Języki rozpoznawane przez NDPDA to oczywiście języki bezkontekstowe.

Obserwacja Niech L_1 będzie bezkontekstowy, a L_2 regularny. Wtedy $L_1 \cap L_2$ jest bezkontekstowy.

Definicja Gramatyka bezkontekstowa $G = \langle \Sigma, N, S, \Pi \rangle$ jest w postaci Greibach jeśli każda produkcja z Π jest postaci $A \to aw$, gdzie $A \in N, a \in \Sigma, w \in N^*$.

Lemat Dla każdej gramatyki bezkontekstowej G istnieje taka gramatyka G' w postaci Greibach, że $L(G) = L(G')$ (z dokładnością do słowa pustego).

Twierdzenie Dla każdej gramatyki bezkontekstowej G istnieje NDPDA, który rozpoznaje dokładnie język $L(G)$.

Operacje wyprowadzania z nieterminali są przemienne, więc zaczniemy zawsze od najbardziej lewego — czyli wyprowadzone słowo rozpoczyna się pewną ilością terminali. Wyprowadzone już terminale to będzie słowo, które automat już wczytał, zaś niewyprowadzone nieterminale, to stos, który automat musi jeszcze pokonać. Automat zaakceptuje wtedy, kiedy będzie miał pusty stos.

JMA: Są dwie możliwości: albo włożymy dobrze, albo na odwrót.

Dowód Bez utraty ogólności możemy założyć, że G jest w postaci Greibach. Jeśli w Π jest produkcja $A \to aw$, gdzie $w \in N^*$, to w δ będzie krotka $\langle q, a, A, q, w \rangle$ (z dokładnością do kierunku wkładania w).

7 Wykład 7 (27/03/2008)

Uzupełnić!

8 Wykład 8 (3/04/2008)

Umówmy się, że język teraz będzie podzbiorem $\mathbb{N}$.

Weźmy sobie pewien ulubiony język programowania (MUJP). Interesują nas programy wczytujące argument będący liczbą naturalną.

Definicja (najważniejsza na tym kursie) Zbiór $A \subseteq \mathbb{N}$ jest rekurencyjny (obliczalny, rozstrzygalny) gdy istnieje taki program P w MUJP, że:

- $\forall_{n \in \mathbb{N}} (n \in A \Rightarrow P(n) = 1) \wedge (n \notin A \Rightarrow P(n) = 0)$.

Obserwacja Suma, przecięcie i dopełnienie zbiorów rekurencyjnych są rekurencyjne.

Eksperyment myślowy Wyobraźmy sobie, że wypiszemy wszystkie programy w MUJP, które zawsze zwracają wynik, w taki sposób, że wiemy, który program ma jaki numer. Rozważmy następujący program:

- wczytaj n;
- wylicz P_n;
- zapuść $P_n(n)$ i do wyniku dodaj 1;
- zwróć to, co wyszło.

Taki program nie znajduje się na liście tych programów.

Bardziej naturalnie jest myśleć o programach, które nie zawsze zwracają wynik.

Definicja Zbiór $A \subseteq \mathbb{N}$ jest rekurencyjnie przeliczalny (r.e.) jeśli taki istnieje P w MUJP, że:

- $\forall_{n\in\mathbb{N}} P(n) = 1 \Leftrightarrow n \in A$.

Obserwacja Przecięcie zbiorów r.e. jest r.e., w podobny sposób, jak wcześniej. Suma zbiorów r.e. również jest r.e. – ale należy być ostrożnym i uruchamiać obydwa programy na przemian na skończone odcinki czasu.

Obserwacja Jeśli $A \subseteq \mathbb{N}$ jest r.e. oraz $\overline{A}$ jest r.e., to A jest rekurencyjny. Inaczej, dopełnienie zbioru r.e. A nie jest r.e., chyba, że A jest rekurencyjny.

Umówmy się teraz, że potrafimy w jakiś sposób ponumerować wszystkie programy w MUJP. Potrafimy też określić, jakie wyniki zwracają dla jakich argumentów. Jeśli nie zwracają wyniku, to mówimy, że zwracają $\perp$.

Obserwacja Zbiór $K = \{n \in \mathbb{N} : \varphi_n(n) \in \mathbb{N}\}$ jest r.e. – wystarczy spojrzeć do tabelki.

Twierdzenie (Turinga o nierozstrzygalności problemu stopu) K nie jest rekurencyjny.

Dowód Załóżmy, że K jest rekurencyjny. Zatem istnieje taki program φ_m, że $\forall_{n\in\mathbb{N}}(n \in K \Rightarrow \varphi_m(n) = 1) \wedge (n \notin K \Rightarrow \varphi_m(n) = 0)$. Rozważmy następujący program:

- wczytaj n;
- wykonaj $\varphi_m(n)$:
  - jeśli otrzymałeś 0, to zwróć 1;
  - jeśli otrzymałeś 1, to się weź i zapętl.

Ten program ma jakiś numer; powiedzmy, że l. Czy $\varphi_l(l) \in \mathbb{N}$? Ten program zwróci 1 tylko, gdy $\varphi_m(l) = 0$, co jest równoważne temu, że $l \notin K$, więc $\varphi_l(l) \notin \mathbb{N}$. Sprzeczność!

Obserwacja $\overline{K}$ nie jest r.e. – w przeciwnym wypadku K byłby rekurencyjny.

9 Wykład 9 (8/04/2008)

9.1 Funkcje obliczalne (rekurencyjne)

Definicja Funkcja częściowa $f : \mathbb{N} \to \mathbb{N}$ nazywa się obliczalna albo rekurencyjna, gdy istnieje taki program φ w MUJP, że:

- $\forall_{n\in\mathbb{N}}(\varphi(n) = k \Rightarrow f(n) = k) \wedge (\varphi(n) = \perp \Rightarrow f(n))$ jest nieokreślona.

Definicja Funkcja całkowita rekurencyjna to funkcja $f : \mathbb{N} \to \mathbb{N}$, która jest rekurencyjna.

9.2 Redukcje obliczalne

Definicja Niech $A, B \subseteq \mathbb{N}$. Powiemy, że A jest nie trudniejszy niż B (w sensie redukcji obliczalnych), gdy istnieje taka funkcja $f : \mathbb{N} \to \mathbb{N}$, całkowita obliczalna, że:

- $\forall_n\, n \in A \Leftrightarrow f(n) \in B$.

Funkcja f nazywa się redukcją, zaś powyższe zdanie zapisujemy jako $A \leqslant_{\text{rek}} B$.

Obserwacja Jeśli $A \leqslant_{\text{rek}} B$ i B jest rekurencyjny, to A też jest rekurencyjny. Podobnie, jeśli $A \leqslant_{\text{rek}} B$ jest r.e. to A też jest r.e.

Jak korzystając z powyższej obserwacji udowodnić, że A nie jest rekurencyjny? Wystarczy pokazać, że $K \leqslant_{\text{rek}} A$.

JMA: Niepotrzebnie powiedziałem, co myślę. Stalin nigdy tak nie robił.

Przykład Niech $A = \{n : \varphi_n(7) = 13\}$. Pokażemy, że A nie jest rekurencyjny, czyli że $K \leqslant_{\text{rek}} A$. Szukamy takiej f, że $\forall_n\, n \in K \Leftrightarrow f(n) \in A$, czyli że $\varphi_n(n) \in \mathbb{N} \Leftrightarrow \varphi_{f(n)}(7) = 13$:

- wczytaj n;
- napisz sobie taki program, ale broń Boże go nie uruchamiaj:
  - wczytaj m;
  - oblicz $\varphi_n(n)$;
  - zwróć 13;
- znajdź numer k tego programu i podstaw $f(n) = k$.

Kiedy $f(n) \in A$?

- $n \in K \Rightarrow \varphi_{f(n)}(7) = 13 \Rightarrow f(n) \in A$;
- $n \notin K \Rightarrow \varphi_{f(n)}(7) \neq 13 \Rightarrow f(n) \notin A$.

Definicja Niech $A \subseteq \mathbb{N}$. Powiemy, że:

1. A jest nietrywialny, jeśli $A \neq \varnothing$ i $A \neq \mathbb{N}$;
2. A jest ekstensjonalny, jeśli dla każdych dwóch liczb naturalnych m, k, jeśli φ_m i φ_k obliczają te same funkcje (częściowe), to $k \in A \Leftrightarrow m \in A$.

Twierdzenie (straszne, Rice'a) Żaden nietrywialny, ekstensjonalny podzbiór $\mathbb{N}$ nie jest rekurencyjny.

JMA: Pan sobie jakieś sudoku rozwiązuje? Przecież nie ma sensu tego pisać; i tak Mietek wszystko napisze...

Dowód Niech $A \subseteq \mathbb{N}$ będzie nietrywialny i ekstensjonalny. Bez utraty ogólności możemy założyć, że żaden numer funkcji pustej nie należy do A. Pokażemy, że $K \leqslant_{\text{rek}} A$. Załóżmy, że $c \in A$. Skonstruujemy redukcję f:

- wczytaj n;
- napisz sobie taki program, a.b.B.g.n.u.:
  - wczytaj m;
  - uruchom $\varphi_n(n)$;
  - zwróć $\varphi_c(m)$;
- znajdź numer k tego pięknego programu i podstaw $f(n) = k$.

Co robi ten program?

- $n \in K \Rightarrow$ wczytuje m i zwraca $\varphi_c(m)$, czyli wylicza φ_c;
- $n \notin K \Rightarrow$ wczytuje m i oddala się w celu zapętlenia, czyli wylicza funkcję pustą.

10 Wykład 10 (10/04/2008)

Uzupełnić!

11 Wykład 11 (15/04/2008)

Poprzednio zaczęliśmy pokazywać, że SemiThue $\geqslant_{\text{rek}} K$. Zaczęliśmy pisać następującą redukcję: dajcie nam n, a zrobimy z niego taką krotkę $\langle \Pi_n, w_n, v_n \rangle$, że $\forall_{n\in\mathbb{N}}\, n \in K \Leftrightarrow \langle \Pi_n, w_n, v_n \rangle \in$ SemiThue.

JMA: Ja naprawdę bym umiał pięknie to sformalizować. Gdzieś w 60. minucie zacząłbym się tym nawet cieszyć.

Mieliśmy alfabet $A_n = Q \cup \{\alpha, \omega, 1, 0, B, \overline{B}, Z_L, Z_R\}$, gdzie Q to wszystkie stany q_n, B to blank, $\overline{B}$ to nieskończenie wiele blanków, zaś Z_L i Z_R to zjadacze w lewo i prawo.

(powtórzony rysunek)

Kod konfiguracji początkowej $\varphi_n(n)$ to słowo $q_0\alpha(\text{binarnie } n)\omega B\overline{B} = w_n$.

Jeśli $q_i, a \to q_j, b, R$ jest instrukcją φ_n, to produkcja $\langle q_i a, bq_j \rangle$ należy do Π_n.

Jeśli $q_i, a \to q_j, b, L$ jest instrukcją φ_n, to w Π_n dla każdego $c \in \{0, 1, \alpha, \omega\}$ znajdzie się produkcja $cq_ia \to q_jcb$.

Dla każdego $q \in Q$ w Π_n jest produkcja $qB\overline{B} \to qBB\overline{B}$.

Wiemy, jakie jest w_n i z grubsza wiemy, jakie jest Π_n. Ale co z v_n?

Dla każdej takiej pary q, a, że w zbiorze instrukcji φ_n nie ma instrukcji zaczynającej się od $q, a \to \ldots$, umieszczamy w Π_n produkcję $q, a \to Z_LZ_R$. Dodatkowo, dla każdego $a \in A$ w Π_n są produkcje $aZ_L \to Z_L$ i $Z_Ra \to Z_R$.

W końcu, przyjmujemy $v_n = Z_LZ_R$.

JMA: Tak mi się skojarzyły te dzwonki... A to Karol mieszał herbatę.

KST: Przepraszam za zamieszanie.

11.1 Symetryczna piszczałka

Przypomnienie Grupa to zbiór G z działaniem $\cdot$, które jest łączne i ma taki element neutralny e, że $\forall_{x\in G}\, x \cdot e = e \cdot x = x$, oraz $\forall_x \exists_y\, x \cdot y = e$. Półgrupa nie ma tego ostatniego warunku.

Grupa wolna F_3, o trzech generatorach, to $a, b, c, a^{-1}, b^{-1}, c^{-1}$, gdzie $\cdot$ jest konkatenacją słów. Przykładowo, $aa^{-1}b = b$. Półgrupa wolna nie ma elementów odwrotnych.

Weźmy grupę wolną F_2 i podzielmy ją przez relację równoważności $ab = ba$. Będzie to $\mathbb{Z} \times \mathbb{Z}$.

Jeśli podzielimy F_2 przez $ab = ba, aaa = \varepsilon, bbb = \varepsilon$, to dostaniemy $\mathbb{Z}_3 \times \mathbb{Z}_3$.

Problem (słów w półgrupie) Dają nam półgrupę (skończony zbiór E równości słów) i słowa w, v. Pytają nas, czy $w =_E v$. To przecież problem słów dla systemów Thuego:

- Thue $= \{\langle \Pi, w, v \rangle : w \leftrightarrow^*_\Pi v\}$, gdzie $\leftrightarrow$ to najmniejsza relacja równoważności zawierająca $\to^*_\Pi$.

Twierdzenie Problem słów dla systemów Thuego jest nierozstrzygalny.

Dowód (ten sam, co poprzednio) Definiujemy w_n, v_n, Π_n tak, jak poprzednio, no i mamy pokazać, że $\forall_n\, n \in K \Leftrightarrow w_n \leftrightarrow^*_\Pi v_n$. Wiemy, że $n \in K \Rightarrow w_n \to^*_\Pi v_n$, czyli tym bardziej $w_n \leftrightarrow^*_\Pi v_n$.

Potrzebny jest lemat, który jest mówi, że jeśli jesteśmy w stanie z pewnego miejsca w dorzeczu dopłynąć w dowolny sposób do ujścia, to jesteśmy też w stanie dopłynąć tam wyłącznie z prądem...

12 Wykład 12 (15/04/2008)

12.1 Problem odpowiedniości Posta (PCP)

JMA: Proszę nie łaskotać Agaty na wykładzie. Zachowujecie się jak dzieci w przedszkolu, tak... W przedszkolu wieczorowym.

Dają nam $\Pi = \{\langle w_1, v_1\rangle, \langle w_2, v_2\rangle, \ldots, \langle w_l, v_l\rangle\}$ — skończony zbiór uporządkowanych par słów nad jakimś alfabetem.

Pytają, czy istnieje takie niepuste słowo $s = s_1 s_2 \ldots s_k; s_i \in \{1, 2, \ldots, l\}$ (zwane rozwiązaniem danej instancji PCP), że $w_{s_1} w_{s_2} \ldots w_{s_k} = v_{s_1} v_{s_2} \ldots v_{s_k}$.

Twierdzenie $\mathrm{PCP} = \{\Pi = \{\langle w_1, v_1\rangle, \ldots, \langle w_l, v_l\rangle\} : \exists_{s=s_1 \ldots s_k \neq \varnothing}\, w_{s_1} \ldots w_{s_k} = v_{s_1} \ldots v_{s_k}\}$ jest nierozstrzygalny.

Dowód (brak rysunku do pierwszej przymiarki do dowodu)

Budujemy przykład $\mathrm{PCP}P(\Pi, w, v)$, który składa się z:

- $\langle \#, \#u_0\# \rangle$;
- $\langle \#u\#, \# \rangle$;
- $\langle a, a \rangle$, takie że a należy do alfabetu Π;
- Π;
- $\langle \#, \# \rangle$.

Ale taki przykład zawsze ma rozwiązanie, ze względu na obecność par $\langle a, a\rangle$.

JMA: Dwie kreski anihilują. Powstaje przy tym energia! Kwant energii kreskowej.

Mamy dane u_0, u, Π. Niech Σ oznacza alfabet Π. Weźmy $\overline{\Sigma} = \{\overline{a} : a \in \Sigma\}$, gdzie kreskę definiujemy następująco:

- $\overline{\overline{a}} = a$;
- $\varepsilon = \overline{\varepsilon}$;
- $w \in (\Sigma \cup \overline{\Sigma})^*, a \in \Sigma \cup \overline{\Sigma} \Rightarrow \overline{aw} = \overline{aw}$.

Budujemy przykład $\mathrm{PCP}P(\Pi, u_o, u)$, którego alfabetem będzie $\Sigma \cup \overline{\Sigma} \cup \{\#, \overline{\#}\}$.

(brak poprawionego rysunku z pierwszej przymiarki do dowodu)

Teraz $P(\Pi, u_0, u)$ będzie się składać z:

- $\langle \overline{\#}, \overline{\#u_0}\# \rangle$;
- $\langle \#u\overline{\#}, \overline{\#} \rangle$;
- $\langle a, \overline{a} \rangle$, gdzie $a \in \Sigma \cup \overline{\sum}$;
- $\langle \#, \overline{\#} \rangle$;
- $\langle \overline{\#}, \# \rangle$;
- $\{\langle s, \overline{t}\rangle, \langle \overline{s}, t\rangle : \langle s, t\rangle \in \Pi\}$.

Wniosek Korzystając z nierozstrzygalności PCPpokażemy, że problem niepustości przekroju dwóch danych języków bezkontekstowych (PCFL) jest nierozstrzygalny.

$\mathrm{PCFL} = \{\langle G, H\rangle \in \mathrm{CFG}^2 : L(G) \cap L(H) \neq \varnothing\}$

Pokażemy, że $\mathrm{PCP} \leqslant_{\mathrm{rek}} \mathrm{PCFL}$, czyli skonstruujemy algorytm (funkcję obliczalną), który jako swoje wejście wczyta Π, a następnie zwróci dwie taki gramatyki G_Π, H_Π, aby zachodziła równoważność $\Pi \in \mathrm{PCP} \Leftrightarrow \langle G_\Pi, H_\Pi\rangle \in \mathrm{PCFL}$.

Weźmy $\Pi = \{\langle w_1, v_1\rangle, \ldots, \langle w_l, v_l\rangle\}$. Skonstruujmy gramatyki G_Π, H_Π nad sumą alfabetu Π i $\{i_1, \ldots, i_l\}$:

- $G_\Pi = S_G \to \varepsilon \mid w_j S_G i_j$, dla $j = 1, \ldots, l$;
- $H_\Pi = S_H \to \varepsilon \mid v_j S_H i_j$, dla $j = 1, \ldots, l$.

13 Wykład 13 (22/04/2008)

13.1 Nierozstrzygalność logiki pierwszego rzędu (rachunku predykatów)

Składnia służąca do pisania zdań:

- pewna ilość stałych: $c, k, \ldots$;
- pewna ilość zmiennych: $x, y, z, t, \ldots$;
- pewna ilość symboli relacyjnych: $E, \ldots$.

Formuła atomowa: $E(x, y), E(c, z), \ldots$

Dla formuły φ napis $FV(\varphi)$ oznacza wszystkie występujące w φ zmienne wolne (free variables).

Dla formuł φ, ψ napisy $\neg\varphi; \varphi \vee \psi; \varphi \wedge \psi$ to również formuły; zaś $FV(\neg\varphi) = FV(\varphi); FV(\varphi \vee \psi) = FV(\varphi \wedge \psi) = FV(\varphi) \cup FV(\psi)$.

Jeśli $x \in FV(\varphi)$, to $\exists_x \varphi; \forall_x$ są formułami; zaś $FV(\exists_x \varphi) = FV(\forall_x \varphi) = FV(\varphi) \setminus \{x\}$.

Zdanie to formuła bez zmiennych wolnych.

Wolno nam też używać równości.

Wyobraźmy sobie, że mamy jakiś (skończony) świat. Co to jest świat? Otóż świat to pewien zbiór punktów... oraz odpowiednio zdefiniowane relacje — czyli interpretacja symboli z sygnaturą.

Napis $m \models \varphi$ oznacza, że formuła φ jest spełniona w modelu m.

Obserwacja Sprawdzanie, czy dana formuła jest prawdziwa w danym modelu (model checking) jest rozstrzygalne w czasie wielomianowym; każdy kwantyfikator kosztuje nas liniowo.

Obserwacja Dają nam φ i pytają, czy istnieje takie m (skończony lub nie), że $m \models \varphi$ (skończona spełnialność lub spełnialność). Oba te pytania są dla logiki pierwszego rzędu (first order logic) nierozstrzygalne. Skończona spełnialność jest rekurencyjnie przeliczalna, bo możemy przeglądać wszystkie skończone modele. Okazuje się, że niespełnialność jest również rekurencyjnie przeliczalna, więc spełnialność nie jest.

Przykład Formuła spełnialna, ale nie skończenie spełnialna:

- $\forall_x \exists_y E(x, y) \wedge \exists_x \forall_y E(y, x) \wedge \forall_{x,y,z}(E(x, z) \wedge E(y, z)) \Rightarrow x = y$

Modelem dla tej formuły jest $\mathbb{N}$.

Twierdzenie Problem spełnialności jest nierozstrzygalny.

Dowód Dla danej liczby naturalnej n zbudujemy taką formułę ψ_n, że $n \notin K$ (φ_n(n się nie zatrzymuje) wtedy i tylko wtedy, gdy ψ_n jest spełnialna.

JMA: Gdyby *ta* pani to tłumaczyła, to obie nazywały by się $P\ldots$

Weźmy dwie relacje binarne H, V (pozioma i pionowa). W ψ_n będzie napisane, że H, V mają takie własności, że:

- każdy wierzchołek ma dokładnie jednego następnika ($\forall_x \exists_y H(x, y) \wedge \forall_{x,y,z}(H(x, y) \wedge H(x, z)) \Rightarrow y = z$; podobnie dla V);
- każdy wierzchołek ma co najwyżej jednego poprzednika;
- $\forall_{x,y,z,t,r} H(x, y) \wedge V(x, z) \wedge V(y, t) \wedge H(z, r) \Rightarrow t = r$.

W ψ_n będą się też pojawiać symbole unarne $J, Z, B, A, \Omega, \{Q_q : q \in Q_n\}$.

Na każdym wierszu siatki chcemy, że była zapisana jedna konfiguracja maszyny Turinga. Następny wiersz istnieje tylko, jeśli istnieje następna konfiguracja.

Umiemy powiedzieć, że $\exists_x Q_{q_0}(x) \wedge A(x)$.

W każdym z wierzchołków siatki zachodzi dokładnie jeden z predykatów Q_q. Oprócz tego gdzieś jest jeszcze głowica.

Poza tym:

- $\forall_{x,y}\, B(x) \wedge H(x,y) \Rightarrow B(y)$
- $\forall_{x,y}\, \neg(H(x,y) \wedge A(y))$;
- $\forall_{x,y}\, V(x,y) \wedge A(x) \Rightarrow A(y)$.

Kopiowanie taśmy odbywa się porządnie:

- $\forall_{x,y}\, J(x) \wedge \neg Q_{q_1}(x) \wedge \neg Q_{q_2}(x) \wedge \cdots \wedge \neg Q_{q_l}(x) \wedge V(x,y) \Rightarrow I(y)$.

Jeśli wśród instrukcji φ_n jest $\langle q, 1, q', 0, R\rangle$, to w ψ_n znajdzie się koniunkt:

- $\forall_{x,y,z}\, J(x) \wedge Q_q(x) \wedge V(x,y) \wedge H(y,z) \Rightarrow Z(y) \wedge Q_{q'}(z)$.

Ciąg dalszy na następnym wykładzie.

14 Wykład 14 (29/04/2008)

14.1 Nierozstrzygalność problemu spełnialności dla FOL (ciąg dalszy)

JMA: Artur to widzi? Tomek to widzi? To jak już oni dwaj widzą, to pole morfogenetyczne się powinno wytworzyć...

Dowód (nowy) Pokażemy, że $\overline{\textsc{Thue}} \leqslant_{\text{rek}} \textsc{Spełnialność FOL}$.

Dają nam $\Sigma = \{1, \ldots, k\}$, skończony $\Pi \subseteq \Sigma^* \times \Sigma^*$.

Czy $1 \leftrightarrow^*_\Pi 2$?

A jeśli mamy w, v i produkcje $w \leftrightarrow 1, v \leftrightarrow 2 \in \Pi$?

Weźmy sygnaturę $\{c, E_1, E_2, \ldots, E_k\} \cup \{R_v : v \in L\}$, gdzie L oznacza zbiór wszystkich prefiksów słow występujących w jakiejś parze w Π. c będzie udawać słowo puste.

Chcemy doprowadzić do tego, że ψ_Π jest spełnialna wtedy i tylko wtedy, gdy $1 \leftrightarrow^*_\Pi 2$.

Zbudujmy formułę pierwszego rzędu, koniunkcję ψ_Π:

- zasadźmy drzewo o korzeniu w c:
  - $\forall_x \exists_y E_1(x,y)$;
  - $\forall_x \exists_y E_2(x,y)$;
  - ...
  - $\forall_x \exists_y E_k(x,y)$;
- $\forall_x R_\varepsilon(x,x)$;
- jeśli $w, wa \in L$, to w formule jest koniunkt:
  - $\forall_{x,y,z}\, R_w(x,y) \wedge E_a(y,z) \Rightarrow R_{wa}(x,z)$;
- dla każdej pary $\langle w, v\rangle \in \Pi$ w ψ_Π jest koniunkt:
  - $\forall_{x,y,z}\, R_w(x,y) \wedge R_v(x,z) \Rightarrow y = z$;
- być może potrzebne będzie również dla każdego $i = 1, \ldots, k$:
  - $\forall_{x,y,z}\, E_i(x,y) \wedge E_i(x,z) \Rightarrow y = z$;
- ostatecznie:
  - $\forall_{x,y}\, E_1(c,x) \wedge E_2(c,y) \Rightarrow x \neq y$.

Pokażmy, że:

1. $1 \not\leftrightarrow^*_\Pi 2 \Rightarrow \psi_\Pi$ jest spełnialna;
2. $1 \leftrightarrow^*_\Pi 2 \Rightarrow \psi_\Pi$ jest niespełnialna.

Pokazaliśmy.

JMA: To się nazywa twierdzenie Gödla i są takie dziewczęta z filozofii, które nie są postmodernistkami i które byłyby bardzo zainteresowane tym, że byliście na takim wykładzie.

14.2 O nierozstrzygalności arytmetyki

Mamy $\mathbb{N}$, prawdziwe liczby naturalne, oraz działania, które lubią ze sobą robić, czyli np. $\cdot, +, \leqslant, =$. Mamy też zmienne, stałe, kwantyfikatory i operatory boolowskie. Możemy więc pisać formuły, np.:

- $\exists_x \forall_y (y > x \Rightarrow \exists_{z,t} z \neq 1 \wedge t \neq 1 \wedge (z \cdot t = y \vee z \cdot t = y + 2))$, czyli, że liczb pierwszych bliźniaczych jest skończenie wiele.

Czy $\langle \mathbb{N}, \cdot, + \rangle \models \varphi$?

Twierdzenie $\{\varphi : \langle \mathbb{N}, \cdot, + \rangle \models \varphi\}$ jest nierozstrzygalny.

Ciąg dalszy za jakiś czas.

JMA: Mieliśmy też taki obrazek...

KST: W tym roku ten obrazek pojawia się pierwszy raz.

JMA: No być może... Ale tego wykładu nie zalicza się w ciągu jednego roku.

15 Wykład 15 (6/05/2008)

15.1 Teoria złożoności obliczeniowej

Mieliśmy uprzednio relację $\leqslant_{\text{rek}}$, dzięki której pokazaliśmy dużo fajnych rzeczy.

15.1.1 Klasa P (PTime)

Problem $X \subseteq \{0,1\}^*$ jest rozstrzygalny w czasie wielomianowym ($X \in \text{PTime}$), jeśli istnieją takie maszyna Turinga M i wielomian p, że:

1. $\forall_{x \in \{0,1\}^*} (M(x) = 1 \Leftrightarrow x \in X) \wedge (M(x) = 0 \Leftrightarrow x \notin X)$;
2. $\forall_{x \in \{0,1\}^*} M(x)$ zatrzyma się po nie więcej niż $p(|x|)$.

(*mażąc tablicę*)

JMA: Zobaczcie jak *on* przyciska tę kredę...

Przykład Dany graf z wagami krawędzi, dwa wierzchołki s, t, liczba $k \in \mathbb{N}$. Czy istnieje ścieżka o wadze $\leqslant k$ z s do t? Wiadomo z olimpiady informatycznej dla gimnazjalistów wieczorowych, że jest to w PTime.

Przykład Dana liczba naturalna n. Czy n jest pierwsza? Okazuje się, że jest takie twierdzenie z roku 2002, że ten problem też jest w PTime.

JMA: Napiszemy to tak, jak by to napisał TWI... Bo on bardzo lubi składać różne rzeczy.

15.1.2 Redukcje wielomianowe

$A \leqslant_p B$, czyli A jest nie trudniejszy niż B w sensie redukcji wielomianowych, gdy istnieją takie maszyna Turinga M i wielomian q, że:

1. $\forall_{x \in \{0,1\}^*} M(x)$ zatrzyma się po nie więcej niż $q(|x|)$ krokach;
2. $\forall_{x \in \{0,1\}^*} x \in A \Leftrightarrow M(x) \in B$.

Obserwacja Jeśli $B \in \text{PTime}$, $A \leqslant_p B$, to $A \in P$. Dlaczego?

Niech M oznacza redukcję dla $A \leqslant_p B$, działającą w czasie q; zaś M_B oznacza maszynę rozpoznającą B, działającą w czasie p. Złożenie maszyn działa w czasie wielomianowym; stopień tego wielomianu jest nie większy niż $p + q$.

JMA: W każdym wierzchołku grafu chcemy postawić ludzika o jednej z trzech możliwych płci... Chciałem powiedzieć rasie, ale to byłoby jeszcze bardziej niepoprawne.

Przykład (3Col) Nie wiemy, czy $3\text{Col} \in \text{PTime}$.

Przypomnienie Literał to zmienna logiczna lub jej negacja, zaś k-klauzula to alternatywa $\leqslant k$ literałów. Formuła jest postaci k-CNF, gdy jest koniunkcją k-klauzul.

JMA: Algorytmu wielomianowego nie widać na pierwszy rzut oka, prawda? I gdyby ktoś znał taki algorytm, to niech nikomu nie mówi... Tylko mnie.

Przykład (3Sat) Dana formuła zdaniowa φ w postaci 3CNF. Czy φ jest spełnialna? Nie wiemy, czy $3\text{Sat} \in \text{PTime}$.

Twierdzenie $3\text{Col} \leqslant_p 3\text{Sat}$.

Dowód Pokażemy wielomianowy algorytm, który wczyta graf nieskierowany G, a zwraca taką formułę $\varphi_G \in 3\text{CNF}$, że $\varphi_G \in 3\text{Sat} \Leftrightarrow G \in 3\text{Col}$.

JMA: Formuły się nie lubią nazywać x! To tak, jakby ktoś w Urugwaju się miał nazywać Zenek.

Niech $G = \langle V, E\rangle$. Zbiorem zmiennych formuły φ_G będzie $\{p_v, q_v, r_v : v \in V\}$. Dla każdego $v \in V$ w φ_G będzie klauzula:

- $p_v \vee q_v \vee r_v$.

Dla każdych takich $v, w \in V$, że $\langle v, w\rangle \in E$ będą w φ_G klauzule:

- $\neg p_w \vee \neg p_v$;
- $\neg q_w \vee \neg q_v$;
- $\neg r_w \vee \neg r_v$.

Redukcja została oczywiście zbudowana w czasie wielomianowym. Trzeba teraz pokazać dwie implikacje.

Pokażemy, że formuła, którą napisaliśmy, jest spełnialna. Na podstawie kolorowania grafu budujemy wartościowanie formuły. Aby się przekonać, że wartościowanie to spełnia formułę, wystarczy się przekonać, że spełnia każdą klauzulę z osobna.

JMA: Za dobrego rektora były przerwy!

Pozostaje jeszcze implikacja w drugą stronę.

Przykład $3\text{Sat} \leqslant_p 3\text{Col}$.

Dowód Teraz wielomianowa redukcja jako argument weźmie formułę w postaci 3CNFi zbuduje graf G_φ w taki sposób, by $\varphi \in 3\text{Sat} \Leftrightarrow G_\varphi \in 3\text{Col}$.

JMA: Jakby to powiedzieli buddyści, te problemy są palcami jednej ręki... Lub czymś takim. Ta ręka ma bardzo dużo palców.

Niech zmiennymi w φ będą (bez utraty ogólności) $p_1, \ldots, p_n$.

(obrazek 15.1)

Zbudujmy po jednym gadżecie dla każdej klauzuli z φ:

(obrazek 15.2)

Kiedy funkcja $k : \{w_1, w_2, w_3\} \to \{P, F, N\}$ daje się przedłużyć do poprawnego 3-kolorowania gadżetu?

Jeśli k jest stała, to nie da się przedłużyć. Jeśli k nie jest stała, to się da — zachodzą dwa rodzaje przypadków.

(obrazek 15.3)

16 Wykład 16 (13/05/2008)

16.1 Klasa NP

NPto klasa problemów rozstrzygalnych w czasie wielomianowym przez algorytm niedeterministyczny.

Deterministyczna maszyna Turinga:

- funkcja częściowa $\Sigma \times Q \to \Sigma \times Q \times \{R, L\}$.

Niedeterministyczna maszyna Turinga:

- relacja zawarta w $(\Sigma \times Q) \times (\Sigma \times Q \times \{R, L\})$.

NDTM akceptuje wtedy, kiedy z początkowej konfiguracji istnieje ścieżka do pewnej konfiguracji akceptującej.

Wszystkie ścieżki zaczynające się w początkowej konfiguracji kończą się szybciej, niż w liczbie kroków zależnej wielomianowo od wielkości danych.

Możemy się też umówić tak, że ścieżka prowadząca do dowolnej konfiguracji akceptującej musi mieć podobnie ograniczoną długość — to tak naprawdę to samo, bo możemy tej maszynie zamontować budzik ograniczający długość nieakceptujących ścieżek.

Twierdzenie Wiemy, że $P \subseteq NP$. Nie wiemy, czy $NP \subseteq P$.

Przykłady SAT $\in$ NP. Głos z kratki daje nam wartościowanie, a my je sprawdzamy w czasie wielomianowym.

3COL $\in$ NP. Podobnie, głos z kratki daje nam kolorowanie...

Izomorfizm grafów (ISO) $\in$ NP. Głos z kratki dyktuje nam, które wierzchołki mamy przenumerować na które, a my porównujemy graf źródłowy z wynikowym.

Tautologie (TAUT) $\notin$ NP (o ile nam wiadomo). Za to TAUT $\in$ co-NP, bo $\overline{\text{TAUT}} \in$ NP.

16.2 Zagadka

Czy są jacyś kandydaci na bycie w $(\text{NP} \cap \text{co-NP}) \setminus \text{P}$?

A co to właściwie jest co-NP?

- co-$S = \left\{A \subseteq \{0,1\}^* : \overline{A} \in S\right\}$.

Wyobraźmy sobie grę, w której dany jest graf skierowany $G = \langle V, E\rangle$; funkcja $V \to N$; funkcja $V \to \{\text{Ja}, \text{Ty}\}$; początek $v_0 \in V$. Ja wygrywam, jeśli największa liczba napotkana po drodze nieskończenie wiele razy jest parzysta — a Ty, jak nieparzysta.

Okazuje się, że dowolna strategia wygrywająca dla tej gry jest pozycyjną strategią wygrywającą; tzn. że dla danych wierzchołków są z góry ustalone ruchy. Toteż głos z kratki może nam dać taką strategię, a my możemy ją sprawdzić.

Obserwacja $A \leqslant_p B, B \in \text{NP} \Rightarrow A \in \text{NP}$. Najpierw wielomianowo, deterministycznye tłumaczymy redukcją x na $f(x)$, po czym niedeterministycznie sprawdzamy, czy $f(x) \in B$.

Twierdzenie Jeśli P $\neq$ NP, to w NPistnieje nieskończenie wiele klas równoważności ze względu na $\leqslant_p \cap \geqslant_p$.

Opinia Okazuje się, że jeśli $(\text{NP} \cap \text{co-NP}) \setminus \text{P}$ jest pusta, to nie istnieje kryptografia jednostronna (asymetryczna).

16.3 NP-zupełność

Jeśli A jest NP-zupełny, to:

1. $A \in$ NP;
2. $\forall_{B \in \text{NP}}\, B \leqslant_p A$.

Twierdzenie (Cooka) 3SAT jest NP-zupełny.

Problem jest NP-zupełny, gdy jest NP-trudny i należy do NP, zaś problem jest NP-trudny, gdy $\forall_{B \in \text{NP}}\, B \leqslant_p A$.

16.4 LogSpace

Maszyna teraz ma dwie niezależne taśmy — jedną tylko do odczytu, z danymi; oraz drugą, o długości zależnej logarytmicznie od długości wejścia.

JMA: Można sobie wyobrazić bardziej wyrafinowane metody bycia pająkiem, oczywiście...

Jeśli mamy jakiś graf na taśmie danych, to na taśmie roboczej potrafimy zapamiętać c wierzchołków, gdzie c jest niezależne od wielkości danych. Jesteśmy takim pająkiem, który ma na każdej stopie oko; stoi sobie w c wierzchołkach; widzi, gdzie stoi i dokąd może pójść — ale nie pamięta, gdzie wcześniej był.

Dany jest graf nieskierowany i wierzchołki s, t. Czy istnieje droga z s do t? Łatwo pokazać, że to się da rozstrzygać w przestrzeni $\log^2(|v|)$. A jak pokazać, że to nie jest w LOGSPACE? Jeśli mamy skończenie wiele nóg, a dają nam bardzo duży graf, to jak mamy sprawdzić, że z jakiegoś s do jakiegoś t można dojść? One niby mogą być bardzo daleko od siebie...

Pewnym zaskoczeniem okazało się w 2004 roku Rheingold pokazał, że osiągalność w grafie nieskierowanym jest w LOGSPACE.

17 Wykład 17 (20/05/2008)

17.1 Twierdzenie Cooka

3SAT jest NP-zupełny.

Lemat 6SAT jest NP-trudny.

Dowód lematu Weźmy dowolny problem $B \in \mathrm{NP}$. Pokażemy, że $B \leqslant_p$ 6SAT.

JMA: Czy pan, który jest ponury, rozumie? Taki był pan ponury, że aż się przeraziłem, że mogłem powiedzieć jakiś dowcip.

Ale jedyne, co wiemy o B, to to, że istnieje taki wielomian p i taka NDTM M_B, że $M_B(x)$ akceptuje wtedy, gdy $x \in B$, zaś zatrzymuje się po $p(|x|)$ ruchach. Odtąd będziemy myśleć, że M_B i p są ustalone.

Musimy skonstruować redukcję, to znaczy funkcję, która weźmie x i zwróci taką formułę $\varphi_x \in$ 6CNF, że $M_B(x)$ akceptuje, gdy φ_x jest spełnialna.

(obrazek 17.1)

JMA: Tak to dzisiaj przy goleniu wymyśliłem... Dlatego nigdy nie ufajcie nieogolonym wykładowcom!

Dla każdego stanu maszyny ze zbioru $Q_{M_B} = \{q_0, \ldots, q_l\}$ i dla każdego wiersza tabelki i tworzymy zmienne $q_0^i, \ldots, q_l^i$. W każdej komórce tabelki znajdują się zmienne $\alpha_{i,j}, \omega_{i,j}, 0_{i,j}, 1_{i,j}, B_{i,j}, q_{i,j}^{r,l}, q_{i,j}^{r,r}, q_{i,j}^{l,r}, q_{i,j}^{l,l}$. Oczywiście $i, j \in \{1, \ldots, p(|x|)\}$.

Chcemy, żeby w każdej komórce było dokładnie jedno z $1, 0, \alpha, \omega, B$. Dlatego dla każdego $i, j \in \{1, \ldots, p(|x|)\}$ w φ_x znajdą się klauzule:

- $(1_{i,j} \vee 0_{i,j} \vee \alpha_{i,j} \vee \omega_{i,j} \vee B_{i,j})$;
- $(\neg 1_{i,j} \vee \neg 0_{i,j})$;
- $(\neg 0_{i,j} \vee \neg \alpha_{i,j})$;
- siedem kolejnych;
- $(\neg \omega_{i,j} \vee \neg B_{i,j})$.

Zadbamy teraz, żeby w pierwszej linii głowica była tam, gdzie powinna:

- $(q_{1,1}^{l,r})$;
- $(\neg q_{1,1}^{l,l})$;
- $(\neg q_{1,1}^{r,l})$;
- $(\neg q_{1,1}^{r,r})$;
- dla każdego $j \in \{2, \ldots, p(|x|)\}$ mamy klauzule:
  - $(\neg q_{1,j}^{l,l})$;
  - $(\neg q_{1,j}^{l,r})$;
  - $(\neg q_{1,j}^{r,r})$;
  - $(\neg q_{1,j}^{r,l})$.

JMA: Na razie spójrzmy w przeszłość; później odetniemy ją grubą kreską i spojrzymy w przyszłość.

Zadbajmy teraz o to, by w każdym miejscu świeciła się co najwyżej jedna zmienna q:

- dla każdego $i \in \{2, \ldots, p(|x|)\}, j \in \{1, \ldots, p(|x|)\}$:
  - $(\neg q_{i,j}^{l,l} \vee \neg q_{i,j}^{l,r})$;
  - cztery kolejne;
  - $(\neg q_{i,j}^{r,l} \vee \neg q_{i,j}^{r,r})$.

Niech to wygląda jak ruch głowicy:

- dla każdego i, j o odpowiednich wartościach potrafimy powiedzieć, skąd przyszliśmy:
  - $(q_{i,j}^{l,*} \Rightarrow (q_{i-1,j-1}^{l,r} \vee q_{i-1,j-1}^{r,r}))$;
  - $(q_{i,j}^{r,*} \Rightarrow (q_{i-1,j+1}^{l,l} \vee q_{i-1,j+1}^{r,l}))$;
  - dwie analogiczne, wynikające z rozwinięcia gwiazdki;
- oraz dokąd idziemy:
  - $(q_{i,j}^{*,r} \Rightarrow (q_{i+1,j+1}^{l,l} \vee q_{i+1,j+1}^{l,r}))$;
  - $(q_{i,j}^{*,l} \Rightarrow (q_{i+1,j-1}^{r,l} \vee q_{i+1,j-1}^{r,r}))$;
  - dwie analogiczne.

JMA: Czy pan się już połapał? Nie? No to chyba trzeba będzie pana dobić...

Nic się na taśmie nie zmienia bez obecności głowicy:

- dla każdego i, j napiszemy 5 klauzul:
  - $((1_{i,j} \wedge \neg q_{i,j}^{l,l} \wedge \neg q_{i,j}^{l,r} \wedge \neg q_{i,j}^{r,l} \wedge \neg q_{i,j}^{r,r}) \Rightarrow 1_{i+1,j})$;
  - cztery analogiczne.

Zostało nam już bardzo niewiele — trzeba jakoś zaimplementować przejścia; napisać, że w każdej linii może być tylko jeden stan.

Załóżmy, że to, co pisze i w jakim kierunku idzie NDTM jest określone; jedynie wybór stanu pozostaje niedeterministyczny. Musimy teraz zadbać, aby:

- "stan aktualny w wierszu" był co najwyżej 1 (eliminując pary);
- by stan początkowy był jak należy:
  - $q_{1,0}$;
- by stan na końcu był akceptujący:
  - $q_{p(|x|),f}$.

Jeśli $\langle q, a, R, \langle q', q'' \rangle, a' \rangle$ jest instrukcją M_B, to...

(cdn)

18 Wykład 18 (27/05/2008)

18.1 Twierdzenie Cooka — ciąg dalszy

Teraz napiszemy klauzule wymuszające zgodność wartościowania z instrukcjami M_B.

Jeśli $\langle a, q, a', \langle q', q'' \rangle, R \rangle$ jest instrukcją M_B, to w φ_x będą następujące klauzule:

- $a_{i,j} \wedge q_{i,q} \Rightarrow \neg q_{i,j}^{l,l}$;
- $a_{i,j} \wedge q_{i,q} \Rightarrow \neg q_{i,j}^{r,l}$;
- $a_{i,j} \wedge q_{i,j}^{l,r} \Rightarrow (q_{i+1,q'} \vee q_{i+1,q''})$ — tu chowa się cały niedeterminizm.

Jak zmienia się znak na taśmie?

- $a_{i,j} \wedge q_{i,j}^{l,r} \wedge q_{i,q} \Rightarrow b_{i+1,j}$;
- $a_{i,j} \wedge q_{i,j}^{r,r} \wedge q_{i,q} \Rightarrow b_{i+1,j}$.

Teraz zajmiemy się stanem początkowym. Niech x_j oznacza j-ty bit x. Mamy zatem klauzule:

- jeśli $x_j = 0$ to $0_{1,j+1}$; w przeciwnym przypadku $1_{1,j+1}$;
- poza tym potrzebujemy jeszcze $\alpha_{1,1}, \omega_{1,|x|+2}, B_{1,|x|+3}$.

Umówmy się z tą maszyną, że jeśli dojdzie do stanu q_f, to już pozostanie w tym stanie. Dzięki temu możemy zostawić ją na czas $p(|x|)$, przyjść i spokojnie odczytać wynik.

Zbudowaliśmy zatem redukcję, która dla x będącego wejściem maszyny M_B zwróci formułę φ_x. Pozostaje udowodnić, że $M_B(x)$ akceptuje wtedy, gdy φ_x jest spełnialna. Załóżmy więc, że nasza maszyna akceptuje. Przebieg maszyny potrafimy odczytać z tabelki i zinterpretować jako wartościowanie zmiennych. Załóżmy teraz, że nasza formuła jest spełnialna. Oznacza to, że zmienne mieszkające w tabelce da się tak zwartościować, aby odtworzyć z nich akceptujący przebieg maszyny.

Zdefiniujmy sobie na przyszłość parę makr.

Niech p oznacza wielomian z dowodu twierdzenia Cooka; wtedy przez $\overline{p}$ oznaczymy liczbę zmiennych mieszkających w jednym wierszu tabelki.

Niech $\textsc{Początek}_x(s_1, \ldots, s_{\overline{p}(|x|)})$ będzie formułą w 6CNF, prawdziwą dla takich wartościowań zmiennych $s_1, \ldots, s_{\overline{p}(|x|)}$, które opisują pierwszy wiersz tabelki z wejściem x; podobnie $\textsc{Koniec}_x$.

Niech $\textsc{Krok}_x(s_1, \ldots, s_{\overline{p}(|x|)}, t_1, \ldots, t_{\overline{p}(|x|)})$ będzie formułą prawdziwą dla takich wartościowań, które opisują dwa sąsiednie wiersze tabelki.

Streszczenie formuły φ_x z twierdzenia Cooka w języku powyższych makr wyglądałoby tak:

- $\varphi_x = \text{Początek}_x(s_{1,1}, \ldots, s_{1,\overline{p}(|x|)}) \wedge Koniec_x(s_{p(|x|),1}, \ldots, s_{p(|x|),\overline{p}(|x|)}) \wedge Krok_x(s_{1_1}, \ldots, s_{1,\overline{p}(|x|)}, s_{2,1}, \ldots, s_{2,\overline{p}(|x|)}) \wedge Krok_x(s_{2,1}, \ldots, s_{2,\overline{p}(|x|)}, s_{3,1}, \ldots, s_{3,\overline{p}(|x|)}) \wedge \cdots \wedge Krok_x(s_{p(|x|)-1,1}, \ldots, s_{p(|x|)-1,p(|x|)}, s_{p(|x|),1}, \ldots, s_{p(|x|),\overline{p}(|x|)})$.

(uczytelnić powyższy zapis)

18.2 Klasa PSpace

Obserwacje

1. PSpace = co-PSpace — wystarczy zanegować wynik algorytmu;
2. PTime $\subseteq$ PSpace — z twierdzenia o brudzeniu garnków;
3. NP $\subseteq$ PSpace — dlaczego?

Mamy $A \in$ NP, więc niedeterministyczna wielomianowa M_A wykonuje w trakcie działania $\leqslant p(|x|)$ niedeterministycznych wyborów. Zbudujemy M tak:

- wczytaj x;
- zaznacz sobie pole wielkości $p(|x|)$;
- napisz tam liczbę $m = 0$;
- póki symulowana maszyna nie zaakceptuje i póki nie wykorzystane zostaną wszystkie możliwe m:
  - symuluj $M_A(x)$ używając ciągu bitów m jako "głosu z kratki";
  - wykonaj $m++$.

Wobec tego również:

3a. co-NP $\subseteq$ PSpace.

JMA: Tu będzie definicja, której nie będzie, bo każdy ją zgadnie.

ExpTime zależy od $2^p(|x|)$; 2ExpTime zależy od $2^{2^{p(|x|)}}$; oczywiście ExpTime = co-ExpTime.

Dalszy ciąg obserwacji

4. PSpace $\subseteq$ ExpTime.

JMA: Pan to widzi? Pan z długopisem w ustach.

???: Widzę.

JMA: No tak, jakby pan miał długopis w oku...

Na pewno wiemy, że ExpTime $\neq$ PTime. Następnym razem zaczniemy od wyjaśnienia losu zaginionej klasy NPSpace.

19 Wykład 19 (29/05/2008)

19.1 Przykład problemu z PSpace, o którym myślimy, że nie jest w NP

19.1.1 Kwantyfikowane formuły boolowskie

Uwaga Co to jest spełnialność?

- $\varphi = q \Rightarrow ((p \vee \neg q) \Rightarrow (\neg r \wedge q)) \vee p$;
- $\psi = \exists_p \exists_q \exists_r \varphi$

ψ jest spełnialna, gdy φ jest prawdziwa.

Definicja Kwantyfikowana formuła boolowska (QBF) to formuła boolowska, w której każda zmienna jest związana kwantyfikatorem.

Formuły boolowskie dzielimy na spełnialne i niespełnialne, a kwantyfikowane formuły boolowskie dzielimy na prawdziwe i fałszywe.

Przez TQBF postaramy się oznaczyć sobie podzbiór prawdziwych QBF.

Obserwacja TQBF $\subseteq$ PSpace.

Gdy mamy zwartościowane wszystkie zmienne, to oczywiście bez trudu potrafimy policzyć wartość formuły — w wielomianowym czasie i pamięci.

Formułę możemy wyobrazić sobie jako drzewo binarne o wysokości równej liczbie zmiennych. Kwantyfikatory są etykietami poziomów w drzewie. Każdy liść w tym drzewie oznacza pewne wartościowanie; toteż jeśli potrafimy dojść do liścia, to potrafimy obliczyć wartość formuły.

Nasz algorytm przegląda drzewo od lewej do prawej, zawsze pamiętając ciąg zer i jedynek będący ścieżką do aktualnego wierzchołka, oraz miejsce, z którego przyszedł do niego — czyli PSpace.

JMA: O, pani podnosi rękę.

???: Nie, nie podnosiłam ręki.

JMA: To nieprawda! ... To była pana ręka...?

Twierdzenie TQBF jest PSpace-zupełny (ze względu na redukcje wielomianowe).

Oczywiście jeśli $A \in$ PSpace i $B \leqslant_p A$, to $B \in$ PSpace — ze względu na składanie wielomianów.

Uwaga Gdyby jakiś problem PSpace-zupełny był w NP, to NP równałoby się PSpace.

19.2 Twierdzenie o tym, że PSpace = NPSpace

Dowód Niech $B \in$ NPSpace, czyli istnieje niedeterministyczna maszyna Turinga M_B, która w przestrzeni ograniczonej przez wielomian p rozstrzyga przynależność do B. Niech x będzie instancją B. Umówmy się, że istnieje jedna konfiguracja końcowa, c_F, no i oczywiście początkowa, c_0.

Wyobraźmy sobie wielki graf, którego wierzchołkami są wszystkie konfiguracje M_B, w których na taśmie jest $\leqslant p(|x|)$ znaków różnych od blanka. Krawędź z wierzchołka c_1 do c_2 będzie w tym grafie wtedy i tylko wtedy, gdy maszyna M_B z konfiguracji c_1 może przejść do konfiguracji c_2 w jednym kroku obliczeń.

JMA: Proszę się pytać mnie!

???: Nie, ja tylko pytałam, co jest napisane.

JMA: ... *Wszystko* jest napisane!

W zbudowanym grafie są 2 wyróżnione wierzchołki — c_0 i c_F. M_B akceptuje x gdy w naszym grafie istnieje ścieżka z c_0 do c_F. Graf ten z grubsza $5^{p(|x|)} \cdot p(|x|) \cdot |Q| \leqslant 2^{3p(|x|)}$ wierzchołków.

Wyobraźmy sobie, że jesteśmy tą wielomianową maszyną i umiemy pamiętać pewien wierzchołek naszego grafu. Jak nam powiedzą drugi wierzchołek, to umiemy ustalić, czy istnieje krawędź pomiędzy tymi wierzchołkami.

Wystarczy, że pokażemy działający w wielomianowej pamięci deterministyczny algorytm rozstrzygający problem istnienia ścieżki z ustalonego wierzchołka s do ustalonego wierzchołka t w grafie, którego nazwy wierzchołków mają długość ograniczoną przez wielomian p i takim, że dla danych dwóch nazw wierzchołków umiemy w przestrzeni wielomianowej ustalić, czy jest między nimi krawędź.

Ścieżka z s do t, jeśli istnieje, ma długość $\leqslant 2^{3p(|x|)}$.

Oto procedura, która sprawdza istnienie ścieżki o długości $\leqslant 2^{i+1}$ z s_1 do s_2:

- przeglądaj kolejne s_3:
  - sprawdź, czy istnieje ścieżka długości $\leqslant 2^i$ z s_1 do s_3;
  - jeśli tak, to sprawdź, czy istnieje ścieżka długości $\leqslant 2^i$ z s_3 do s_2.

Ile potrzebujemy pamięci? Tyle, co na przechowanie s_3 (czyli $p(|x|)$) plus tyle, co na wykonanie naszej procedury dla 2^i.

Żeby sprawdzić, czy istnieje ścieżka z s do t, potrzebujemy stosu wywołań o głębokości co najwyżej $3p(|x|)$. Zatem cały algorytm potrzebuje pamięci $3p^2(|x|)$.

Uwaga ExpSpace = NExpSpace.

Dowód *Takisam.* Itd.

20 Wykład 20 (3/06/2008)

Twierdzenie TQBF jest PSpace-zupełny.

Dowód Weźmy dowolny problem $D \in$ PSpace. Pokażemy, że $D \leqslant_p$ TQBF.

Wiemy, że istnieje taka maszyna Turinga M_D i taki wielomian p, że $\forall_{x \in \{0,1\}^*} M_D(x)$ akceptuje wtedy, gdy $x \in D$, zaś $M_D(x)$ używa nie więcej niż $p(|x|)$ komórek taśmy.

JMA: W każdym jej wierszu mieszka ile zmiennych?

AFU: Co?

JMA: Nie co, tylko ile.

Teraz pokażemy, jak dla danego x zbudować taką formułę $\varphi_x \in$ QBF, że $M_D(x)$ akceptuje wtedy, gdy $\varphi_x \in$ TQBF.

Wyobraźmy sobie teraz tabelkę o $p(|x|)$ kolumnach i $2^{3p(|x|)}$ wierszach — zgodnie z rachunkiem przyjętym na poprzednich spotkaniach. W każdym jej wierszu mieszka $\overline{p}$ zmiennych.

Mieliśmy formuły 'mówiące', że krotki wartości zmiennych mają jakąś własność:

- $\textsc{Początek}_x(s_1, \ldots, s_{\overline{p}(|x|)})$ mówi, że w wierszu "$s_1, \ldots, s_{\overline{p}(|x|)}$" tabelki 'jest' początkowa konfiguracja $M_D(x)$;
- $\textsc{Koniec}_x(s_1, \ldots, s_{\overline{p}(|x|)})$ 'mówi', że w wierszu "$s_1, \ldots, s_{\overline{p}(|x|)}$" jest końcowa konfiguracja $M_D(x)$;
- $\textsc{Krok}_x(s_1, \ldots, s_{\overline{p}(|x|)}, t_1, \ldots, t_{\overline{p}(|x|)})$ 'mówi', że "$s_1, \ldots, s_{\overline{p}(|x|)}$" i "$t_1, \ldots, t_{\overline{p}(|x|)}$" są dwiema kolejnymi konfiguracjami $M_D(x)$.

Zaczynamy pisać formułę φ_x:

- $\exists_{\overline{s},\overline{t}}\, \textsc{Początek}(\overline{s}) \wedge \textsc{Koniec}(\overline{t}) \wedge \textsc{Krok}^{3p(|x|)}(\overline{s}, \overline{t})$.

Pierwsza przymiarka do budowy formuły $\textsc{Krok}^{i+1}(\overline{s}, \overline{t})$:

- $\textsc{Krok}^{i+1}(\overline{s}, \overline{t}) = \exists_{\overline{u}} \textsc{Krok}^i(\overline{s}, \overline{u}) \wedge \textsc{Krok}^i(\overline{u}, \overline{t})$.

Niepokojące jest to, że po rozplątaniu takiej formuły znaleźlibyśmy w niej same kwantyfikatory egzystencjalne, co prowadzi nas do Sat $\in$ NP. Nie chcielibyśmy przypadkiem udowodnić, że PSpace $\in$ NP.

No to teraz zrobimy lepiej:

- $\textsc{Krok}^{i+1}(\overline{s}, \overline{t}) = \exists_{\overline{u}} \forall_{\overline{v},\overline{w}} (\overline{v} = \overline{s} \wedge \overline{w} = \overline{u}) \vee (\overline{v} = \overline{u} \wedge \overline{w} = \overline{t}) \Rightarrow \textsc{Krok}^i(\overline{v}, \overline{w})$.

Oczywiście wszyscy widzimy, że powyższa formuła jest krótsza, niż poprzednia.

20.1 Drugi fajny problem PSpace-zupełny

Totalność wyrażenia regularnego to problem, w którym dają nam wyrażenie regularne φ nad alfabetem Σ i pytają, czy $L_\varphi = \Sigma^*$. Jest to szczególny przypadek pytania o ekstensjonalną równość wyrażeń regularnych.

Twierdzenie Totalność wyrażeń regularnych (TotRE) jest PSpace-zupełna.

A gdyby do wyrażeń regularnych dodać potęgowanie lub przecięcie? Będą to nasze pierwsze problemy w ExpTime.

A co się stanie, gdy dodamy dopełnienie? Jeśli φ jest wyrażeniem regularnym nad Σ, to $\overline{\varphi}$ jest takim wyrażeniem regularnym, że $L_{\overline{\varphi}} = \Sigma^* \setminus L_\varphi$.

Twierdzenie Problem totalności wyrażeń regularnych z dopełnieniem nie daje się rozstrzygać w czasie ograniczonym przez 2 do 2 do ... do 2 (i tak l razy) do $p(|x|)$ — dla żadnego $l \in \mathbb{N}$.

(cdn)

21 Wykład 21 (5/06/2008)

21.1 PSpace-zupełność problemu totalności wyrażeń regularnych

Uwaga A jest PSpace-zupełny wtedy, gdy $\overline{A}$ jest PSpace-zupełne.

Istnieje taka wielomianowa $f : \{0,1\}^* \to \{0,1\}^*$, że $\forall_x\, x \in \overline{A} \Leftrightarrow f(x) \in A$.

Chcielibyśmy pokazać, że $A \leqslant_p \overline{A}$. Weźmy sobie f i spójrzmy na nią krzywo. Okazuje się, że $\forall_x\, x \notin \overline{A} \Leftrightarrow f(x) \notin A$, czyli $\forall_x\, x \in A \Leftrightarrow f(x) \in \overline{A}$.

Dowód Niech $A \in$ PSpace. Pokażemy, że $A \leqslant_p$ TotRE.

Wiemy, że istnieje taka maszyna Turinga M_A i taki wielomian p, że M_A rozstrzyga przynależność do A w pamięci ograniczonej przez $p(|x|)$, gdzie x oznacza słowo wejściowe.

Zbudujemy teraz takie wyrażenie regularne φ_x nad alfabetem Σ, że $L_{\varphi_x} = \Sigma^* \Leftrightarrow M_A(x)$ akceptuje. Wielkość φ_x będzie ograniczona przez jakiś wielomian od $|x|$.

Niech Q będzie zbiorem M_A. Niech $B = Q \cup \{0, 1, \alpha, \omega, B, \#\}$. Niech $\Sigma = B \times B$, czyli można sobie wyobrazić, że słowo składa się z dwóch warstw symboli z B.

Jak chcemy zakodować obliczenie maszyny $M_A(x)$? W górnej warstwie damy:

- $\underbrace{q_0\alpha \ldots x \ldots \omega B \ldots B}_{p(|x|)}\#$ konfiguracja 1 # konfiguracja 2 $\# \ldots \# \ldots q_F \ldots \#$.

W dolnej zaś:

- konfiguracja 1 # konfiguracja 2 $\# \ldots \# \ldots q_F \ldots \# \ldots q_F \ldots \#$.

Słowa, które nie kodują akceptującego obliczenia $M_A(x)$ nazwiemy *brzydkimi*. Twierdzenie będzie udowodnione, gdy napiszemy wielomianowej długości wyrażenie φ, oznaczające zbiór wszystkich brzydkich słów.

Czemu słowo może być brzydkie:

1. może się źle zaczynać w górnej warstwie;
2. może się źle kończyć;
3. jakieś kolejne dwa znaki (dwuwarstwowe) stanowią "brzydką czwórkę";
4. jakaś obietnica z i-tej konfiguracji w dolnej warstwie nie jest dotrzymana w $(i+1)$-tej konfiguracji w górnej warstwie.

Nasze wyrażenie φ będzie sumą następujących wyrażeń:

1. suma po $i = 0$ do $p(|x|)$ wyrażeń mówiących "ma nie tak w miejscu i-tym";
2. $\Sigma^*(\Sigma \setminus \left\{\binom{\#}{\#}\right\}) + \Sigma^*\binom{\#}{\#}(\Sigma \setminus \left\{\binom{\#}{\#}, \binom{q_F}{q_F}\right\})^*\binom{\#}{\#}$;
3. $\Sigma^*(a_1b_1 + a_2b_2 + \cdots + a_kb_k)\Sigma^*$;
4. suma po wszystkich takich parach $\binom{a}{b}, \binom{c}{d}$, że $b \neq c$ wyrażeń postaci $\Sigma^*\binom{a}{b}\underbrace{\Sigma\Sigma\ldots\Sigma}_{p(|x|)}\binom{c}{d}\Sigma^*$.

Twierdzenie Totalność wyrażeń regularnych z potęgowaniem jest ExpSpace-zupełna.

Dowód Przepisz dowód PSpace-zupełności problemu TotRE, modyfikując wyrażenie φ:

1. w wyrażeniu odpowiedzialnym za "brzydki początek" możemy opisać explicite tylko pierwsze $|x|+4$ znaków;
2. w wyrażeniu odpowiedzialnym za "brzydkie obietnice" możemy skorzystać z potęgowania, sumując wyrażenia postaci $\Sigma^*\binom{a}{b}\Sigma^{2^{p(|x|)}}\binom{c}{d}\Sigma^*$.

Czy ten problem w ogóle jest w ExpSpace? Pozbądźmy się potęgowania, rozpisując je. W stosunku do wykładniczej długości otrzymanego wyrażenia potrafimy rozstrzygnąć problem totalności w wielomianowej przestrzeni, co daje wykładniczą przestrzeń.

22 Wykład 22 (10/06/2008)

Twierdzenie PSpace $\subset$ ExpSpace $\wedge$ PSpace $\neq$ ExpSpace.

Zbudujemy sobie pewną bardzo dziwną maszynę Turinga M_{BD}. Ona będzie rozpoznawać pewien język L_{BD}. Oszacujemy przestrzeń, jakiej wymaga działanie M_B — w zupełności wystarczy jej przestrzeń wykładnicza. Następnie pokażemy, że dla każdego języka $L \in$ PSpace istnieje takie $a_L \in \{0,1\}^*$, że $a_L \in L \Leftrightarrow a_l \in L_{BD}$.

Dowód Niech $f : \mathbb{N} \to \mathbb{N}$ będzie dowolną funkcją o następujących własnościach (gdzie $\mathbb{N}$ skrótem dla $\{0,1\}^*$:

- f jest niemalejąca;
- f jest nieograniczona.
- $f(x) \leqslant \log(|x|)$, ale np. $\log^*(|x|)$ jeż jest dobra.

JMA: Widzę w Tobie trzy rzeczy... Śniadanie, obiad i kolację.

Wczujmy się teraz w działanie M_{BD}:

- wczytuję jakieś słowo $x \in \{0,1\}^*$;
- będę myślała o tym x na 3 sposoby:
  1. najzwyczajniej, jak o danej;
  2. jak o kodzie, czyli liście instrukcji pewnej maszyny M_x — ale nie o x tak będę myślała, tylko o $\overline{x}$:
     - $\overline{\varepsilon} = \varepsilon$; $\overline{0} = \varepsilon$; $\overline{1} = \varepsilon$;
     - $\overline{00x} = \overline{x}$;
     - $\overline{01x} = \overline{10x} = 0\overline{x}$;
     - $\overline{11x} = 1\overline{x}$;
  3. jak o wielomian $p_x = 1 + z + z^2 + \cdots + z^{f(x)}$.
- rezerwuję sobie $p_x(|x|)$ miejsca na taśmie;
- symuluje $M_x(x)$ w zarezerwowanej przestrzeni:
  - jeśli $M_x(x)$ akceptuje, to M_{BD} odrzuca;
  - jeśli $M_x(x)$ odrzuca, to M_{BD} akceptuje;
  - jeśli $M_x(x)$ się zapętla lub nie mieści w wyznaczonej przestrzeni, to M_{BD} akceptuje.

M_{BD} rozpoznaje pewien dziwaczny język L_{BD}.

Rozważmy jakiś $L \in$ PSpace. Pokażemy, że $L_{BD} \neq L$.

Co my wiemy o tym L? Rozpoznaje go pewna maszyna M_L o kodzie $\overline{y}$, działająca w przestrzeni ograniczonej przez pewien wielomian q. Niech $d \in \mathbb{N}$ będzie taka, że $p_d > q$.

Niech t będzie takie, że $\overline{t} = \overline{y}$ i takie, że $t > d$. Zatem $p_t > q$.

Co robi $M_{BD}(t)$:

- uruchomi M_L dla słowa t — na pewno starczy jej miejsca, ponieważ $t > d$;
- jeśli $M_L(t)$ akceptuje, to $M_{BD}(t)$ nie akceptuje — i *vice versa*.

Oznacza to, że L_{BD} jest inny, niż dowolny, dający wyliczyć się w wielomianowej pamięci.

Pozostało jeszcze oszacować przestrzeń, w jakiej działa maszyna M_{BD}. Wielkość ta daje się z góry oszacować przez:

- $A = f(|x|) \cdot |x|^{f}(|x|)$.

Ale ponieważ $f(|x|) \leqslant \log(||x||) = |||x|||$:

- $A \leqslant |||x||| \cdot |x|^{|||x|||} = |||x||| \cdot 2^{||x|| \cdot |||x|||}$

23 Wykład 23 (12/06/2008)

Brak.

24 Wykład 24 (17/06/2008)

24.1 Nieelementarność problemu totalności dla wyrażeń regularnych z dopełnieniem (TotCom)

Problem jest elementarny, jeśli istnieje taka liczba naturalna l, że ten problem daje się rozwiązywać w czasie ograniczonym przez $\Delta_l^{|x|} = 2^{2^{2^{\cdots^{2^{|w|}}}}}$, gdzie l oznacza liczbę dwójek.

Niech A daje się rozstrzygać w czasie ograniczonym przez $\Delta_l^{|w|}$. Pokażemy, że $A \leqslant_p$ TotCom.

Niech M_A rozstrzyga przynależność do A w czasie $\Delta_l^{|w|}$. Będziemy sobie wyobrażać obliczenia tej maszyny jako bardzo długie słowa, podzielone na sektory, z których każdy będzie zawierał kolejne konfiguracje maszyny M_A. Naszym celem jest napisanie takiego wyrażenie regularnego z dopełnieniem φ_w, że L_{φ_w} będzie oznaczać zbiór wszystkich brzydkich słów — czyli takich, które nie kodują akceptującego obliczenia $M_A(w)$.

Z jakich kawałków składa się φ_w i które z nich umiemy od ręki napisać? Łatwo napisać podwyrażenia φ_w oznaczające słowa, które się brzydko zaczynają i te, które się brzydko kończą. Ponownie, tak jak w wykładzie 21., skorzystamy z "brzydkich czwórek" i "brzydkich obietnic", w których odległość między odpowiadającymi sobie znakami będzie równa $\Delta_l^{|w|}$.

Jedyna rzecz, której nam brakuje, to patent do mierzenia bardzo długich rzeczy.

24.1.1 Jak mierzyć takie strasznie wielkie odległości

Poprzednio zdefiniowane warstwy nazwiemy piwnicą i parterem. Nadbudujemy na wierzch jeszcze l pięter. Alfabet symboli występujących na piętrach to $\{0, 1, \underline{0}, \underline{1}, \#\}$.

Umówmy się, że jeśli na piętrze $k+1$ jest # i słowo jest ładne, to pod nim na piętrze k też jest # (nie licząc parteru).

Słowo jest ładne, jeśli ma "ładną piwnicę i parter", a ponadto na 1. piętrze musi mieć wypisane kolejne zerojedynkowe liczby długości $|w|$, oddzielone #, począwszy od liczby zero. Ponadto podkreślone są symbole (zera i jedynki), które w kolejnym sektorze się zmienią — wobec czego bardzo łatwo zweryfikować, czy dany sektor jest ładnie podkreślony.

JMA: Bardzo ciężko się mówi o idei, jeśli się nie powie o technice. Ale równie ciężko się mówi o technice, jeśli się nie powie o idei. Ale czy to oznacza, że należy milczeć?

Jeśli na jakimś piętrze znajduje się wzorzec $\#(1+\underline{1})^*\#$, to nad prawym # tego wzorca piętro wyżej też jest #. To są jedyne # piętro wyżej. Na każdym piętrze zaczynamy oczywiście od bloku samych zer.

Słowo jest k-ładne jeśli jest ładne na piętrach $1, \ldots, k$, zaś jest k-brzydkie, jeśli nie jest k-ładne. Czy umiemy napisać wyrażenie oznaczające wszystkie słowa 1-brzydkie? Z punktu widzenia 1. piętra:

- $\Sigma^* \underline{1} \underbrace{\Sigma\Sigma\ldots\Sigma}_{|w|}(1+\underline{1}+\#)\Sigma^*$;
- $\Sigma^* \underline{0} \underbrace{\Sigma\Sigma\ldots\Sigma}_{|w|}(0+\underline{0}+\#)\Sigma^*$;
- $\Sigma^* 1 \underbrace{\Sigma\Sigma\ldots\Sigma}_{|w|}(0+\underline{0}+\#)\Sigma^*$;
- $\Sigma^* 0 \underbrace{\Sigma\Sigma\ldots\Sigma}_{|w|}(1+\underline{1}+\#)\Sigma^*$;

Dla każdego poziomu napiszemy 3 wyrażenia:

1. pełny cykl na poziomie i (zaczynający się od bloku zer i kończący na bloku jedynek i #): γ_i;
2. przesunięty γ_i: α_i;
3. jeden dowolny blok na poziomie i: β_i.

JMA: Paweł umie? Niech pan się naumie!

Zakładamy, że mamy napisane te wyrażenia dla jakiegoś i i chcemy je napisać dla $i+1$. Łatwo zauważyć, że z dokładnością do równości symboli na różnych piętrach zachodzi równość:

- $\beta_{i+1} = \gamma_i$.

Napiszmy teraz wyrażenie oznaczające wszystkie słowa $(i+1)$-brzydkie:

- $\Sigma^*(\alpha_i \cap \underline{1}\Sigma^*)1\Sigma^*$;

Następnie:

- $\gamma_{i+1} = \beta^*_{i+1} \cap$ "$(i+1)$-ładne" $\cap$ zaczyna się na poziomie $i+1$ od bloku zer $\cap$ kończy się na poziomie $i+1$ na bloku jedynek ma dokładnie jeden blok jedynek;

Zostało jeszcze tylko pokazać, że umiem sobie przesunąć γ:

- $\alpha_{i+1} = \alpha^*_i \cap$ "$(i+1)$-ładne";
- powiedzmy, że resztę też umiem.

Co, jeśli będę już miał α_l? Po co ja to w ogóle wszystko robiłem?