

Wykład 2. Dowody Interaktywne

Wykładowca: Stefan Dziembowski

Skryba: Stanisław Wierchoła

Streszczenie. W trakcie wykładu wprowadzimy pojęcie dowodu interaktywnego i klasy języków mających dowody interaktywne. Następnie udowodnimy ważne twierdzenie które charakteryzuje tą klasę jako równoważną PSPACE (Twierdzenie Shamira).

1 Wstęp

Dowód jest jednym z fundamentalnych pojęć w matematyce i informatyce. Tradycyjnie dla danego zbioru aksjomatów i reguł wnioskowania, *dowodem*(proof) twierdzenia jest odpowiedni ciąg formuł. Gdyby wszystkie twierdzenia, prawdziwe w danej teorii, przedstawić jako łańcuchy znaków nad skończonym alfabetem to o teorii moglibyśmy myśleć jak o języku L nad tym alfabetem. Dowód - proof będziemy również reprezentować jako słowo nad tym alfabetem.

Systemem dowodzenia dla języka L będzie algorytm weryfikacji V (weryfikator), taki że:

1. (Pełność) Twierdzenia prawdziwe mają dowody. Jeśli dowolne $x \in L$, to istnieje proof takie, że $V(x, \text{proof}) = \text{true}$
2. (Poprawność) Twierdzenia fałszywe nie mają dowodów. Jeśli jakieś $x \notin L$, to dla każdego proof^* , zachodzi $V(x, \text{proof}^*) = \text{false}$

W dalszej części wykładu interesować nas będą systemy dowodzenia, w których weryfikator działa w czasie wielomianowym. Jak nietrudno spostrzec, klasa języków dla których istnieją systemy dowodzenia, w których weryfikator V jest deterministyczną maszyną Turinga, jest równoważna klasie NP.

Zauważmy, że nasza definicja systemu dowodzenia, nie nakłada żadnego ograniczenia na trudność znalezienia takiego proof, o którym mowa w warunku *Pełność*. Możemy więc myśleć, że proof zostało wyprodukowane przez jakąś maszynę Turinga o nieograniczonej mocy obliczeniowej. Jej „zadaniem” jest przekonanie weryfikatora o tym, że $x \in L$. Taką maszynę nazwiemy P - udowadniaczem (ang.: *prover*).

Uogólnimy teraz pojęcie systemu dowodzenia:

- Weryfikator V może być niedeterministyczną maszyną Turinga
- Zamiast wysyłania jednego dowodu proof od udowadniacza P do weryfikatora V , dopuszczamy wielokrotną iterację
- Weryfikator V , może z odpowiednio małym prawdopodobieństwem akceptować elementy nie należące do języka

Definicja 1 Niech $L \subseteq \{0, 1\}^*$.

Systemem dowodu interaktywnego dla języka L nazywamy parę (V, P) , gdzie:

Weryfikator V — dowolna zrandomizowana interaktywna maszyna Turinga działająca w czasie wielomianowym.

Udowadniacz (ang.: Prover) P — dowolna interaktywna maszyna Turinga.

Maszyny V i P dostają wspólne wejście x i wchodzi z sobą w interakcję. Iteracja polega na wymianie ciągu wiadomości m_1, m_2, \dots, m_k , przy czym P wysyła wiadomości o nieparzystych numerach, a V o parzystych. W rezultacie V zwraca wartość true albo false.

Formalnie wiadomości definiujemy w następujący sposób: $m_1 = P(x)$ - oznacza to że pierwsza wiadomość jest tworzona przez Udowadniacz na podstawie słowa wejściowego. Następnie mamy $m_{2i} = V(x, m_1 \dots m_{2i-1}, r_i)$ oraz $m_{2i-1} = P(x, m_1 \dots m_{2i-2})$, gdzie r_i oznacza ciąg losowy wielomianowej długości wykorzystywany przez Weryfikator w trakcie i -tej wymiany. Na koniec jeśli ostatnią wiadomością m_k wysłaną przez V jest true lub false, zwraca on że dane wejściowe są akceptowane lub odrzucane. Niech $(V, P)(x)$ oznacza tę wartość.

Powiemy że (V, P) akceptuje język L jeśli:

1. (Pełność) Jeśli dowolne $x \in L$, to $(V, P)(x) = \text{true}$.

2. (Poprawność) Jeśli jakieś $x \notin L$, to dla dowolnego P^* mamy $(V, P^*)(x) = \text{false}$ z prawdopodobieństwem co najmniej 0.5.

Uwaga: 0.5 jest wyborem dość arbitralnym. Prawdopodobieństwo tego że P oszuka V możemy dowolnie zmniejszyć wykonując protokół wielokrotnie.

Definicja 2 Dla dowolnej funkcji $f : \mathcal{N} \rightarrow \mathcal{N}$, $\mathcal{IP}(f(\cdot))$ oznacza rodzinę języków mających dowód interaktywny, taki, że dla dowolnego x strony wymieniają co najwyżej $f(x)$ wiadomości.

Niech:

$$\mathcal{IP} := \bigcup_{p \in \text{poly}} \mathcal{IP}(\text{poly})$$

gdzie poly oznacza zbiór wszystkich wielomianów.

Zauważmy, że:

$$\text{NP} \subset \mathcal{IP}(1)$$

W dalszej części wykładu rozważać będziemy jak klasa \mathcal{IP} ma się w stosunku do innych klas złożoności.

2 Przykład dowodu interaktywnego

Aby zilustrować siłę dowodów interaktywnych pokażemy dowód interaktywny dla problemu *nieizomorfizmu grafów*.

Definicja 3 Mamy dane dwa grafy $G = (V, E)$ i $G' = (V, E')$ o takim samym zbiorze wierzchołków V . Są one izomorficzne, jeśli istnieje permutacja π zbioru V , taka że $G' = \pi(G)$. Przez $\pi(G)$ oznaczamy graf $(V, [\pi(u), \pi(v)] : [u, v] \in E)$. Izomorfizm grafów G i G' zapisujemy:

$$G \cong G'$$

Izomorfizmem grafów nazywamy język $\text{GI} := \{(G, H) : G \cong H\}$.

Nieizomorfizmem grafów nazywamy język GNI będący uzupełnieniem języka GI .

Problem izomorfizmu grafów jest problemem NP (mając świadka w postaci permutacji łatwo możemy stwierdzić czy dwa grafy są izomorficzne), nie wiemy natomiast czy jest problemem NP -zupełnym i czy należy do P . Analogicznie o problemie nieizomorfizmu grafów nie potrafimy powiedzieć czy jest w NP .

Potrafimy jednak pokazać, że problem nieizomorfizmu grafów jest w \mathcal{IP} [2].

Twierdzenie 4 $\text{GNI} \in \mathcal{IP}$.

Dowód:

Dowód należenia języka do klasy \mathcal{IP} będzie polegał na opisanu protokołu pomiędzy weryfikatorem V i udowadniającym P , a następnie pokazaniu jego Pełności i Poprawności. Protokół wygląda tak:

Wejście: grafy $G_0 = (\{1, \dots, n\}, E_0)$ i $G_1 = (\{1, \dots, n\}, E_1)$.

1. V : wybierz losowo $b \in \{0, 1\}$ i permutację na $\{1, \dots, n\}$. Niech $H = \pi(G_b)$. Wyślij H do P .
2. P : jeśli $G_0 \cong H$, to niech $c := 0$, wpp: $c = 1$. Wyślij c do V
3. V : Jeśli $c = b$ to zwróć true , wpp zwróć false .

Pełność: jeśli $G_0 \not\cong G_1$, to $G_0 \cong H$ wtedy i tylko wtedy gdy $b = 0$, więc V zaakceptuje zawsze.

Poprawność: jeśli $G_0 \cong G_1$, to H ma identyczny rozkład, przy $b = 0$ i $b = 1$ (zostawiamy, to jako ćwiczenie dla czytelnika). Zatem P ma szansę 0.5 zgadnięcia b poprawnie. □

Uwaga: powyższy dowód pokazuje nawet, że $\text{GNI} \in \mathcal{IP}(2)$.

3 Twierdzenie Shamira

W tym rozdziale pokażemy następujące fundamentalne twierdzenie charakteryzujące klasę \mathcal{IP}

Twierdzenie 5 $\mathcal{IP} = \text{PSPACE}$. [4]

Dowód: (Zarys)

Będziemy dowodzić $\text{PSPACE} \subseteq \mathcal{IP}$. Dowód zawierania w drugą stronę pozostawiamy jako ćwiczenie.

Dowód będzie polegał na pokazaniu, że problem prawdziwości *Kwantyfikowanych Formuł Boolowskich* (ang.: *Quantified Boolean Formulas QBF*), który jest problemem zupełnym w PSPACE należy do \mathcal{IP} .

Definicja 6 Kwantyfikowana Formuła Boolowska (ang.: Quantified Boolean Formulas QBF) może być:

- zmienną zdaniową, taką jak x_i
- wyrażeniem postaci $\neg x_i$, gdzie x_i zmienną zdaniową
- wyrażeniem postaci $(\phi_1 \vee \phi_2)$ lub $(\phi_1 \wedge \phi_2)$, gdzie ϕ_1 i ϕ_2 są kwantyfikowanymi formułami boolowskimi
- wyrażeniem postaci $\forall_{x_i} \phi_1$ lub $\exists_{x_i} \phi_1$ gdzie x_i jest zmienną zdaniową, a ϕ_1 kwantyfikowaną formułą boolowską

QBF jest zamknięta jeśli nie zawiera zmiennych wolnych. W dalszej części rozważań będziemy używać pojęcia QBF mając na myśli zamknięte QBF.

Definiujemy $\text{QBFT} \subset \text{QBF}$ jako

$$\text{QBFT} := \{\varphi : \varphi \text{ jest prawdziwe}\}$$

Problem ten jest PSPACE -zupełny (dowód np. w [3])

Dowód dla szczególnego przypadku

Aby lepiej zilustrować ideę dowodu rozważymy najpierw nieformalnie szczególny przypadek, gdy formuła jest postaci:

$$\varphi = \exists x_0 \cdots \exists x_n. \psi(x_0, \dots, x_n), \quad (1)$$

gdzie $\psi(x_0, \dots, x_n)$ jest formułą bez kwantyfikatorów.

Problem spełnialności formuł w postaci φ równoważny jest problemowi *SAT*, który jest problemem NP-zupełnym. Nas interesować będzie uzupełnienie tego problemu, które jest oczywiście coNP-zupełne. Pokażemy dowód interaktywny tego, że formuła φ postaci (1) jest fałszywa.

Kluczowym pomysłem jest *arytmetyzacja* formuły φ . Arytmetyzacja polegać będzie na stworzeniu wyrażenia arytmetycznego α_φ równoważnego formule φ . Zaczynamy od przerobienia $\psi(x_0, \dots, x_n)$ na wyrażenie arytmetyczne ze zmiennymi x_0, \dots, x_n . Robimy to przez indukcję po strukturze:

- $\alpha_{x_i} = x_i$
- $\alpha_{\neg x_i} := 1 - x_i$
- $\alpha_{\psi_0 \vee \psi_1} := (\alpha_{\psi_0}) + (\alpha_{\psi_1})$
- $\alpha_{\psi_0 \wedge \psi_1} := (\alpha_{\psi_0}) \cdot (\alpha_{\psi_1})$

Następnie przerabiamy każdy kwantyfikator \exists_{x_i} na $\sum_{x_i \in \{0,1\}}$

(gdzie $\sum_{x_i \in \{0,1\}} \alpha(x) = \alpha(0) + \alpha(1)$).

W wyniku arytmetyzacji φ otrzymujemy:

$$\alpha_\varphi := \sum_{x_1 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \alpha_\psi(x_1, \dots, x_n)$$

Otrzymane wyrażenie α_φ ma dwie ważne własności:

- α_φ jest z góry ograniczone przez jakieś $p > 2^n$. Wynika to w dość oczywisty sposób ze sposobu w jaki konstruujemy wyrażenie.

- zachodzi również:

$$\varphi \text{ jest prawdziwe wtedy i tylko wtedy gdy } \alpha_\varphi \neq 0. \quad (2)$$

Aby pokazać (2) wystarczy udowodnić, że dla ustalonego wartościowania zmiennych x_1, \dots, x_n w $\{0, 1\}$ zachodzi:

$$\varphi(x_1, \dots, x_n) \text{ jest prawdziwe wtedy i tylko wtedy gdy } \alpha_\varphi(x_1, \dots, x_n) \neq 0. \quad (3)$$

Dowód: Zastosujemy indukcję strukturalną po budowie φ .

Powyższa własność jest oczywiście prawdziwa dla literalów. Rozpatrzmy formułę $\varphi = \varphi_1 \vee \varphi_2$. Jest ona prawdziwa jeśli φ_1 lub φ_2 ma wartość true, co na mocy założenia indukcyjnego zachodzi wtedy i tylko wtedy gdy przynajmniej jedno z wyrażeń α_{φ_1} , α_{φ_2} ma wartość dodatnią. To z kolei zachodzi wtedy i tylko wtedy gdy $\alpha_\varphi = \alpha_{\varphi_1} + \alpha_{\varphi_2}$ ma wartość dodatnią. Dla \wedge dowód przebiega analogicznie. \square

Zatem wystarczy pokazać dowód interaktywny faktu, że arytmetyzacja wyrażenia φ postaci (1) ma wartość 0.

Wejście: Formuła $\varphi = \exists x_0 \dots \exists x_n. \psi(x_0, \dots, x_n)$

1. P, V : Niech F_p będzie jakimś ciałem ($p > 2^n$), $d = |\varphi|$ i $\alpha_\varphi = \sum_{x_1 \in \{0,1\}} \dots \sum_{x_n \in \{0,1\}} \alpha_\psi(x_1, \dots, x_n)$ arytmetyzacja formuły φ .

2. P : Usuwaamy z α_φ pierwsze \sum_{x_1} . Otrzymujemy wielomian p_1 względem zmiennej z :

$$p_1(z) := \sum_{x_2 \in \{0,1\}} \dots \sum_{x_n \in \{0,1\}} \alpha_\psi(z, x_2, \dots, x_n)$$

P dysponuje wystarczającą mocą obliczeniową żeby policzyć $p_1(z)$. $p_1(z)$ może być stopnia co najwyżej n więc P może go wysłać do V .

3. V : sprawdza, czy $p_1(0) + p_1(1) = 0$. Jeśli nie to kończy i zwraca false.

4. V : wybiera losowe $r_1 \in F_p$ i wysyła je do P .

5. V : oblicza $v_1 = p_1(r_1)$

6. P, V : Dla $i = 2$ do n wykonujemy:

(a) P : Pozbywamy się kolejnego \sum_x i otrzymujemy wielomian:

$$p_i(z) := \sum_{x_i \in \{0,1\}} \dots \sum_{x_n \in \{0,1\}} \alpha_\psi(r_1, \dots, r_{i-1}, z, x_{i+1}, \dots, x_n)$$

(b) V : sprawdza, czy $p_i(0) + p_i(1) = v_{i-1}$. Jeśli nie to kończy i zwraca false.

(c) V : wybiera losowe $r_i \in F_p$ i wysyła je do P .

(d) V : oblicza $v_{i-1} = p_i(r_i)$

7. V : Akceptuje jeśli $p_n(r_i) = \alpha_\psi(r_1, \dots, r_n)$

Dowód:

Pełność: Protokół jest tak skonstruowany że jeśli φ jest fałszywe co oznacza że $\alpha_\varphi = 0$ i P zachowuje się jak należy to V przejdzie wszystkie testy i zwróci $(V, P) = \text{true}$. Zauważmy $p_1(0) + p_1(1) = 0$, $p_i(0) + p_i(1) = v_{i-1}$ dla każdego $i > 1$, i $p_n(r_i) = \alpha_\psi(r_1, \dots, r_n)$.

Poprawność: Jeśli φ jest prawdziwe, to wówczas $\alpha_\varphi = v \neq 0$. Pokażemy że bez względu na przyjętą strategię P^* nie będzie potrafił przekonać V że $\alpha_\varphi = 0$ z odpowiednio dużym prawdopodobieństwem.

Niech $p_1(z), \dots, p_n(z)$ oznacza wielomiany policzone prawidłowo, a $p_1^*(z), \dots, p_n^*(z)$ wysłane przez P^* (dla ciągu losowego r_1, \dots, r_n). W pierwszej rundzie bez względu na wysłane przez P^* p_1^* będzie zachodzić: (a) $p_1^*(0) + p_1^*(1) \neq 0$, lub (b) $p_1^* \neq p_1$. W przypadku (a) V zwróci od razu false. Do rozpatrzenia pozostaje przypadek gdy nie zachodzi (a) i zachodzi (b). Nie zachodzi (a), więc p_1^* przechodzi test $p_1^*(0) + p_1^*(1) = 0$. Dla wybranej przez V losowej wartości $r_1 \in F_p$ z dużym prawdopodobieństwem ($\geq 1 - d/|F_p|$) $p_1(r_1)^* \neq p_1(r_1)$ (ponieważ wielomiany p_1^* i p_1 są stopnia co najwyżej d , zachodzi $p_1^*(r) = p_1(r)$ dla co najwyżej d wartości, a losujemy je ze zbioru F_p). Zauważmy że jeśli P będzie miał szczęście w losowaniu i $p_1(r_1)^* = p_1(r_1)$ to uda mu się oszukać V . Wtedy wystarczy że w następnych rundach będzie mu wysyłał wielomiany a $p_2^*(z) = p_2(z), \dots, p_n^*(z) = p_n(z)$.

Jeśli P nie udało się oszukać V w żadnej z $i-1$ rund to w i -tej musi oszukiwać dalej. Analiza i -tej rundy przebiegać będzie analogicznie jak pierwszej. Z założenia o poprzednich rundach wiemy że:

$$p_{i-1}^*(r_{i-1}) \neq p_{i-1}(r_{i-1}) = p_i(0) + p_i(1)$$

Bez względu na wysłane przez P^* p_i^* musi zachodzić: (a) $p_i^*(0) + p_i^*(1) \neq p_{i-1}^*(r_{i-1})$, lub (b) $p_i^* \neq p_i$. W przypadku (a) V zwróci od razu false. Do rozpatrzenia pozostaje przypadek gdy nie zachodzi (a) i zachodzi (b). Nie zachodzi (a), więc p_i^* przechodzi test $p_i^*(0) + p_i^*(1) = 0$. Dla wybranej przez V losowej wartości $r_i \in F_p$ z dużym prawdopodobieństwem ($\geq 1 - d/|F_p|$) $p_i(r_i)^* \neq p_i(r_i)$.

Po wykonaniu n -kroków, z prawdopodobieństwem co najmniej $1 - n * d/|F_p|$, weryfikator zwróci false lub $p_n^*(r_n) \neq p_n(r_n) = \alpha_\psi(x_1, \dots, x_n)$. \square

Przejdźmy teraz do ogólnego przypadku

Skonstruujemy dowód iteracyjny dla języka QBFT. Główna idea dowodu będzie taka sama jak w szczególnym przypadku. Zanim przejdziemy do konstrukcji protokołu, musimy rozważyć kilka zagadnień.

Na początku musimy rozszerzyć arytmetyzację formuły φ . Może mieć ona teraz dowolną postać i zawierać kwantyfikatory uniwersalne. Będziemy je arytmetyzować w następujący sposób:

- $\alpha_{\forall_x \varphi} = \prod_{x \in \{0,1\}} \alpha_\varphi$

(gdzie $\prod_{x \in \{0,1\}} \alpha(x) = \alpha(0) \cdot \alpha(1)$).

Pojawiają się problemy:

1. Wartość α_φ może być podwójnie wykładnicza ze względu na $|\varphi|$. Przykład: jeśli

$$\varphi = \forall_{x_1} \dots \forall_{x_n} (\text{true} \vee \text{true})$$

to wartość α_φ wynosi

$$\left(\left((2^2)^2 \right)^2 \dots \right)^2 = 2^{2^n}$$

Weryfikator V , działający w czasie wielomianowym, będzie mieć problemy z liczbami podwójnie wykładniczymi.

2. Stopień wielomianu może być wykładniczy, np. jeśli

$$\varphi = \forall_{x_1} \dots \forall_{x_n} (x_1 \vee \dots \vee x_n)$$

to dostajemy

$$\alpha_\varphi = x_1^{2^n}.$$

Zatem wielomian jest za długi, żeby go przesłać weryfikatorowi!

Zanim rozwiążemy pierwszy problem udowodnimy następujący lemat:

Lemat 7 *Jeśli wartość wyrażenia α_φ , o długości n jest różna od zera to istnieje liczba pierwsza p z przedziału od 2^n do $2^{3 \cdot n}$, taka że $\alpha_\varphi \not\equiv 0 \pmod p$*

Dowód: Przypuśćmy, że $\alpha_\varphi \equiv 0 \pmod p$ dla wszystkich liczb pierwszych p z przedziału od 2^n do $2^{3 \cdot n}$. Z chińskiego twierdzenia o resztach wynika, że α_φ ma również resztę z dzielenia równą zero modulo iloczyn tych liczb. Pokażemy, że ten iloczyn jest większy niż wartość α_φ , czyli reszta z dzielenia nie może wynosić zero.

Łatwo można zauważyć że wartość α_φ nie może być większa niż 2^{2^n} . Każda operacja może oznaczać co najwyżej podniesienie do kwadratu wartości wyrażenia, a operacji jest co najwyżej n .

Liczb pierwszych z przedziału od 2^n do $2^{3 \cdot n}$ jest co najmniej 2^n , czyli ich iloczyn jest większy niż 2^{2^n} . Więcej informacji na temat dowodu tego faktu można znaleźć w Papadimitru. \square

Z lematu wynika że cały protokół możemy przeprowadzić modulo liczba p z przedziału od 2^n do $2^{3 \cdot n}$. Każemy więc prooverowi pokazać takie p i przedstawić weryfikatorowi. Weryfikator może sprawdzić, czy p jest liczbą pierwszą (od niedawna znany jest deterministyczny test pierwszość). Alternatywnie P może przesłać V dowód pierwszości p .

Z drugim problemem radzimy sobie wprowadzając pojęcie prostych kwantyfikowanych formuł boolowskich.

Definicja 8 *Formuła QBF jest prosta jeśli wystąpienie każdej zmiennej x_i nie jest w niej oddzielone od miejsca swej kwantyfikacji przez więcej niż jeden kwantyfiaktor uniwersalny.*

Przykład prostej formuły:

$$\forall x_1 \forall x_2 \exists x_3 [(x_1 \vee x_2) \wedge \forall x_4 (x_2 \wedge x_3 \wedge x_4)]$$

Przykład formuły która nie jest prosta

$$\forall x_1 \forall x_2 [(x_1 \wedge x_2) \wedge \forall x_3 (\neg x_1 \wedge x_3)]$$

(z powodu x_1).

Lemat 9 Każda formuła QBF o rozmiarze n może być wydajnie przekształcona na prostą formułę QBF o rozmiarze wielomianowym względem n .

Dowód: Rozpatrzmy dowolny kwantyfikator uniwersalny $\forall y$ i dowolną zmienną która jest związana z kwantyfikatorem stojącym przed $\forall y$ i występuje w formule po $\forall y$. Formuła φ ma zatem postać $\dots Qx \dots \forall y \psi(x)$. Przekształcamy ją następująco:

$$\dots Qx \dots \forall y \exists x' ((x = x') \wedge \psi(x'))$$

gdzie '=' jest zdefiniowane w standardowy sposób za pomocą ' \vee ' i ' \wedge '.

Algorytm zamieniania będzie działać w następujący sposób. Przeglądamy formułę φ od lewej strony. Po napotkaniu $\forall y$ dla każdej zmiennej x związanej kwantyfikatorem z lewej strony $\forall y$ i występującej z prawej strony $\forall y$ wprowadzamy x' . Dla każdego napotkanego $\forall y$ taką operację musimy powtórzyć co najwyżej raz dla każdej z występujących zmiennych.

Wprowadzanie nowych zmiennych nie zwiększa ilości zmiennych które musimy rozpatrywać. Zauważmy że wprowadzenie zmiennej x' dla x , powoduje że dla następnego kwantyfikatora uniwersalnego po jego prawej stronie nie będzie już zmiennej x , tylko x' . Algorytm będzie działać w czasie $O(n^2)$. \square

Przykład:

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \exists x_5 \dots Q(x_1, x_2, x_3, x_4, x_5, \dots)$$

przekształcamy na:

$$\exists x_1^0 \forall x_2^0 \exists x_1^1 (x_1^1 = x_1^0 \wedge \exists x_3^0 \forall x_4^0 \exists x_1^2 (x_1^2 = x_1^1 \wedge \exists x_2^1 (x_2^1 = x_2^0 \wedge \exists x_3^1 (x_3^1 = x_3^0 \wedge \exists x_5^0 \dots Q(x_1^2, x_2^1, x_3^1, x_4^0, x_5^0, \dots))))))$$

Teraz możemy założyć, że na wejściu P i V dostają prostą formułę φ .

Wejście: Formuła prosta φ , o długości n .

Dla skrócenia zapisu zamiast α_φ będziemy używać α

1. P : wybierz liczbę pierwszą p z przedziału od 2^n do $2^{3 \cdot n}$ taką, że $\alpha \neq 0 \pmod p$. Wyślij p do V .
2. V : sprawdź czy p jest liczbą pierwszą. Jeśli nie, to zwróć false i zakończ.
3. P oblicza wartość v wyrażenia α i wysyła ją do V następnie P i V wykonują protokół $\text{eval}(\alpha, v)$ zdefiniowany rekurencyjnie następująco:
 - (a) Jeśli $\alpha = \alpha_0 + \alpha_1$, to P oblicza wartości v_0 i v_1 wyrażen α_0 i α_1 , odpowiednio; wysyła je do V ; V sprawdza, czy $v = v_0 + v_1$ i wykonywane jest $\text{eval}(\alpha_0)$ oraz $\text{eval}(\alpha_1)$.
 - (b) Jeśli $\alpha = \alpha_0 \cdot \alpha_1$ — to podobnie
 - (c) Jeśli $\alpha = x$, gdzie x jest jakąś liczbą, i $x \neq v$, to zwracane jest false i następuje koniec.
 - (d) Jeśli $\alpha = \sum_{x \in \{0,1\}} \alpha'(x)$, to P oblicza wielomian $p(x) = \alpha'(x)$ i przesyła go w postaci kanonicznej do V . Zauważmy, że stopień zmiennej x w wielomianie $p(x)$ jest liniowy względem długości α . (wynika to z tego że φ jest formułą prostą).
Zatem opis p jest wielomianowej długości.
 V sprawdza, czy $p(0) + p(1) = v$ — jeśli nie, to zwraca false i następuje koniec.
Wpp. V wysyła do P losową liczbę r z przedziału $\{0, \dots, p-1\}$.
 P oblicza $v' = f(r)$, wysyła je do V .
 V sprawdza, czy $v' = f(r)$, jeśli nie, to zwraca false i następuje koniec.
Wpp. wywoływane jest $\text{eval}(\alpha', v')$, gdzie α' powstaje z α przez podstawienie na zmienną x wartości r .
 - (e) Jeśli $\alpha = \prod_{x \in \{0,1\}} \alpha'(x)$ — to analogicznie.

Jeśli podczas wywołania tej procedury v nigdy nie zwróci false, to na koniec zwraca true.

Pełność: Protokół jest tak skonstruowany że jeśli φ jest fałszywe co oznacza że $\alpha_\varphi = 0$ i P zachowuje się jak należy to V przejdzie wszystkie testy i zwróci $(V, P) = \text{true}$.

Poprawność: Dowód należy przeprowadzić przy użyciu indukcji, analogicznie jak w rozważanym wcześniej prostym przypadku.

Zainteresowany czytelnik znajdzie więcej szczegółów w [5], [1], [4] na których zostały oparte powyższe notatki. □

Czytelnikom zainteresowanym głębszym i szerszym zrozumieniem tematyki przedstawionej na tym wykładzie polecamy znakomicie napisany i wydany w serii "Klasyka Informatyki" podręcznik Christosa H. Papadimitriou "Złożoność obliczeniowa" [3].

Literatura

- [1] Oded Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*, volume 17 of *Algorithms and Combinatorics*. Springer-Verlag, 1998.
- [2] Avi Wigderson Oded Goldreich, Silvio Micali. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, May 1987.
- [3] Christos H. Papadimitriou. *Złożoność obliczeniowa*. Klasyka Informatyki. Wydawnictwo Naukowo-Techniczne, 2002.
- [4] Adi Shamir. $\text{Ip} = \text{pspace}$. *Journal of the ACM*, 39(4):869–877, October 1992.
- [5] Salil Vadhan. Probabilistic proof systems - part i. 2000.