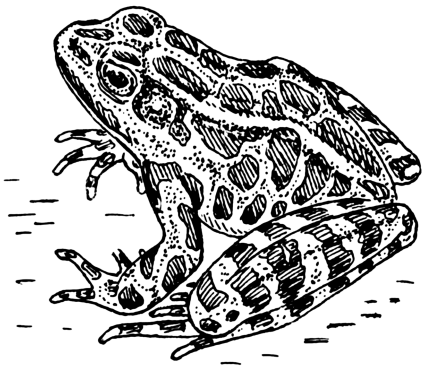


Egzamin licencjacki

Odpowiedzi na pytania

„Nigdy nie rozumiałem po co ludzie czytają treści z obiektówki”

Autorzy

Dziurawy Ponton
Załatany Ponton
Puchaty Pompon
Zatopiony Ponton
Tonący Ponton
V
Nijaki Ponton
Pontus Euxinus

Kraków
Anno Domini 2024

Spis treści

Licencja

Rozdział 1

Analiza Matematyczna

1.1 Szeregi liczbowe. Podstawowe kryteria zbieżności.

1.1.1 Szeregi liczbowe

Definicja 1.1.1. Nieskończonym ciągiem liczbowym (liczb rzeczywistych) nazywamy dowolną funkcję $a : \mathbb{N} \to \mathbb{R}$. Jej wartości $a(1), a(2), ..., a(n), ...$ oznaczamy przez $a_1, a_2, ..., a_n, ...$, sam ciąg oznaczamy przez $\{a_n\}$.

Definicja 1.1.2. Dla ciągu $\{a_n\}$ rozważamy ciąg $\{s_n\}$ *sum częściowych*:

$$\begin{aligned} s_1 &= a_1 \\ s_2 &= a_1 + a_2 \\ &\vdots \\ s_n &= \sum_{i=1}^{n} a_i \end{aligned}$$

Szereg liczbowy oznaczamy przez $a_1 + a_2 + ... + a_n + ...$ lub $\sum_{n=1}^{\infty} a_n$. Wyrazy ciągu $\{a_n\}$ w tym kontekście nazywamy *wyrazami szeregu*.

Definicja 1.1.3. Szereg $\sum_{n=1}^{\infty} a_n$ jest *zbieżny* jeśli jego ciąg sum cząstkowych $\{s_n\}$ jest zbieżny, czyli ma granicę skończoną s, wtedy liczbę s nazywamy *sumą szeregu nieskończonego*. Szereg, który nie jest zbieżny, nazywamy rozbieżnym.

[Rozpatruje się też szeregi $\sum_{n=0}^{\infty} a_n$ lub ogólnie szeregi postaci $\sum_{n=n_0}^{\infty} a_n$]

Przykład 1.1.1. Szereg $\sum_{n=0}^{\infty} \frac{1}{2^n}$ jest zbieżny, bo:

$$s_n = \sum_{k=0}^{n} \frac{1}{2^k} = 1 - \frac{1}{2^n}$$

Wobec tego mamy $\lim_{n\to\infty} s_n = 1$ [obrazek z dowodem graficznym nie wklejamy].

Definicja 1.1.4. Szereg $\sum_{n=1}^{\infty} a_n$ jest *bezwzględnie zbieżny* jeśli szereg $\sum_{n=1}^{\infty} |a_n|$ jest zbieżny.

1.1.2 Kryteria zbieżności

Twierdzenie 1.1.1 (Warunek konieczny zbieżności szeregu).

Twierdzenie 1.1.2 (Mnożenie przez stałą).

Twierdzenie 1.1.3 (Kryterium Cauchy'ego).

Twierdzenie 1.1.4 (O zbieżności bezwzględnej).

Twierdzenie 1.1.5 (O zbieżności szeregu geometrycznego).

Twierdzenie 1.1.6 (O zbieżności szeregu harmonicznego).

Twierdzenie 1.1.7 (Kryterium porównawcze).

Twierdzenie 1.1.8 (Kryterium d'Alamberta).

Twierdzenie 1.1.9 (Szereg Dirichleta).

1.2 Ciągłość funkcji w punkcie i na zbiorze. Warunki równoważne ciągłości. Własności funkcji ciągłych na zbiorach zwartych.

1.3 Pochodna funkcji jednej zmiennej rzeczywistej (definicja i interpretacja geometryczna). Zastosowanie rachunku różniczkowego do badania przebiegu zmienności funkcji jednej zmiennej.

1.4 Wzór Taylora i jego przykładowe zastosowania.

1.5 Całka Riemanna. Podstawowe metody całkowania funkcji jednej zmiennej rzeczywistej. Całkowanie przez części. Całkowanie przez podstawienie.

1.6 Pochodne cząstkowe. Różniczkowalność funkcji wielu zmiennych. Zależność między istnieniem pochodnych cząstkowych a różniczkowalnością.

1.7 Ekstrema funkcji wielu zmiennych. Efektywne metody wyznaczania ekstremów lokalnych funkcji wielu zmiennych.

1.8 Całkowanie funkcji wielu zmiennych. Twierdzenie Fubiniego. Twierdzenie o zamianie zmiennych.

Rozdział 2

Mφ

Wprowadzenie

W MFI jest dużo faktów, które nie są bezpośrednio wymagane w pytaniach, ale ten przedmiot buduje bardzo mocno na kolejnych definicjach. Ja (Puchaty Pompon) będę pomijał dowody rzeczy, które wydają się oczywiste (np. że para nieuporządkowana jest tylko jedna), ale jeśli ktoś uważa, że są one potrzebne, to zachęcam do dodania lub do kontaktu ze mną.

2.1 Definicje dodawania, mnożenia, potęgowania i odejmowania w operaciu o twierdzenie o definiowaniu przez indukcję. Własności tych działań

2.1.1 Definiowanie przez schemat rekursji prostej

Twierdzenie 2.1.1. Niech dane będą zbiory A, Z oraz funkcje $g : A \to Z, h : Z \times \mathbb{N} \times A \to Z$.

Wtedy istnieje dokładnie jedna funkcja $f : \mathbb{N} \times A \to Z$ dla której zachodzi

1. $f(0, a) = g(a)$ – warunek początkowy (bazowy)
2. $f(n', a) = h(f(n, a), n, a)$ – rekursor

Dowód. Rozważmy zbiór

$$P = \{n \in \mathbb{N} \mid \exists_{f_n} : f_n : n' \times A \to Z \text{ która spełnia zadane warunki } \}$$

Pokażemy, że P jest induktywny.

1. $0 \in P$

   $f_0 : \{0\} \times A \to Z$ definiujemy jako $f(0, a) = g(a)$ i nic innego nie możemy zrobić.

2. $n \in P \implies n' \in P$

   Mając $f_n : n' \times A \to Z$ konstruujemy $f_{n'} : n'' \times A \to Z$ w następujący sposób:

$$f_{n'}(k, a) = \begin{cases} f_n(k, a) & \text{gdy } k \leq n \\ h(f_n(n, a), n, a) & \text{gdy } k = n' \end{cases}$$

Ponieważ f_n spełniało oba warunki, a $f_{n'}$ zdefiniowane jest jak jest to widać że $f_{n'}$ również je spełnia.

W takim razie P jest induktywny, czyli $P = \mathbb{N}$.

Mamy też bardzo fajną własność, a mianowicie $f_n \subset f_{n'}$ (bo funkcje rozszerzamy przez dorzucenie do nich par (argument, wynik)).

Możemy zatem wziąć $f = \bigcup P = \bigcup_{n\in\mathbb{N}} f_n$ otrzymując funkcję zdefiniowaną na całym $\mathbb{N}\times A$. □

2.1.2 Operacje

Uzbrojeni w schemat rekursji prostej możemy zdefiniować znane nam działania na liczbach naturalnych.

Same zaś liczby oraz indukcję definiujemy w osobnym pytaniu – 2.6

2.1.2.1 Dodawanie

Definiujemy:

- $A = Z = \mathbb{N}$
- $g(n) = n$
- $h(p, n, a) = p'$

Dostajemy

$$\begin{cases} f(0, n) = g(n) = n \\ f(k', n) = h(f(k, n), k, n) = f(k, n)' \end{cases}$$

Dodawanie definiujemy jako $a + b = f(a, b)$

Własności f możemy zapisać nieco intuicyjniej:

$$\begin{cases} 0 + n = n \\ 1 + n = n' \\ (k + 1) + n = (k + n) + 1 \end{cases}$$

2.1.2.2 Mnożenie

Definiujemy:

- $A = Z = \mathbb{N}$
- $g(n) = 0$
- $h(p, n, a) = p + a$

Dostajemy

$$\begin{cases} f(0, n) = g(n) = 0 \\ f(k', n) = h(f(k, n), k, n) = f(k, n) + n \end{cases}$$

Mnożenie definiujemy jako $a \cdot b = f(a, b)$

Własności f możemy zapisać jako:

$$\begin{cases} 0 \cdot n = n \\ (k+1) \cdot n = k \cdot n + n \end{cases}$$

2.1.2.3 Potęgowanie

Definiujemy:

- $A = Z = \mathbb{N}$
- $g(n) = 1$
- $h(p, n, a) = p \cdot a$

Dostajemy

$$\begin{cases} f(0, n) = g(n) = 1 \\ f(k', n) = h(f(k, n), k, n) = f(k, n) \cdot n \end{cases}$$

Mnożenie definiujemy jako $b^a = f(a, b)$

Własności f możemy zapisać jako:

$$\begin{cases} n^0 = 1 \\ n^{k+1} = n^k \cdot n \end{cases}$$

2.1.3 Własności operacji

Zdefiniowaliśmy operacje o których wiemy (a posteriori) że mają pewne własności np. przemienność i łączność. Zwykle te własności przyjmuje się jako dane, tu zobaczymy że faktycznie wynikają one z konstrukcji liczb.

Większość z nich idzie bardzo podobnie (indukcyjnie) dlatego przedstawimy jedynie parę szkiców.

Lemat 2.1.1. $n + 0 = n$

Dowód. Może i wygląda to głupio, ale o dodawaniu wiemy z definicji tylko że $0 + n = n$, a jeszcze nie mamy przemienności.

Robimy indukcję po n

- Baza: $n = 0$, mamy $n + 0 = 0 + 0 = 0 = n$ z definicji
- Krok indukcyjny: $n' + 0 = (n + 0)' = n'$

□

Lemat 2.1.2. $a' + b = a + b'$

Dowód. Robimy indukcję po a

- Baza: $0' + b = (0 + b)' = b' = 0 + b'$
- Krok indukcyjny: $a'' + b \overset{\text{def.}+}{=} (a' + b)' \overset{\text{zał. ind}}{=} (a + b')' \overset{\text{def.}+}{=} a' + b'$

□

Twierdzenie 2.1.2 (Przemienność dodawania)**.** $a + b = b + a$

Dowód. Robimy indukcję po a

- Baza: $0 + b = b = b + 0$
- Krok indukcyjny: $a' + b = a + b' = b' + a = b + a'$

□

Twierdzenie 2.1.3 (Prawo skróceń)**.** $n + p = k + p \implies n = k$

Dowód. Indukcja po p

- Baza: $n + 0 = k + 0 \implies n = k$ tak po prostu
- Krok indukcyjny: $n + p' = k + p' \implies n' + p = k' + p \implies n' = k' \implies n = k$

□

2.2 Domykanie relacji ze względu na różne własności. Podaj przykłady własności na które istnieje i nie istnieje domknięcie

2.3 Zbiory przeliczalne i ich przykłady

2.4 Konstrukcja Cantora liczb rzeczywistych. Porządek na liczbach rzeczywistych. Twierdzenie o rozwinięciu liczby rzeczywistej w szereg

2.5 Iloczyn kartezjański i jego własności. Pojęcia relacji, złożenia, relacji odwrotnej, własności tych pojęć

Ta sekcja wprowadza (w miarę) formalnie definicje i własności, z którymi mieliśmy do czynienia wielokrotnie podczas studiów. Często stosowaliśmy je odruchowo. Dlatego sekcja jest długa, ale dowodów raczej nie trzeba się uczyć na pamięć — każdy bez problemu przeprowadzi odpowiedni dowód w razie potrzeby, a to jest tylko miejsce, które prezentuje je dla kompletności.

2.5.1 Iloczyn kartezjański — definicje

Aby zdefiniować iloczyn kartezjański, będziemy musieli skorzystać z paru aksjomatów:

- aksjomat pary nieuporządkowanej
- aksjomat zbioru potęgowego

Potrzebujemy też następującej definicji.

Definicja 2.5.1 (Para uporządkowana)**. Parą uporządkowaną** (a, b) nazywamy zbiór

$$\{\{a\}, \{a, b\}\}$$

Intuicyjnie, doubleton podaje elementy pary bez porządku, natomiast singleton ustala, który element jest pierwszy.

Warto przy tej okazji zastanowić się, gdzie żyją pary uporządkowane, których pierwszy element pochodzi z x, a drugi z y.

$$\begin{gathered} a \in x, b \in y \implies a, b \in x \cup y \\ \{a\}, \{a, b\} \in \mathcal{P}(x \cup y) \\ \{\{a\}, \{a, b\}\} \subseteq \mathcal{P}(x \cup y) \\ (a, b) = \{\{a\}, \{a, b\}\} \in \mathcal{P}(\mathcal{P}(x \cup y)) \end{gathered}$$

To wystarcza nam, aby zdefiniować iloczyn kartezjański.

Definicja 2.5.2 (Iloczyn kartezjański)**. Iloczynem kartezjańskim** zbiorów x, y nazywamy zbiór

$$x \times y = \{z \in \mathcal{P}(\mathcal{P}(x \cup y)) : \exists_{a \in x} \exists_{b \in y} (a, b) = z\}$$

Intuicyjnie, ze świata, w którym żyją pary uporządkowane, wybieramy te elementy, które faktycznie są parami uporządkowanymi (bo mogą tam też być inne „śmieci") oraz na pierwszej pozycji mają coś ze zbioru x, a na drugiej coś ze zbioru y.

2.5.2 Iloczyn kartezjański — własności

Iloczyn kartezjański ma parę przyjemnych własności przy działaniu na zbiorach w konkretnej postaci.

- $x \times \varnothing = \varnothing$

  Dowód. Inkluzja $x \times \varnothing \supseteq \varnothing$ jest trywialna. Wystarczy pokazać inkluzję w drugą stronę, tzn. $z \in x \times \varnothing \implies z \in \varnothing$.

  Jeśli założymy, że $z \in x \times \varnothing$, to z definicji $\exists_a \exists_b z = (a, b)$ oraz $a \in x \wedge b \in \varnothing$. Możemy zatem spokojnie postawić sobie implikację

  $$a \in x \wedge b \in \varnothing \implies z \in \varnothing$$

  ponieważ jej poprzednik jest fałszywy, a do fałszywego poprzednika możemy dołożyć dowolny następnik, jaki nam się podoba. □

- $x \times (y \cup z) = (x \times y) \cup (x \times z)$

  Dowód. Przejdziemy tutaj raz przez prawdziwe królestwo kwantyfikatorów, a w kolejnych podpunktach będziemy już dowodzić bardziej intuicyjnie.

  $$\begin{aligned} \xi \in x \times (y \cup z) &\iff \exists_{a \in x} \exists_{b \in y \cup z} \xi = (a, b) \\ &\iff \exists_{a \in x} \exists_b ((b \in y \vee b \in z) \wedge \xi = (a, b)) \\ &\iff \exists_{a \in x} \exists_b ((b \in y \wedge \xi = (a, b)) \vee (b \in z \wedge \xi = (a, b))) \\ &\iff \exists_{a \in x} (\exists_b (b \in y \wedge \xi = (a, b)) \vee \exists_b (b \in z \wedge \xi = (a, b))) \\ &\iff (\exists_{a \in x} \exists_b (b \in y \wedge \xi = (a, b))) \vee (\exists_{a \in x} \exists_b (b \in z \wedge \xi = (a, b))) \\ &\iff (\exists_{a \in x} \exists_{b \in y} \xi = (a, b)) \vee (\exists_{a \in x} \exists_{b \in z} \xi = (a, b)) \\ &\iff \xi \in (x \times y) \vee \xi \in (x \times z) \\ &\iff \xi \in (x \times y) \cup (x \times z) \end{aligned}$$

Skorzystaliśmy tutaj z rozdzielności konjunkcji względem alternatywy oraz z faktu, że

$$\exists_v(\varphi \vee \psi) \iff (\exists_v\varphi) \vee (\exists_v\psi)$$

□

- $x \times (y \cap z) = (x \times y) \cap (x \times z)$

  Dowód.

$$\begin{aligned}(a,b) \in x \times (y \cap z) &\iff a \in x \wedge b \in y \cap z \\ &\iff a \in x \wedge (b \in y \wedge b \in z) \\ &\iff (a \in x \wedge b \in y) \wedge (a \in x \wedge b \in z) \\ &\iff (a,b) \in (x \times y) \wedge (a,b) \in (x \times z) \\ &\iff (a,b) \in (x \times y) \cap (x \times z)\end{aligned}$$

  Skorzystaliśmy tu z rozdzielności konjunkcji względem niej samej. □

- $x \times (y \setminus z) = (x \times y) \setminus (x \times z)$

  Dowód.

$$\begin{aligned}(a,b) \in x \times (y \setminus z) &\iff a \in x \wedge b \in (y \setminus z) \\ &\iff a \in x \wedge (b \in y \wedge b \notin z) \\ &\iff (a \in x \wedge b \in y) \wedge (a \in x \wedge b \notin z) \\ &\iff (a,b) \in (x \times y) \wedge (a,b) \notin (x \times z) \\ &\iff (a,b) \in (x \times y) \setminus (x \times z)\end{aligned}$$

  Ponownie skorzystaliśmy tu z rozdzielności konjunkcji względem niej samej. □

Ponadto, iloczyn kartezjański jest też *monotoniczny* ze względu na każdą współrzędną. Zgadza się to z intuicją — jeśli na jednej ze współrzędnych mamy „więcej" możliwości wyboru elementu, to również zbiór możliwych par będzie „większy".

- $x \subseteq y \implies (x \times z) \subseteq (y \times z)$

  Dowód. Załóżmy, że $(a,b) \in (x \times z)$. Oznacza to, że $a \in x$ i $b \in z$. Skoro $a \in x$ oraz $x \subseteq y$, to $a \in y$. Skoro zatem $a \in y$ i $b \in z$, to $(a,b) \in (y \times z)$. To dowodzi postulowanej inkluzji. □

- $x \subseteq y \implies (z \times x) \subseteq (z \times y)$

  Dowód. Analogiczny jak dla pierwszej współrzędnej. □

2.5.3 Relacje — definicje

Definicja 2.5.3 (Relacja)**.** Dowolny podzbiór iloczynu kartezjańskiego $R \subseteq x \times y$ nazywamy **relacją**.

Na relacjach możemy zdefiniować intuicyjne operacje. W poniższych definicjach przyjmujemy, że mamy już dane jakieś relacje $R \subseteq A \times B$ i $S \subseteq B \times C$.

Definicja 2.5.4 (Złożenie relacji)**.**

$$S \circ R \coloneqq \{(x,z) \in A \times C : \exists_{y \in B}(x,y) \in R \wedge (y,z) \in S\}$$

Uwaga na kolejność!

Definicja 2.5.5 (Relacja odwrotna)**.**

$$R^{-1} \coloneqq \{(y,x) \in B \times A : (x,y) \in R\}$$

Definicja 2.5.6 (Projekcje)**.**

$$R_{\mathrm{L}} \coloneqq \{x \in A : \exists_{y \in B}(x,y) \in R\}$$
$$R_{\mathrm{P}} \coloneqq \{y \in B : \exists_{x \in A}(x,y) \in R\}$$

2.5.4 Relacje — własności

Relacje mają bardzo dużo różnych własności. Większość z nich nie jest szczególnie ciekawa, natomiast trafiają się perełki, które bardzo zapadają w pamięć, ponieważ da się je przeczytać jak słowa. Tak, tak, mam na myśli słynne „rusot równa się rotusot". Do tego też dotrzemy.

Pierwsza grupa własności. Przyjmijmy, że mamy następujące trzy relacje:

$$R \subseteq A \times B$$
$$S \subseteq B \times C$$
$$T \subseteq C \times D$$

Wtedy

- $T \circ (S \circ R) = (T \circ S) \circ R$ (składanie relacji jest łączne — własność znana także jako „to-sor równa się tos-or")

  Dowód. „Typ" się zgadza, ponieważ obie strony są podzbiorami $A \times D$.

  $$\begin{aligned}
  (a,d) \in T \circ (S \circ R) &\iff a \in A \wedge d \in D \wedge \exists_{c \in C}((a,c) \in S \circ R \wedge (c,d) \in T) \\
  &\iff a \in A \wedge d \in D \\
  &\qquad \wedge \exists_{c \in C}((\exists_{b \in B}((a,b) \in R \wedge (b,c) \in S)) \wedge (c,d) \in T) \\
  &\iff a \in A \wedge d \in D \\
  &\qquad \wedge \exists_{c \in C} \exists_{b \in B}((a,b) \in R \wedge (b,c) \in S \wedge (c,d) \in T) \\
  &\iff a \in A \wedge d \in D \\
  &\qquad \wedge \exists_{b \in B}((a,b) \in R \wedge (\exists_{c \in C}((b,c) \in S \wedge (c,d) \in T))) \\
  &\iff a \in A \wedge d \in D \wedge \exists_{b \in B}((a,b) \in R \wedge (b,d) \in T \circ S) \\
  &\iff (a,d) \in (T \circ S) \circ R
  \end{aligned}$$

  □

- $(S \circ R)^{-1} = R^{-1} \circ S^{-1}$

  Dowód. Zauważamy najpierw, że zgadza się „typ", ponieważ obie strony są podzbiorami $C \times A$.

  $$\begin{aligned}
  (c,a) \in (S \circ R)^{-1} &\iff (a,c) \in (S \circ R) \\
  &\iff a \in A \wedge c \in C \wedge \exists_{b \in B}((a,b) \in R \wedge (b,c) \in S) \\
  &\iff a \in A \wedge c \in C \wedge \exists_{b \in B}\big((b,a) \in R^{-1} \wedge (c,b) \in S^{-1}\big) \\
  &\iff (c,a) \in R^{-1} \circ S^{-1}
  \end{aligned}$$

  □

- $R \subseteq R_{\mathrm{L}} \times R_{\mathrm{P}}$

  Dowód.

$$\begin{aligned}(a,b) \in R &\implies a \in A \wedge b \in B \wedge (\exists_{y \in B}(a,y) \in R) \wedge (\exists_{x \in A}(x,b) \in R) \\ &\iff (a \in A \wedge \exists_{y \in B}(a,y) \in R) \wedge (b \in B \wedge \exists_{x \in A}(x,b) \in R) \\ &\iff a \in R_{\mathrm{L}} \wedge b \in R_{\mathrm{P}} \\ &\iff (a,b) \in R_{\mathrm{L}} \times R_{\mathrm{P}}\end{aligned}$$

  Pierwsze przejście jest tylko implikacją. Oczywiście jeśli $(a,b) \in R$, to możemy powiedzieć, że zarówno a, jak i b mają parę w R. Jednak z samego faktu, że a i b mają parę w R, nie wynika jeszcze, że a i b *są* parą w R. Dlatego w ogólnym przypadku mamy tu zaledwie inkluzję, a nie równość, co łatwo zobaczyć na przykładzie:

$$\begin{aligned}A &= \{a_0, a_1\} \\ B &= \{b_0, b_0\} \\ R &= \{(a_0,b_0),(a_1,b_1)\} \\ R_{\mathrm{L}} &= (a_0,a_1) \\ R_{\mathrm{P}} &= (b_0,b_1) \\ R_{\mathrm{L}} \times R_{\mathrm{P}} &= \{(a_0,b_0),(a_0,b_1),(a_1,b_0),(a_1,b_1)\} \\ R &\subsetneq R_{\mathrm{L}} \times R_{\mathrm{P}}\end{aligned}$$

  □

- $(S \circ R)_{\mathrm{L}} \subseteq R_{\mathrm{L}}$

  Dowód. Zwróćmy uwagę, że $S \circ R \subseteq A \times C$. Mamy

$$\begin{aligned}a \in (S \circ R)_{\mathrm{L}} &\iff a \in A \wedge \exists_{c \in C}(a,c) \in S \circ R \\ &\iff a \in A \wedge \exists_{c \in C} \exists_{b \in B}((a,b) \in R \wedge (b,c) \in S) \\ &\implies a \in A \wedge \exists_{c \in C}((\exists_{b \in B}(a,b) \in R) \wedge (\exists_{b \in B}(b,c) \in S)) \\ &\implies a \in A \wedge \exists_{c \in C}(\exists_{b \in B}(a,b) \in R) \\ &\implies a \in A \wedge \exists_{b \in B}(a,b) \in R \\ &\iff a \in R_{\mathrm{L}}\end{aligned}$$

  Warto zauważyć, że trzy z powyższych przejść są tylko implikacjami (bo zapominamy na przykład o pewnych członach konjunkcji) i nie jest to przypadek. Istotnie, zachodzi tylko inkluzja w podaną stronę. Aby się o tym przekonać możemy przyjąć

$$\begin{aligned}A &= \{a\} \\ B &= \{b\} \\ C &= \{c\} \\ R &= \{(a,b)\} \subseteq A \times B \\ S &= \varnothing \subseteq B \times C\end{aligned}$$

  Wtedy $S \circ R = \varnothing$, zatem również $(S \circ R)_{\mathrm{L}} = \varnothing$. Natomiast $R_{\mathrm{L}} = \{a\}$, więc istotnie zachodzi dowiedziona inkluzja (byłby przypał, gdyby było inaczej), ale nie ma równości. □

- $(S \circ R)_{\mathrm{P}} \subseteq S_{\mathrm{P}}$

Dowód. Analogiczny jak dla lewej projekcji. □

- $(R^{-1})_{\mathrm{L}} = R_{\mathrm{P}}$

  Dowód.

$$\begin{aligned} b \in \left(R^{-1}\right)_{\mathrm{L}} &\iff b \in B \wedge \exists_{a \in A}(b,a) \in R^{-1} \\ &\iff b \in B \wedge \exists_{a \in A}(a,b) \in R \\ &\iff b \in R_{\mathrm{P}} \end{aligned}$$

□

Druga grupa własności. Przyjmijmy, że mamy następujące trzy relacje:

$$\begin{aligned} R &\subseteq B \times C \\ S &\subseteq B \times C \\ T &\subseteq A \times B \end{aligned}$$

Wtedy

- $(R \cup S)^{-1} = R^{-1} \cup S^{-1}$

  Dowód.

$$\begin{aligned} (c,b) \in (R \cup S)^{-1} &\iff (b,c) \in R \cup S \\ &\iff (b,c) \in R \vee (b,c) \in S \\ &\iff (c,b) \in R^{-1} \vee (c,b) \in S^{-1} \\ &\iff (c,b) \in R^{-1} \cup S^{-1} \end{aligned}$$

□

- $(R \cap S)^{-1} = R^{-1} \cap S^{-1}$

  Dowód.

$$\begin{aligned} (c,b) \in (R \cap S)^{-1} &\iff (b,c) \in R \cap S \\ &\iff (b,c) \in R \wedge (b,c) \in S \\ &\iff (c,b) \in R^{-1} \wedge (c,b) \in S^{-1} \\ &\iff (c,b) \in R^{-1} \cap S^{-1} \end{aligned}$$

□

- $(R^{-1})^{-1} = R$

  Dowód. Zbyt skomplikowany, zdecydowanie wykracza poza materiał przedmiotu. Dlatego pomijamy. □

- $(R \cup S) \circ T = (R \circ T) \cup (S \circ T)$ (rozdzielność złożenia względem sumy — „rusot równa się rotusot")

  Dowód.

$$\begin{aligned} (a,c) \in (R \cup S) \circ T &\iff \exists_{b \in B}((a,b) \in T \wedge (b,c) \in R \cup S) \\ &\iff \exists_{b \in B}((a,b) \in T \wedge ((b,c) \in R \vee (b,c) \in S)) \\ &\iff \exists_{b \in B}(((a,b) \in T \wedge (b,c) \in R) \vee ((a,b) \in T \wedge (b,c) \in S)) \\ &\iff (\exists_{b \in B}((a,b) \in T \wedge (b,c) \in R)) \\ &\qquad \vee (\exists_{b \in B}((a,b) \in T \wedge (b,c) \in S)) \\ &\iff (a,c) \in (R \circ T) \vee (a,c) \in (S \circ T) \\ &\iff (a,c) \in (R \circ T) \cup (S \circ T) \end{aligned}$$

Skorzystaliśmy tutaj z rozdzielności konjunkcji względem alternatywy oraz rozbiliśmy kwantyfikator egzystencjonalny na alternatywie. □

- $(R \cap S) \circ T \subseteq (R \circ T) \cap (S \circ T)$

  Dowód.

$$\begin{aligned}
(a,c) \in (R \cap S) \circ T &\iff \exists_{b \in B}((a,b) \in T \wedge (b,c) \in R \cap S) \\
&\iff \exists_{b \in B}((a,b) \in T \wedge ((b,c) \in R \wedge (b,c) \in S)) \\
&\iff \exists_{b \in B}(((a,b) \in T \wedge (b,c) \in R) \wedge ((a,b) \in T \wedge (b,c) \in S)) \\
&\implies (\exists_{b \in B}((a,b) \in T \wedge (b,c) \in R)) \\
&\qquad \wedge (\exists_{b \in B}((a,b) \in T \wedge (b,c) \in S)) \\
&\iff (a,c) \in (R \circ T) \wedge (a,c) \in (S \circ T) \\
&\iff (a,c) \in (R \circ T) \cap (S \circ T)
\end{aligned}$$

  Skorzystaliśmy tutaj z rozdzielności konjunkcji względem niej samej oraz rozbiliśmy kwantyfikator egzystencjonalny — tym razem na konjunkcji, więc przejście działa tylko w jedną stronę. □

2.6 Konstrukcja liczb naturalnych von Neumanna, twierdzenie o indukcji. Własności liczb naturalnych

2.7 Zasada minimum. Zasada maksimum. Twierdzenie o definiowaniu przez indukcję

2.8 Relacje równoważności i podziały zbiorów. Relacja równoważności jako środek do definiowania pojęć abstrakcyjnych

Tymczasowy tekst żeby LaTeXsię ogarnął, bo inaczej wszystko renderuje się na jednej stronie, z jakiegoś powodu

2.9 Twierdzenie Cantora-Bernsteina. Twierdzenie Cantora. Czy istnieje zbiór wszystkich zbiorów? Odpowiedź uzasadnij

2.9.1 Twierdzenie Cantora-Bernsteina

Twierdzenie 2.9.1 (Cantor-Bernstein)**.** Dla dowolnych zbiorów A, B zachodzi implikacja:

$$(A \leq_{\mathrm{m}} B \wedge B \leq_{\mathrm{m}} A) \implies A \sim_{\mathrm{m}} B$$

Dowód. Z założeń mamy dane dwie iniekcje – $f : A \to B$ oraz $g : B \to A$.

Korzystając z lematu Banacha (2.15.1) możemy podzielić zbiory A i B:

$$A = A_1 \cup A_2, A_1 \cap A_2 = \varnothing$$

$$B = B_1 \cup B_2, B_1 \cap B_2 = \varnothing$$

w taki sposób, że

$$\overrightarrow{f}(A_1) = B_1, \overrightarrow{g}(B_2) = A_2$$

Ostatnia własność mówi nam, że zawężenia $f_{|A_1} : A_1 \to B_1$ i $g_{|B_2} : B_2 \to A_2$ są suriekcjami. Ponieważ jednak z założeń f, g są iniekcjami, to ich zawężenia są eleganckimi bijekcjami.

Dzięki nim możemy zdefiniować $h : A \to B$

$$h(x) = \begin{cases} f_{|A_1}(x) & \text{gdy } x \in A_1 \\ (g_{|B_2})^{-1}(x) & \text{gdy } x \in A_2 \end{cases}$$

które jest bijekcją, o czym świadczy odwrotność

$$h^{-1}(x) = \begin{cases} (f_{|A_1})^{-1}(x) & \text{gdy } x \in B_1 \\ g_{|B_2}(x) & \text{gdy } x \in B_2 \end{cases}$$

Mamy zatem bijekcję między A i B, a zatem $A \sim_{\text{m}} B$

□

2.9.2 Twierdzenie Cantora

Twierdzenie 2.9.2 (Cantor)**.** Dla dowolnego zbioru A

$$A <_{\text{m}} \mathcal{P}(A)$$

Dowód. Oczywiście $A \leq_{\text{m}} \mathcal{P}(A)$ bo mamy iniekcję $f : A \ni a \mapsto \{a\} \in \mathcal{P}(A)$

Załóżmy teraz nie wprost, że istnieje bijekcja $f : A \to \mathcal{P}(A)$ i zdefiniujmy

$$R = \{x \in A \mid x \notin f(x)\}$$

R zawiera jedynie elementy z A, więc niewątpliwie $R \in \mathcal{P}(A)$, a ponieważ f jest bijekcją to $\exists x_0 : f(x_0) = R$.

Mamy teraz dwie możliwości:

- Jeśli $x_0 \in R = f(x_0)$ to $x_0 \notin f(x_0) = R$
- Jeśli $x_0 \notin R$ to $x_0 \in f(x_0) = R$

W obu przypadkach mamy sprzeczność, co dowodzi, że nie może istnieć bijekcja między A i $\mathcal{P}(A)$ □

2.9.3 Zbiór wszystkich zbiorów

Twierdzenie 2.9.3. Nie istnieje zbiór wszystkich zbiorów

Dowód. Nie wprost niech istnieje x taki, że $\forall y : y \in x$.

1. Po pierwsze mamy sprzeczność z aksjomatem regularności – $x \cap \{x\} = x$
2. Po drugie mamy sprzeczność z tw. Cantora.

   Z jednej strony $\mathcal{P}(x) \subseteq x$ więc $\mathcal{P}(x) \leq_{\text{m}} x$.

   Z drugiej strony $x \leq_{\text{m}} \mathcal{P}(x)$, a więc $x \sim_{\text{m}} \mathcal{P}(x)$, co istotnie jest sprzeczne z tw. Cantora.

□

2.10 Ciągłość i gęstość porządku. Zbiór liczb wymiernych a zbiór liczb rzeczywistych

2.11 Lemat Kuratowskiego-Zorna i przykłady jego zastosowania

2.12 Konstrukcja liczb całkowitych. Działania na liczbach całkowitych. Konstrukcja liczb wymiernych i działania na nich

2.13 Przykłady zbiorów nieprzeliczalnych

2.14 Aksjomatyczne ujęcie teorii mnogości. Aksjomat wyboru.

2.15 Twierdzenie Knastera-Tarskiego (dla zbiorów). Lemat Banacha

2.15.1 Potrzebne pojęcia

Aby zrozumieć, o czym w ogóle mówią twierdzenia z tego pytania, musimy zdefiniować kilka pojęć.

Definicja 2.15.1 (Obraz i przeciwobraz)**.** Niech $f\colon X \to Y$ oraz $A \subseteq X$ i $B \subseteq Y$.

Obrazem zbioru A względem f nazywamy

$$\overrightarrow{f}(A) \coloneqq \{y \in Y : \exists_{x \in A} f(x) = y\}$$

czyli po prostu zbiór wartości, które f przyjmuje na argumentach pochodzących z A.

Przeciwobrazem zbioru B względem f nazywamy

$$\overrightarrow{f}^{-1}(B) \coloneqq \{x \in X : \exists_{y \in B} f(x) = y\}$$

czyli argumenty, które f przenosi na wartości pochodzące z B.

Możemy traktować je jak funkcje

$$\overrightarrow{f}\colon \mathcal{P}(X) \to \mathcal{P}(Y)$$
$$\overrightarrow{f}^{-1}\colon \mathcal{P}(Y) \to \mathcal{P}(X)$$

Jeśli chcemy być bardzo fancy i niezrozumiali, to przeciwobraz singletonu nazywamy **włóknem** (lub **poziomicą** lub **warstwicą**, jak podaje Wikipedia).

Definicja 2.15.2 (Monotoniczność)**.** Mówimy, że $f\colon \mathcal{P}(X) \to \mathcal{P}(Y)$ jest **monotoniczna ze względu na inkluzję, gdy**

$$x \subseteq y \implies f(x) \subseteq f(y)$$

„Większy" argument przechodzi na „większą" wartość.

Zauważmy, że funkcje obrazu i przeciwobrazu są monotoniczne.

Definicja 2.15.3 (Punkt stały)**.** Jeśli dla funkcji $f\colon \mathcal{P}(X) \to \mathcal{P}(X)$ i zbioru $x_0 \subseteq X$ zachodzi

$$f(x_0) = x_0$$

to x_0 nazywamy **punktem stałym** funkcji f.

Jest **najmniejszym** punktem stałym, gdy

$$f(y) = y \implies x_0 \subseteq y$$

Jest **największym** punktem stałym, gdy

$$f(y) = y \implies y \subseteq x_0$$

2.15.2 Właściwe twierdzenia

Twierdzenie 2.15.1 (Knaster-Tarski)**.** Każde monotoniczne odwzorowanie $f\colon \mathcal{P}(X) \to \mathcal{P}(X)$ posiada najmniejszy punkt stały i największy punkt stały.

Dowód. Tak naprawdę dwukrotnie przeprowadzimy bardzo sprytne i na swój sposób analogiczne rozumowania.

Najmniejszy punkt stały. Zdefiniujmy

$$\alpha = \{z \in \mathcal{P}(X) : f(z) \subseteq z\}$$

Zauważmy, że $\alpha \neq \varnothing$, ponieważ $X \in \alpha$.

Weźmy $x_0 = \bigcap \alpha$. Z własności przecięcia mamy

$$\forall_{z\in\alpha} x_0 \subseteq z$$

i dalej z monotoniczności

$$\forall_{z\in\alpha} f(x_0) \subseteq f(z)$$

Skoro jednak rozważamy $z \in \alpha$, to z definicji zbioru α mamy $f(z) \subseteq z$. Czyli tak naprawdę otrzymujemy

$$\forall_{z\in\alpha} f(x_0) \subseteq z$$

Skoro $f(x_0)$ zawiera się w każdym elemencie zbioru α, to mamy też $f(x_0) \subseteq \bigcap \alpha = x_0$.

A skoro $f(x_0) \subseteq x_0$, to z monotoniczności mamy

$$f(f(x_0)) \subseteq f(x_0)$$

co oznacza, że $f(x_0) \in \alpha$. No a jeśli $f(x_0) \in \alpha$, to

$$f(x_0) \supseteq \bigcap \alpha = x_0$$

Mamy zatem zarówno $f(x_0) \subseteq x_0$, jak i $f(x_0) \supseteq x_0$. To oznacza po prostu, że $f(x_0) = x_0$, czyli x_0 jest punktem stałym! Zauważmy, że jest najmniejszym: jeśli mamy jakiś punkt stały z, to $f(z) = z$, czyli w szczególności $f(z) \subseteq z$, czyli $z \in \alpha$. A wiemy, że wtedy

$$x_0 = \bigcap \alpha \subseteq z$$

Czyli rzeczywiście x_0 zawiera się w każdym innym punkcie stałym. Sprytne, nie? Aż chce się stworzyć sequel tego dowodu!

Największy punkt stały. Zdefiniujmy

$$\beta = \{z \in \mathcal{P}(X) : z \subseteq f(z)\}$$

Oczywiście $\varnothing \in \beta$, więc $\beta \neq \varnothing$. Widać już, dokąd z tym zmierzamy, prawda? Tak jest, bierzemy $y_0 = \bigcup \beta$. Z własności sumy zachodzi

$$\forall_{z \in \beta} z \subseteq y_0$$

i dalej z monotoniczności

$$\forall_{z \in \beta} f(z) \subseteq f(y_0)$$

Przytaczając definicję zbioru β, możemy z tego wyciągnąć po prostu

$$\forall_{z \in \beta} z \subseteq f(y_0)$$

Innymi słowy, wszystkie elementy zbioru β mieszczą się pod $f(y_0)$. Czyli mieści się tam również ich suma:

$$y_0 = \bigcup \beta \subseteq f(y_0)$$

Monotoniczność pozwala nam zamienić $y_0 \subseteq f(y_0)$ na $f(y_0) \subseteq f(f(y_0))$, co pokazuje, że $f(y_0) \in \beta$. No, a skoro $f(y_0) \in \beta$, to

$$f(y_0) \subseteq \bigcup \beta = y_0$$

Skoro mamy obydwie inkluzje — $y_0 \subseteq f(y_0)$ i $f(y_0) \subseteq y_0$ — to y_0 jest punktem stałym.

Czy największym? Tak! Jeśli $f(z) = z$, to w szczególności $z \subseteq f(z)$, czyli $z \in \beta$, czyli $z \subseteq \bigcup \beta = y_0$.

□

Lemat 2.15.1 (Banach)**.** Niech $f\colon X \to Y$ oraz $g\colon Y \implies X$.

Istnieją:

- rozkład $\{A_1, A_2\}$ zbioru X $(A_1 \cup A_2 = X \wedge A_1 \cap A_2 = \varnothing)$
- rozkład $\{B_1, B_2\}$ zbioru Y $(B_1 \cup B_2 = Y \wedge B_1 \cap B_2 = \varnothing)$

takie, że

- $\overrightarrow{f}(A_1) = B_1$
- $\overrightarrow{g}(B_2) = A_2$

Uwaga! Dopuszczamy tu trywialne rozkłady, to znaczy niektóre ze zbiorów A_1, A_2, B_1, B_2 mogą być puste.

Dowód. Zdefiniujmy następującą funkcję:

$$\varphi\colon \mathcal{P}(X) \to \mathcal{P}(X)$$
$$\varphi(x) = X \setminus \overrightarrow{g}\left(Y \setminus \overrightarrow{f}(x)\right)$$

Intuicyjnie, funkcja φ

1. zaczyna w zbiorze $x \subseteq X$

2. przechodzi z niego funkcją f do Y
3. wyrzuca z Y elementy, na które przeszła
4. z elementów pozostałych w zbiorze Y przechodzi funkcją g do X
5. wyrzuca z X elementy, na które przeszła

Zauważmy najpierw, że φ jest monotoniczna:

$$\begin{aligned} A &\subseteq B \\ \overrightarrow{f}(A) &\subseteq \overrightarrow{f}(B) \\ Y \setminus \overrightarrow{f}(A) &\supseteq Y \setminus \overrightarrow{f}(B) \\ \overrightarrow{g}\left(Y \setminus \overrightarrow{f}(A)\right) &\supseteq \overrightarrow{g}\left(Y \setminus \overrightarrow{f}(B)\right) \\ X \setminus \overrightarrow{g}\left(Y \setminus \overrightarrow{f}(A)\right) &\subseteq X \setminus \overrightarrow{g}\left(Y \setminus \overrightarrow{f}(B)\right) \end{aligned}$$

Na mocy twierdzenia Knastera-Tarskiego, funkcja φ posiada punkt stały. Nazwijmy go A_1. Resztę ustalmy następująco:

$$\begin{aligned} B_1 &\coloneqq \overrightarrow{f}(A_1) \\ B_2 &\coloneqq Y \setminus B_1 \\ A_2 &\coloneqq X \setminus A_1 \end{aligned}$$

Z definicji mamy zatem spełnione wymaganie, że $\overrightarrow{f}(A_1) = B_1$. Zauważamy też

$$\begin{aligned} \overrightarrow{g}(B_2) &= \overrightarrow{g}(Y \setminus B_1) && \text{z definicji } B_2 \\ &= \overrightarrow{g}\left(Y \setminus \overrightarrow{f}(A_1)\right) && \text{z definicji } B_1 \\ &= X \setminus \left(X \setminus \overrightarrow{g}\left(Y \setminus \overrightarrow{f}(A_1)\right)\right) && \text{jesteśmy w uniwersum } X \\ &= X \setminus \varphi(A_1) && \text{z definicji } \varphi \\ &= X \setminus A_1 && A_1 \text{ jest punktem stałym} \\ &= A_2 && \text{z definicji } A_2 \end{aligned}$$

Zatem również drugie wymaganie $\overrightarrow{g}(B_2) = A_2$ jest spełnione. □

2.16 Równoliczność zbiorów na przykładach $\left(A^B\right)^C \sim_{\mathrm{m}} A^{B\times C}$ oraz $(A \times B)^C \sim_{\mathrm{m}} A^C \times B^C$

Definicja 2.16.1 (Równoliczność i podobne)**.** Mówimy, że zbiory A i B są **równoliczne**, gdy istnieje bijekcja $f\colon A \to B$. Oznaczamy to jako $A \sim_{\mathrm{m}} B$.

Gdy istnieje injekcja $f\colon A \to B$, to oznaczamy $A \leq_{\mathrm{m}} B$.

Gdy istnieje injekcja i nieprawda, że $A \sim_{\mathrm{m}} B$, to oznaczamy $A <_{\mathrm{m}} B$.

Zauważmy, że relacja $\sim_{\mathrm{m}}$ jest relacją równoważności, ponieważ jest

- zwrotna — każdy zbiór jest w bijekcji z samym sobą (identyczność)
- symetryczna — bijekcję da się odwrócić

- przechodnia — złożenie bijekcji jest bijekcją.

Twierdzenie 2.16.1. Dla dowolnych zbiorów A, B, C zachodzi

$$\left(A^B\right)^C \sim_{\mathrm{m}} A^{B\times C}$$

Dowód. Z definicji — konstruujemy bijekcję

$$\alpha\colon \left(A^B\right)^C \to A^{B\times C}$$

Funkcja α przyjmuje funkcję $C \to A^B$ i zwraca funkcję $B\times C \to A$. Zdefiniujmy α następująco:

$$\begin{aligned}\alpha(f) &\coloneqq (\text{taka funkcja } \gamma, \text{ że } \gamma(b,c) = (f(c))(b))\\ &\coloneqq \{((b,c),(f(c))(b)) : (b,c) \in B\times C\}\end{aligned}$$

Nadużywając trochę notacji, możemy to bardziej intuicyjnie zapisać jako

$$(\alpha(f))(b,c) \coloneqq (f(c))(b)$$

Musimy pokazać, że α bijekcją.

Injektywność. Weźmy różne $f_1, f_2 \in \left(A^B\right)^C$ i pokażmy, że $\alpha(f_1) \neq \alpha(f_2)$.

Co to znaczy, że f_1, f_2 są różne? To są funkcje $C \to A^B$, więc gdy są różne, to na jakimś argumencie $c_0 \in C$ przyjmują różne wartości:

$$f_1(c_0) \neq f_2(c_0)$$

Ale to też są funkcje, tym razem $B \to A$. Czyli skoro są różne, to na pewnym argumencie $b_0 \in B$ przyjmują różne wartości:

$$(f_1(c_0))(b_0) \neq (f_2(c_0))(b_0)$$

Możemy to zapisać za pomocą α jako

$$(\alpha(f_1))(b_0,c_0) \neq (\alpha(f_2))(b_0,c_0)$$

Oznacza to, że funkcje $\alpha(f_1), \alpha(f_2)\colon B\times C \to \alpha$ przyjmują różne wartości na argumencie $(b_0,c_0) \in B\times C$. A zatem są to różne funkcje $\alpha(f_1) \neq \alpha(f_2)$, co właśnie chcieliśmy udowodnić.

Surjektywność. Weźmy $g \in A^{B\times C}$. Musimy pokazać, że istnieje $f \in \left(A^B\right)^C$ takie, że

$$\alpha(f) = g$$

Niewątpliwie, w dziedzinie funkcji α znajduje się w szczególności $f_0\colon C \to A^B$ zdefiniowane następująco

$$\begin{aligned}f_0(c) &\coloneqq (\text{taka funkcja } \gamma, \text{ że } \gamma(b) = g(b,c))\\ &\coloneqq \{(b, g(b,c)) : b \in B\}\end{aligned}$$

co można, ponownie z drobnym nadużyciem notacji można zapisać jako

$$(f_0(c))(b) \coloneqq g(b,c)$$

Zobaczmy, że dla dowolnej pary $(b_0,c_0) \in B\times C$ zachodzi

$$\begin{aligned}(\alpha(f_0))(b_0,c_0) &= (f_0(c_0))(b_0) && \text{z definicji } \alpha\\ &= g(b_0,c_0) && \text{z definicji } f_0\end{aligned}$$

Zatem istotnie $\alpha(f_0) = g$ jako funkcje.

□

Twierdzenie 2.16.2. Dla dowolnych zbiorów A, B, C zachodzi

$$(A \times B)^C \sim_{\mathrm{m}} A^C \times B^C$$

Dowód. Korzystamy z przemienności i konstruujemy bijekcję

$$\alpha \colon A^C \times B^C \to (A \times B)^C$$

Funkcja α przyjmuje parę funkcji $(C \to A, C \to B)$ i zwraca funkcję $C \to A \times B$. Definiujemy ją następująco:

$$\begin{aligned} \alpha(f, g) &\coloneqq (\text{taka funkcja } \eta, \text{ że } \eta(c) = (f(c), g(c))) \\ &\coloneqq \{(c, (f(c), g(c))) : c \in C\} \end{aligned}$$

Inaczej:

$$(\alpha(f, g))(c) = (f(c), g(c))$$

Musimy pokazać, że α jest bijekcją.

Injektywność. Weźmy różne pary $(f_1, g_1), (f_2, g_2) \in A^C \times B^C$. Skoro są to różne pary funkcji, to istnieje takie $c_0 \in C$, że $f_1(c_0) \neq f_2(c_0)$ lub istnieje takie $c_0' \in C$, że $g_1(c_0') \neq g_2(c_0')$. Bez straty ogólności, przyjmijmy pierwszą opcję. Widzimy, że

$$(\alpha(f_1, g_1))(c_0) = (f_1(c_0), g_1(c_0)) \neq (f_2(c_0), g_2(c_0)) = (\alpha(f_2, g_2))(c_0)$$

Oznacza to, że $\alpha(f_1, g_1)$ i $\alpha(f_2, g_2)$ są różnymi funkcjami — tak, jak chcieliśmy.

Surjektywność. Weźmy $h \in (A \times B)^C$. Musimy pokazać, że istnieje para $(f_0, g_0) \in A^C \times B^C$ taka, że

$$\alpha(f_0, g_0) = h$$

Weźmy

$$\begin{gathered} f_0 \colon C \to A \\ f_0(c) \coloneqq \text{lewy element pary } h(c) \end{gathered}$$

oraz

$$\begin{gathered} g_0 \colon C \to B \\ g_0(c) \coloneqq \text{prawy element pary } h(c) \end{gathered}$$

Teraz dla dowolnego $c_0 \in C$ mamy

$$(\alpha(f_0, g_0))(c_0) = (f_0(c_0), g_0(c_0)) = (\text{lewy z } h(c_0), \text{prawy z } h(c_0)) = h(c_0)$$

co dowodzi, że $\alpha(f_0, g_0) = h$ jako funkcje.

Jako bonus możemy zastanowić się, jak teoriomnogościowo wyciągać lewy i prawy element pary uporządkowanej. Wykorzystamy definicję 2.5.1.

- $\bigcup\bigcap p$ jest lewym elementem pary p

$$p = (a, b) = \{\{a\}, \{a, b\}\} \xrightarrow{\bigcap} \{a\} \xrightarrow{\bigcup} a$$

- $\bigcup(\bigcup p \setminus \bigcap p)$ jest prawym elementem pary p

$$p = (a, b) = \{\{a\}, \{a, b\}\} \xrightarrow{\bigcap} \{a\} \xrightarrow{\bigcup\setminus(\cdot)} \{a, b\} \setminus \{a\} = \{b\} \xrightarrow{\bigcup} b$$

□

2.17 Zasada indukcji pozaskończonej a dobry porządek

2.18 Liczby porządkowe von Neumanna i ich własności. Antynomia Burali-Forti

Rozdział 3

MAI

3.1 Grupy, ich przykłady i zastosowania

3.1.1 Definicja

Definicja 3.1.1. Grupą nazywamy tuplę $(G, \cdot)$, gdzie:

1. G to zbiór elementów grupy;
2. $\cdot$ to funkcja z $G \times G$ w G, spełniająca następujące warunki:
   - **Łączność** – dla każdych trzech elementów $a, b, c \in G$ jest spełnione: $a \cdot (b \cdot c) = (a \cdot b) \cdot c$
   - **Element neutralny** – istnieje element $e \in G$, taki, że dla każdego elementu $a \in G$ prawdą jest $e \cdot a = a \cdot e = a$
   - **Element odwrotny** – dla każdego elementu a istnieje element a^{-1}, taki, że $a \cdot a^{-1} = a^{-1} \cdot a = e$

Definicja 3.1.2. Mówimy, że grupa jest **abelowa**, jeśli dla dowolnych dwóch elementów $x, y \in G$ jest tak, że:

$$x \cdot y = y \cdot x$$

3.1.2 Przykłady

Lemat 3.1.1. Liczby całkowite ze składaniem $(\mathbb{Z}, +)$ są grupą.

Dowód. Łączność jest oczywista. Element neutralny to 0, a żeby otrzymać element odwrotny dla a, wystarczy wziąć $-a$. □

Lemat 3.1.2. Dla każdego $n \in \mathbb{N}_{>0}$ wszystkie permutacji z operacją kompozycji $(S_n, \circ)$ są grupą.

Dowód. Element neutralny to permutacja w której dla każdego i na miejscu i stoi liczba i. Dla pokazania łączności wystarczy przy składaniu permutacji rozpatrzyć każdy element osobno.

Żeby uzyskać element odwrotny z danej permutacji σ, wystarczy zrobić taką permutacje

$$\sigma^{-1}(i) = j, \text{ gdzie } \sigma(j = i)$$

, skoro każda permutacja jest bijekcją, to to się uda. □

3.1.3 Zastosowania

Definicja grupy obejmuje wiele znanych struktur matematycznych, więc dowodząc jakieś twierdzenie dla grup, tak naprawdę dostajemy dużo wyników. Na przykład grupy są często wykorzystane w teorii liczb.

Grupy są również użyteczne do definiowania innych obiektów matematycznych, np. ciał.

3.2 Przestrzenie wektorowe, baza (czy zawsze istnieje), wymiar. Podaj przykłady.

3.2.1 Ciała

Definicja 3.2.1. Ciałem nazywamy tuplę $\mathbb{K} = (K, +, \cdot)$, w której:

1. działanie $+$ jest łączne, (tzn. $\forall_{x,y,z \in K} \ (x + y) + z = x + (y + z)$);
2. działanie $\cdot$ jest łączne, (tzn. $\forall_{x,y,z \in K} \ (x \cdot y) \cdot z = x \cdot (y \cdot z)$);
3. działanie $+$ jest przemienne (tzn. $\forall_{x,y \in K} \ x + y = y + x$)
4. działanie $\cdot$ jest przemienne (tzn. $\forall_{x,y \in K} \ x \cdot y = y \cdot x$);
5. istnieje taki element (oznaczany jako $0 \in K$), który jest elementem neutralnym działania $+$ (tzn. $\forall_{x \in K} \ x + 0 = x$);
6. istnieje taki element (oznaczany jako $1 \in K$), który jest elementem neutralnym działania $\cdot$ (tzn. $\forall_{x \in K} \ x \cdot 1 = x$);
7. dla każdego $x \in K$ istnieje jakiś element (oznaczany jako $-x$), taki że $x + (-x) = 0$);
8. dla każdego $x \in K$ takiego że $x \neq 0$ istnieje taki element $x' \in K$, że $x \cdot x' = 1$;
9. działanie $\cdot$ jest rozdzielne względem działania $+$ (tzn. $\forall x, y, z \in K \ x \cdot (y + z) = x \cdot y + x \cdot z$.

3.2.2 Przestrzenie wektorowe

Definicja 3.2.2. Przestrzeń wektorowa to tupla $\mathbb{V} = (V, \oplus, \odot, \mathbb{K})$ taka, że:

1. $\mathbb{K} = (K, +, \cdot)$ jest ciałem;
2. $\oplus$ jest funkcją $V \times V \to V$;
3. $\odot$ jest funkcją $K \times V \to V$;
4. $(V, \oplus)$ jest grupą abelową;
5. spełnione są następujące własności:
   (a) dla dowolnego $\lambda \in K$ i dowolnych $x, y \in V$ jest tak, że $\lambda \odot (x \oplus y) = (\lambda \odot x) \oplus (\lambda \odot y)$;
   (b) dla dowolnych $\lambda, \mu \in K$ i dla dowolnego $x \in V$ mamy $(\lambda + \mu) \odot x = (\lambda \odot x) \oplus (\mu \odot x)$;
   (c) dla dowolnych $\lambda, \mu \in K$ i dla dowolnego $x \in V$ mamy $(\lambda \cdot \mu) \odot x = \lambda \odot (\mu \odot x)$;
   (d) Jeśli 1 to element neutralny z $\mathbb{K}$ względem operacji $\cdot$, to dla dowolnego $x \in V$ jest tak, że $1 \odot x = x$.

W praktyce $\odot$ i $\cdot$ notuje się w taki sam sposób (jako $\cdot$), a $\oplus$ i $+$ analogicznie (jako $+$). Formalnie to są jednak różne działania, które *a priori* mogą mieć się do siebie nijak, o ile spełniają zapostulowane własności.

Elementy V zwykliśmy określać mianem *wektorów*, a elementy K mianem skalarów.

Przykład 3.2.1. Weźmy sobie jakieś ciało $\mathbb{K} = (K, +, \cdot)$. Wówczas możemy sobie zdefiniować przestrzeń wektorową dla $V = K^n$, gdzie $n \in \mathbb{N}$ (innymi słowy: dla tupli długości n, zawierających elementy z K).

Działanie $\oplus$ możemy zdefiniować jako point-wise aplikację działania $+$ z ciała:

$$(a_1, a_2, a_3, \ldots, a_n) \oplus (b_1, b_2, b_3, \ldots, b_n) = (a_1 + b_1, a_2 + b_2, a_3 + b_3, \ldots, a_n + b_n).$$

Jak i działanie $\odot$ możemy zdefiniować jako mnożenie point-wise, korzystając z działania $\cdot$ z ciała:

$$\lambda \odot (a_1, a_2, a_3, \ldots, a_n) = (\lambda \cdot a_1, \lambda \cdot a_2, \lambda \cdot a_3, \ldots, \lambda \cdot a_n)$$

Uwaga: w praktyce (jak już pisaliśmy w definicji) coś takiego zapisze się tak:

$$(a_1, a_2, a_3, \ldots, a_n) + (b_1, b_2, b_3, \ldots, b_n) = (a_1 + b_1, a_2 + b_2, a_3 + b_3, \ldots, a_n + b_n).$$

$$\lambda \cdot (a_1, a_2, a_3, \ldots, a_n) = (\lambda \cdot a_1, \lambda \cdot a_2, \lambda \cdot a_3, \ldots, \lambda \cdot a_n)$$

Ale podmiot liryczny niniejszego tekstu nie jest fanem takiego stanu rzeczy. Znaczy, tak później też zaczyna notować bo jest szybciej, ale warto mimo wszystko być świadomym.

Przykład 3.2.2. Analogicznie jak wyżej, ale z macierzami zawierającymi elementy z K.

Przykład 3.2.3. Weźmy sobie $\mathbb{K} = (K, +, \cdot)$. Wówczas doprawdy fascynującą przestrzenią wektorową jest:

$$\mathbb{V} = (K, +, \cdot, \mathbb{K})$$

Skalary z ciała są jednocześnie wektorami, a za dodawanie wektorów (i ich mnożenie) służą nam zwyczajne operacje dodawania i mnożenia z ciała. Wzruszające.

Zasadniczo, przestrzeń wektorową jesteśmy w stanie zdefiniować niemalże w identyczny sposób dla jakichkolwiek struktur o jasno zdefiniowanej semantyce „współrzędnych" (np. dla wielomianów o współczynnikach z K). Nie są to, oczywiście, jedynie przestrzenie wektorowe które możemy sobie definiować. To nie jest tak, że przestrzenie wektorowe można jedynie definiować nad tuplami w $\mathbb{R}^n$ (określanymi również jako wektory, haha).

Definicja 3.2.3. Mówimy, że $\mathbb{V}_2 = (V_2, \oplus, \odot, \mathbb{K})$ jest **podprzestrzenią wektorową** $\mathbb{V}_1 = (V_1, \oplus, \odot, \mathbb{K})$ (oznaczane jako $\mathbb{V}_2 \leq \mathbb{V}_1$) jeśli:

1. $V_2 \subseteq V_1$;
2. V_2 jest zamknięte na $\oplus$[1].

[1]Zamkniętość ta dotyczy wykonania **skończenie wielu** takich operacji. W nieskończoności może nas „wyrzucić".

Lemat 3.2.1. Dla przestrzeni wektorowej $\mathbb{V} = (V, \oplus, \odot, \mathbb{K})$ oraz dowolnych dwóch jej podprzestrzeni wektorowych:

$$\mathbb{A} = (A, \oplus, \odot, \mathbb{K}) \leq \mathbb{V}$$
$$\mathbb{B} = (B, \oplus, \odot, \mathbb{K}) \leq \mathbb{V}$$

Mamy, że $\mathbb{C} = (A \cap B, \oplus, \odot, \mathbb{K})$ jest podprzestrzenią wektorową $\mathbb{V}$.

Dowód. 1. Jako, że $A \subseteq V$ i $B \subseteq V$, to trywialne mamy, że $A \cap B \subseteq V$;

2. Dla dowolnej pary $x, y \in A \cap B$ wiemy, że $x \oplus y \in A$ (bo $x, y \in A$) oraz, że $x \oplus y \in B$ (bo $x, y \in B$) (korzystamy z faktu, że $\mathbb{A} \leq \mathbb{V}$ oraz $\mathbb{B} \leq \mathbb{V}$).

□

Definicja 3.2.4. Mówimy, że $\mathbb{V}_2 = (V_2, \oplus, \odot, \mathbb{K})$ jest **podprzestrzenią generowaną zbiorem** X dla $X \subseteq V$ (oznaczane również jako $\mathrm{Lin}(X)$ jeśli jest to najmniejsza podprzestrzeń przestrzeni $\mathbb{V} = (V, \oplus, \odot, \mathbb{K})$ zawierająca zbiór X. Mamy, że:

$$\begin{aligned} V_2 = \mathrm{Lin}(X) &= \bigcap \{A \ : \ ((A, \oplus, \odot, \mathbb{K}) \leq \mathbb{V}) \wedge (X \subseteq A)\} \\ &= \left\{ \sum_{i=1}^{n} \lambda_i x_i : \quad n \in \mathbb{N}, \quad \lambda_1, \lambda_2, \ldots, \lambda_n \in \mathbb{K}, \quad x_1, x_2, \ldots, x_n \in X \right\} \end{aligned}$$

W ostatniej równości sumujemy się z użyciem operacji $\oplus$. Jednocześnie $\lambda_i x_i$ to formalnie $\lambda_i \odot x_i$.

Definicja 3.2.5. Jeśli $\mathbb{V} = (V, \oplus, \odot, \mathbb{K})$ jest przestrzenią wektorową, to mówimy że wektor x jest **kombinacją liniową** wektorów z $X \subseteq V$ jeśli $x \in \mathrm{Lin}(X)$.

Równoważnie, mówimy że x jest kombinacją liniową wektorów z X, o ile istnieją takie $x_1, x_2, \ldots, x_n \in X$ i $\lambda_1, \lambda_2, \ldots, \lambda_n \in K$, że:

$$x = \sum_{i=1}^{n} \lambda_i x_i$$

gdzie $\sum$ (ponownie) sumuje z użyciem operacji $\oplus$.

Definicja 3.2.6. Mówimy, że zbiór $X \subseteq V$ (gdzie $\mathbb{V} = (V, \oplus, \odot, \mathbb{K})$) jest **zbiorem liniowo niezależnym**, jeżeli dla każdego $x \in X$ jest tak, że $x \notin \mathrm{Lin}(X \setminus \{x\})$.

Równoważnie, zbiór X jest liniowo niezależny jeżeli żaden $x \in X$ nie jest kombinacją liniową elementów z $X \setminus \{x\}$.

Równoważnie, zbiór X jest liniowo niezależny jeżeli dla $\lambda_1, \lambda_2, \ldots, \lambda_n \in K$ i $x_1, x_2, \ldots, x_n \in V$ zachodzi:

$$\sum_{i=1}^{n} \lambda_i x_i = 0$$

to $\lambda_1 = \lambda_2 = \cdots = \lambda_n = 0$.

Definicja 3.2.7 (Baza)**.** Mówimy, że zbiór X jest **bazą** przestrzeni wektorowej $\mathbb{V} = (V, \oplus, \odot, \mathbb{K})$, jeżeli $\mathrm{Lin}(X) = V$ oraz X jest liniowo niezależny.

Twierdzenie 3.2.1. Każda przestrzeń wektorowa ma bazę.

Dowód. Zdefiniujmy sobie przestrzeń wektorową $\mathbb{V} = (V, \oplus, \odot, \mathbb{K})$. Weźmy sobie jakieś $G \subseteq V$ takie, że $\mathrm{Lin}(G) = V$. Takie G musi istnieć (w szczególności $G = V$).

Rozpatrzmy teraz zbiór P, taki że:

$$P = \{G' : G' \subseteq G \wedge G' \text{ jest liniowo niezależny}\}$$

Na zbiorze P definiujemy poset $\mathbb{P} = (P, \subseteq)$ (a więc uporządkowany relacją inkluzji.

Teraz tego się nie spodziewaliście, bo wchodzi tu coś mocniejszego niż hiszpańska inkwizycja. To **lemat Kuratowskiego-Zorna**. Ponieważ relacja inkluzji w oczywisty sposób „tworzy" nam porządek, wystarczy nam pokazać że majoranta dowolnego łańcucha w $\mathbb{P}$ znajduje się w P:

1. Jako majorantę łańcucha L bierzemy $M = \bigcup L$.
2. Dowodzimy, że $M \in P$:
   (a) Zakładamy nie wprost, że $M \notin P$, a zatem M nie jest zbiorem liniowo niezależnym.
   (b) Założenie nie wprost implikuje, że istnieje takie $x \in M$, że jest ono kombinacją liniową elementów z $M \setminus \{x\}$.
   (c) Zatem (z definicji) mamy, że istnieją takie $\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n \in K$ oraz $x_1, x_2, \ldots, x_n \in M \setminus \{x\}$ że:
   $$x = \sum_{i=1}^{n} \lambda_i x_i$$
   (d) Z tego, że $M = \bigcup L$ wynika, że istnieją jakieś zbiory $X_1, X_2, \ldots, X_n \in M$ takie, że $x_1 \in X_1, x_2 \in X_2, \ldots, x_n \in X_n$.
   (e) Bez straty ogólności $X_1 \subseteq X_2 \subseteq \cdots \subseteq X_n$ (jakiś porządek liniowy na nich musi być, bo należą do jednego łańcucha).
   (f) Wnioskujemy wobec tego, że $x_1, x_2, \ldots, x_n \in X_n$.
   (g) Jako że $x \in M$, to istnieje również jakieś $X \in L$ takie, że $x \in X$.
   (h) Jako, że $X \in L$ i $X_n \in L$ to te dwa byty są porównywalne:
      i. Jeśli $X \subseteq X_n$ to $x \in X_n$. Zauważmy jednak, że jako że $X_n \in P$, to jest on liniowo niezależny, a x z założenia nie wprost był kombinacją liniową $x_1, x_2, \ldots, x_n \in X_n$. Sprzeczność.
      ii. Jeśli $X_n \subseteq X$, to $x_1, x_2, \ldots, x_n \in X$ (oraz oczywiście $x \in X$). Ponownie, x z założenia nie wprost jest kombinacją liniową $x_1, x_2, \ldots, x_n$, ale X jest liniowo niezależne (bo $X \in P$). Sprzeczność.

W takim razie w $\mathbb{P}$ znajduje się element maksymalny. Czym jest element maksymalny w $\mathbb{P}$? Otóż jest to maksymalny pod względem inkluzji zbiór $\gamma \subseteq G$ taki, że jest liniowo niezależny.

Zauważamy teraz 2 szokujące rzeczy dotyczące zbioru γ:

1. Dla dowolnego wektora $v \in \gamma$ jest tak, że $v \in \mathrm{Lin}(\gamma)$. To jest ta mniej szokująca rzecz.

2. Dla dowolnego wektora v takiego, że $v \in G$, ale $v \notin \gamma$ dzieje się coś iście skandalicznego. Rozpatrzmy sobie zbiór $\gamma' = \gamma \cup \{v\}$. Niewątpliwie $\gamma \subseteq \gamma'$. Jednak to γ jest elementem maksymalnym w $\mathbb{P}$, co może oznaczać tylko jedno – γ' jest liniowo zależny!

   Ale skoro γ nie jest liniowo zależny, a γ' już jest, to znaczy że v jest kombinacją liniową elementów z γ. Czyli $v \in \mathrm{Lin}(\gamma)$.

Z powyższych dwóch punktów wynika, że $G \subseteq \mathrm{Lin}(\gamma)$, skąd $\mathrm{Lin}(G) \subseteq \mathrm{Lin}(\gamma)$. Jednocześnie $\mathrm{Lin}(G) = V$, czyli $V = \mathrm{Lin}(G) \subseteq \mathrm{Lin}(\gamma)$.

Skoro γ generuje całe V i w dodatku jest liniowo niezależne to znaczy, że jest bazą $\mathbb{V}$. □

Fakt 3.2.1. Jeśli $\mathbb{V}$ jest przestrzenią wektorową a B_1, B_2 to jej bazy, to $|B_1| = |B_2|$.

Definicja 3.2.8. Jeśli $\mathbb{V}$ jest przestrzenią wektorową, a B jest jej bazą, to **wymiarem** $\mathbb{V}$ nazwiemy $|B|$.

Przykład 3.2.4. Załóżmy $n \in \mathbb{N}$ oraz $\mathbb{V} = (\mathbb{R}^n, \oplus, \odot, \mathbb{R})$ z $\oplus$ i $\odot$ działającymi „point-wise" (jak we wcześniejszym przykładzie).

Wówczas bazą $\mathbb{V}$ jest (na przykład) zbiór wektorów $\{(1, 0, \ldots, 0), (0, 1, \ldots, 0), \ldots, (0, 0, \ldots, 1)\}$. Tym samym wymiar tej przestrzeni wektorowej to n.

Przykład 3.2.5. W przykładzie 3.2.3 poruszyliśmy temat bardzo fascynującej przestrzeni wektorowej. Jej przykładową bazą jest $\{1\}$, bo dowolny wektor λ możemy zapisać jako $\lambda \cdot 1$. Tym samym jej wymiar to 1.

3.3 Opisz kilka metod rozwiązywania układów równań liniowych.

3.3.1 Reprezentacja równań w postaci macierzowej

Układy równań postaci:

$$\begin{cases} a_{1,1} \cdot x_1 + a_{1,2} \cdot x_2 + \ldots a_{1,n} \cdot x_n = b_1 \\ a_{2,1} \cdot x_1 + a_{2,2} \cdot x_2 + \ldots a_{2,n} \cdot x_n = b_2 \\ \quad \vdots \\ a_{m,1} \cdot x_1 + a_{m,2} \cdot x_2 + \ldots a_{m,n} \cdot x_n = b_m \end{cases}$$

będziemy zapisywać w ten sposób:

$$\underbrace{\begin{bmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,n} \\ a_{2,1} & a_{2,2} & \ldots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \ldots & a_{m,n} \end{bmatrix}}_{\text{Macierz } A} \cdot \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}}_{\text{Wektor } x} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_m \end{bmatrix}}_{\text{Wektor } b}$$

Celem znalezienia rozwiązania oczywiście poszukiwać będziemy wektora x, mając daną macierz A i wektor b.

3.3.2 Wzory Cramera

Twierdzenie 3.3.1 (Wzory Cramera)**.** Jeśli $A \in \mathbb{R}^{n\times n}$ jest kwadratową, odwracalną macierzą współczynników, a $x \in \mathbb{R}^n$ i $b \in \mathbb{R}^n$ wektorami, to równanie:

$$Ax = b$$

ma dokładnie jedno rozwiązanie. Rozwiązaniem tym jest wektor x taki, że jego i-ta współrzędna wynosi:

$$x_i = \frac{\det A_i[b]}{\det A}$$

gdzie jako $A_i[b]$ rozumiemy macierz A, gdzie w miejscu i-tej kolumny znajduje się wektor b (tzn. $A_{1,\,i} = b_1$, $A_{2,\,i} = b_2$ i tak dalej).

Dowód. Wyprowadzenie tych wzorów zdaje się być całkiem nieprzyjemne. Na szczęście nie musimy ich wyprowadzać (bo już ktoś to zrobił), a my jedynie udowodnimy ich poprawność, haha[2]!

Po pierwsze musimy jednak zauważyć, że dla równania postaci $Ax = b$ możemy otrzymać poprawny wynik, mnożąc przez macierz odwrotną do macierzy A, tzn. A^{-1}. Wynik to wtedy $x = A^{-1}b$. Jest to również jedyne rozwiązanie (bo wykonaliśmy przekształcenie równoważne)[3].

Po drugie: pokazujemy, że dla równania postaci:

$$I \cdot x = b$$

wzór ten daje poprawny wynik. Chcemy zatem policzyć (dla każdego i):

$$x_i = \frac{\det A_i[b]}{\det I} = \det A_i[b]$$

Oczywiście w przypadku takiego równania chcielibyśmy, by śmieszne wzorki dały nam, że dla każdego i jest tak, że $x_i = b_i$.

Pozostaje pytanie ile to $\det A_i[b]$. Zobaczmy zatem jak wygląda ta macierz:

$$\begin{bmatrix} 1 & 0 & \dots & b_1 & \dots & 0 \\ 0 & 1 & \dots & b_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & \dots & b_n & \dots & 1 \end{bmatrix}$$

Korzystając z permutacyjnej definicji wyznacznika, łatwo widać że $\det A_i[b] = b_i$ (bo b_i będzie na przekątnej).

Powiecie: dzięki wielkie ponton za ten fascynujący dowód, ale fakt działania wzorów Cramera w tak trywialnym przypadku średnio nam pomaga.

Dlatego robimy teraz coś zabawnego: pokazujemy, że jeśli wymnożymy nasze równanie stronami (ze strony lewej) przez jakąś macierz odwracalną, to rozwiązanie dane nam przez wzory Cramera się nie zmieni.

[2] W chwili pisania tych słów nie było wiadomo, czy taki trik przejdzie na egzaminie licencjackim.

[3] Formalnie należy jeszcze udowodnić, że jeśli istnieje jedno rozwiązanie, to A jest odwracalna.

Lemat 3.3.1 (Kacpra Topolskiego)**.** Jeśli Q jest macierzą odwracalną (w szczególności ma niezerowy wyznacznik), to wzory Cramera dla równań:

$$Ax = b$$

i

$$QAx = Qb$$

dadzą identyczny wynik, tzn.

$$\forall_i \frac{\det A_i[b]}{\det A} = \frac{\det (QA)_i[Qb]}{\det QA}$$

Dowód. Podstawowa obserwacja którą należy tutaj wykonać jest taka, że:

$$(QA)_i[Qb] = Q(A_i[b])$$

Innymi słowy, jeśli wymnożymy macierze Q i A i potem do i-tej kolumny wstawimy wektor b, to wynik nasz jest taki sam jakbyśmy do macierzy A wstawili do i-tej kolumny wektor b, a potem wymnożyli ją z macierzą Q.

Może zdawać się to nieintuicyjne, więc zaczniemy ze spokojną obserwacją: zarówno w przypadku lewej, jak i prawej strony macierze te „różnią się" z macierzą QA jedynie w i-tej kolumnie.

W przypadku lewej strony jest to oczywiste (bo wzięliśmy macierz QA i wstawiliśmy coś do i-tej kolumny).

W przypadku prawej strony jest to również oczywiste, bo jeśli zmodyfikowaliśmy jedynie i-tą kolumnę w macierzy A, to w mnożeniu macierzowym ta i-ta kolumna może „wpłynąć" jedynie na wyniki które będą się znajdować w i-tej kolumnie wynikowej macierzy.

Tym samym, wystarczy nam teraz pokazać że wartości w i-tych kolumnach obu stron są identyczne.

Weźmy sobie zatem element z j-tego wiersza w i-tej kolumnie, zarówno dla strony lewej jak i prawej.

W przypadku lewej strony, jest to j-ty element wektora Qb. Innymi słowy:

$$(Qb)_j = Q_{j,\,1} \cdot b_1 + Q_{j,\,2} \cdot b_2 + \cdots + Q_{j,\,n} \cdot b_n$$

Po prawej stronie sytuacja rysuje się podobnie:

$$\begin{aligned} Q(A_i[b])_{j,\,i} &= Q_{j,\,1} \cdot (A_i[b])_{1,\,i} + Q_{j,\,2} \cdot (A_i[b])_{2,\,i} + \cdots + Q_{j,\,n} \cdot (A_i[b])_{n,\,i} \\ &= Q_{j,\,1} \cdot b_1 + Q_{j,\,2} \cdot b_2 + \cdots + Q_{j,\,n} \cdot b_n \\ &= (Qb)_j \end{aligned}$$

Jeśli mamy tę obserwację, to zauważamy że:

$$\begin{aligned}\frac{\det (QA)_i[Qb]}{\det QA} &= \frac{\det Q(A_i[b])}{\det Q \det A} \\ &= \frac{\det Q \det A_i[b]}{\det Q \det A} \\ &= \frac{\det A_i[b]}{\det A}\end{aligned}$$

A to jest to, co chcieliśmy wykazać. □

Jest to prawdziwie szokujące odkrycie, bo w ten sposób możemy już wykazać działanie wzorów Cramera dla dowolnego przypadku, gdzie mamy macierz odwrotną macierzy A.

Bierzemy równanie które dostaliśmy:

$$Ax = b$$

mnożymy je lewostronnie przez A^{-1}:

$$Ix = A^{-1}b$$

otrzymujemy teraz równanie, dla którego wzory Cramera na pewno działają poprawnie i, co więcej, ma identyczne rozwiązanie z rozwiązaniem naszego równania. Jednocześnie dowiedliśmy, że mnożenie przez macierz odwracalną (taką jak A^{-1}, bo jej odwrotnością jest A) zachowuje nam wynik zwracany przez wzory Cramera. Tym samym wnioskujemy, że dla równań postaci $Ax = b$ również działają poprawnie (pod warunkiem, że istnieje macierz odwracalna).

□

3.3.3 Eliminacja Gaussa

3.4 Odwzorowania liniowe i wieloliniowe.

Definicja 3.4.1 (Transformacja liniowa)**.** Jeżeli $\mathbb{K} = (K, +, \cdot)$ to ciało, a $\mathbb{V}_1 = (V_1, \oplus_1, \odot_1, \mathbb{K})$ i $\mathbb{V}_2 = (V_2, \oplus_2, \odot_2, \mathbb{K})$ są przestrzeniami wektorowymi nad tym ciałem, to funkcję $f : V_1 \to V_2$ nazywamy **transformacją liniową**, jeżeli:

1. $f(x + y) = f(x) + f(y)$
2. dla $\lambda \in K$ $f(\lambda \odot_1 x) = \lambda \odot_2 f(x)$ (co zwykło się po prostu notować jako $f(\lambda x) = \lambda f(x)$)

Transformacja liniowa jest również określana jako **homomorfizm**.

Definicja 3.4.2 (Morfizmy)**.** Niech $\mathbb{K} = (K, +, \cdot)$ to ciało, a $\mathbb{V}_1 = (V_1, \oplus_1, \odot_1, \mathbb{K})$ i $\mathbb{V}_2 = (V_2, \oplus_2, \odot_2, \mathbb{K})$ będą przestrzeniami wektorowymi nad tym ciałem.

Niech $f : V_1 \to V_2$ jest transformacją liniową. Wówczas:

1. Jeśli f jest iniektywna, nazwiemy ją **monomorfizmem**;
2. jeśli f jest surjektywna, nazwiemy ją **epimorfizmem**;
3. jeśli f jest bijektywna, nazwiemy ją **izomorfizem**;
4. jeśli $\mathbb{V}_1 = \mathbb{V}_2$, nazwiemy ją **endomorfizmem**;
5. jeśli f jest bijektywna i jest endomorfizmem, nazwiemy ją **endomorfizmem bijektywnym**.

Fakt 3.4.1. Jeśli $f : \mathbb{V}_1 \to \mathbb{V}_2$ jest transformacją liniową, to:

1. $f(0) = 0$
2. Dla $x \in V_1$ mamy $f(-x) = -f(x)$, gdzie „$-$” oznacza wzięcie elementu odwrotnego względem odpowiednio operacji $\oplus_1$ i $\oplus_2$.
3. Dla $\lambda_0, \lambda_1, \ldots, \lambda_n \in K$ oraz $x_1, x_2, \ldots, x_n \in V_1$ mamy, że:

$$f\left(\sum_{i=1}^{n} \lambda_i x_i\right) = \sum_{i=1}^{n} \lambda_i f(x_i)$$

Przykład 3.4.1. Jeśli $\mathbb{V}$ jest przestrzenią wektorową, to $f : V \to V$ taka, że $f(x) = x$ jest transformacją liniową.

Przykład 3.4.2. Jeśli $\mathbb{V} = (V, \oplus, \odot, \mathbb{K})$ jest przestrzenią wektorową wielomianów nad ciałem $\mathbb{K}$ (tzn. $V = \{v \in K^{\mathbb{N}} : v \text{ ma skończenie wiele współrzędnych z niezerowymi wartościami}\}$ i intuicyjnymi operacjami dodawania i wymnażania przez skalar), to operacja „wzięcia pochodnej” wielomianu jest przekształceniem liniowym.

Definicja 3.4.3 (Transformacja wieloliniowa)**.** Jeśli $\mathbb{K}$ jest ciałem, a $\mathbb{V}_1, \mathbb{V}_2, \ldots, \mathbb{V}_n$ oraz $\mathbb{W}$ są przestrzeniami wektorowymi nad tym ciałem, to funkcję f:

$$f : V_1 \times V_2 \times \cdots \times V_n \to W$$

Nazywamy **odwzorowaniem wieloliniowym** jeśli dla każdego i zachodzi:

1. $f(x_1, x_2, \ldots, x_i + x_i', \ldots, x_n) = f(x_1, x_2, \ldots, x_i, \ldots, x_n) + f(x_1, x_2, \ldots, x_i', \ldots, x_n)$
2. $f(x_1, x_2, \ldots, \lambda x_i, \ldots, x_n) = \lambda f(x_1, x_2, \ldots, x_i, \ldots, x_n)$

gdzie $x_1 \in V_1, x_2, \in V_2, \ldots, x_n \in V_n$, a $\lambda \in K$.

3.5 Wyznacznik macierzy i jego zastosowania.

3.5.1 Definicje wyznacznika

Definicja 3.5.1 (Definicja permutacyjna wyznacznika)**.** Jeśli $\mathbb{K}$ jest ciałem i $n \in \mathbb{N}$, to **wyznacznikiem** nazwiemy funkcję $f : K^{n \times n}$ taką, że:

$$f(A) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \cdot A_{1,\,\sigma(1)} \cdot A_{2,\,\sigma(2)} \cdot A_{3,\,\sigma(3)} \cdot \cdots \cdot A_{n,\,\sigma(n)}$$

gdzie A_{ij} oznacza komórkę macierzy znajdującą się w i-tym wierszu i j-tej kolumnie, a S_n oznacza zbiór wszystkich permutacji zbioru n-elementowego.

Definicja 3.5.2 (Definicja „objętościowa” wyznacznika)**.** Niech $\mathbb{K}$ będzie ciałem i $n \in \mathbb{N}$. Ponadto, niech $v_1, v_2, v_3, \ldots, v_n \in K^n$. Wówczas tuplę $(v_1, v_2, v_3, \ldots, v_n) \in (K^n)^n$ możemy trywialnie utożsamić z macierzą $M \in K^{n \times n}$ (i vice versa), poprzez utożsamienie każdego wektora K^n z pojedynczym wierszem tej macierzy.

Stosując taką notację, mówimy że funkcja $f : K^{n \times n} \to K$ jest **wyznacznikiem**, jeśli spełnia następujące warunki:

1. $f(v_1, v_2, \ldots, \lambda v_i, \ldots, v_n) = \lambda \cdot f(v_1, v_2, \ldots, v_i, \ldots, v_n)$

2. $f(v_1, v_2, \ldots, v_i + v_i', \ldots, v_n) = f(v_1, v_2, \ldots, v_i, \ldots, v_n) + f(v_1, v_2, \ldots, v_i', \ldots, v_n)$
3. Jeżeli istnieje takie i, że $v_i = v_{i+1}$, to $f(v_1, v_2, \ldots, v_i, v_{i+1}, \ldots v_n) = 0$
4. f na macierzy identycznościowej przyjmuje wartość 1.

Postulaty te wynikają z chęci stworzenia funkcji obliczającej zorientowaną objętość wielowymiarowego równoległościanu (opisywanego wektorami).

Można wykazać, że dla określonego $n \in \mathbb{N}$ istnieje dokładnie 1 funkcja spełniająca wyżej wymienione warunki. Definicja ta okazuje się być równoważna z tą wcześniejszą.

Wyznacznik oznaczamy jako det.

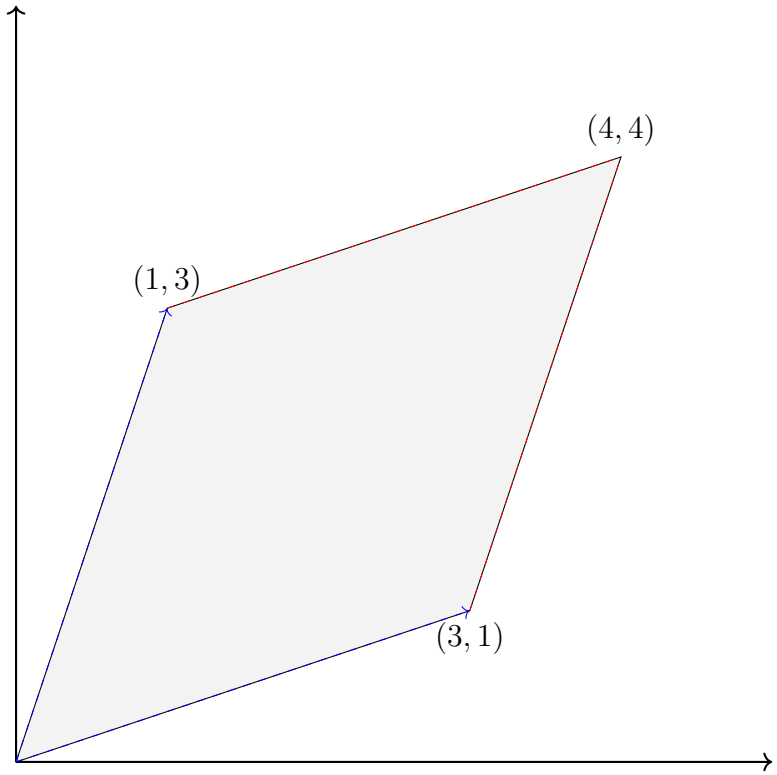
3.5.2 Własności wyznacznika

Fakt 3.5.1 (Własności wyznacznika)**.** Jeśli $\mathbb{K}$ jest ciałem, $n \in \mathbb{N}$, a $M, N \in K^{n \times n}$ są macierzami nad tym ciałem, to:

1. Zamiana wierszy macierzy miejscami zmienia jedynie znak wyznacznika;
2. przemnożenie wiersza macierzy przez stałą λ zwiększa wartość wyznacznika λ razy;
3. dodanie do wiersza innego wektora (przemnożonego przez jakąś stałą λ) nie zmienia wartości wyznacznika;
4. $\det MN = \det M \det N$
5. $\det \lambda M = \lambda^n \det M$
6. $\det M^{-1} = \frac{1}{\det M}$
7. $\det M^T = \det M$

3.5.3 Zastosowania wyznacznika

1. Wzory Cramera przy rozwiązywaniu równań liniowych;
2. liczenie objętości n-wymiarowego równoległościanu (opisanego wektorami znajdującymi się w kolejnych wierszach macierzy)
3. jeśli wyznacznik macierzy jest niezerowy to wiemy, że ma ona odwrotność.



Rysunek 3.1: Dwuwymiarowa figura geometryczna wyznaczona parą wektorów $(1, 3)$ i $(3, 1)$. Jeśli obliczymy wartość $\det \begin{bmatrix} 1 & 3 \\ 3 & 1 \end{bmatrix}$ to wyjdzie nam -8 (a więc po wzięciu wartości bezwzględnej dostaniemy 8, i to jest w istocie pole tej figury). Ale śmieszne!

3.6 Wartości własne macierzy – własności, metody liczenia i praktyczne zastosowania.

3.6.1 Definicje

Definicja 3.6.1 (Wektor własny macierzy)**.** Niech $\mathbb{K} = (K, +, \cdot)$ będzie ciałem, a $M \in K^{n \times n}$ będzie macierzą nad tym ciałem (gdzie $n \in \mathbb{N}$).

Wówczas wektor $v \in K^n$ nazwiemy **wektorem własnym** macierzy M, jeżeli istnieje takie $\lambda \in K$ że:

$$M \cdot v = \lambda \cdot v$$

W szczególności, wektor zerowy jest wektorem własnym dowolnej macierzy.

Definicja 3.6.2 (Wartość własna macierzy)**.** Niech $\mathbb{K} = (K, +, \cdot)$ będzie ciałem, a $M \in K^{n \times n}$ będzie macierzą nad tym ciałem (gdzie $n \in \mathbb{N}$).

Wówczas skalar $\lambda \in K$ nazwiemy **wartością własną** macierzy M jeśli istnieje taki (niezerowy[4]) wektor $v \in K^n$ że:

$$M \cdot v = \lambda \cdot v$$

[4]Inaczej ta definicja byłaby niepoważna

3.6.2 Własności i sposoby obliczania

Twierdzenie 3.6.1. Niech $\mathbb{K} = (K, +, \cdot)$ będzie ciałem, a $M \in K^{n \times n}$ będzie macierzą nad tym ciałem (gdzie $n \in \mathbb{N}$). Ponadto, niech $\lambda \in K$.

Następujące stwierdzenia są równoważne:

1. λ jest wartością własną macierzy M.
2. $\det(M - \lambda I) = 0$, gdzie I to macierz identycznościowa (rozmiaru $n \times n$).

Dowód. Zanim przejdziemy do rozumowania, przekształćmy równoważnie warunek na to, żeby λ było wartością własną:

$$\begin{aligned} Mv &= \lambda v \\ Mv - \lambda v &= 0 \\ (M \cdot I - \lambda \cdot I) \cdot v &= 0 \\ (M - \lambda I) \cdot v &= 0 \end{aligned}$$

(1) $\implies$ (2): Zakładamy λ jest wartością własną oraz (nie wprost), że $(M - \lambda I)$ ma niezerowy wyznacznik. W takim razie jest odwracalna. Lewostronnie więc mnożymy równanie przez jej odwrotność i otrzymujemy $v = 0$ jako jedyne rozwiązanie (co przeczy założeniu że λ jest wartością własną). Otrzymana sprzeczność pokazuje, że wyznacznik musi być zerowy.

(2) $\implies$ (1): Załóżmy, że wyznacznik macierzy $M - \lambda I$ jest zerowy. W takim razie wiemy, że zbiór wektorów tworzących wiersze macierzy M jest liniowo **zależny**. Z definicji liniowej zależności oznacza to, że istnieją takie $\lambda_1, \lambda_2, \ldots, \lambda_n \in K$ (przy czym oraz istnieje takie i, że $\lambda_i \neq 0$) że:

$$\sum_{i=1}^{n} \lambda_i \cdot x_i = 0$$

gdzie x_i to wektor znajdujący się w i-tym wierszu macierzy.

Oznacza to, że wektor $v = [\lambda_1, \lambda_2, \ldots, \lambda_n]$ jest taki, że:

$$(M - \lambda I)v = 0$$

czyli v jest wektorem własnym, a λ wartością własną. Ale fajnie.

□

Definicja 3.6.3 (Wielomian charakterystyczny)**.** Niech $\mathbb{K} = (K, +, \cdot)$ będzie ciałem, a $M \in K^{n \times n}$ będzie macierzą nad tym ciałem (gdzie $n \in \mathbb{N}$).

Funkcję $f : K \to K$ nazwiemy **wielomianem charakterystycznym** macierzy M, jeżeli $f(x) = \det(M - x \cdot I)$.

Fakt 3.6.1. Niech $\mathbb{K} = (K, +, \cdot)$ będzie ciałem, a $M \in K^{n \times n}$ będzie macierzą nad tym ciałem, a f będzie wielomianem charakterystycznym macierzy M.

Jeśli jakieś $\lambda \in K$ jest takie, że $f(\lambda) = 0$ (tzn. λ jest miejscem zerowym wielomianu f), to λ jest wartością własną macierzy M.

Dowód.

$$f(\lambda) = 0 \implies \det(M - \lambda I) = 0 \iff \lambda \text{ jest wartością własną}$$

□

Z powyższego faktu wynika, że aby poznać wartości własne możemy szukać miejsc zerowych wielomianu charakterystycznego.

3.7 Iloczyn skalarny i jego zastosowania.

3.7.1 Definicja

Definicja 3.7.1 (Iloczyn skalarny nad $\mathbb{R}$)**.** Niech $\mathbb{K} = (\mathbb{R}, +, \cdot)$ będzie ciałem, a $\mathbb{V} = (V, \oplus, \odot, \mathbb{K})$ będzie przestrzenią wektorową nad tym ciałem.

Wówczas **iloczynem skalarnym** nazwiemy funkcję $f : V \times V \to \mathbb{R}$ taką, że:

1. Jeśli $v_1, v_2 \in V$, to zachodzi $f(u, v) = f(v, u)$;
2. jeśli $v_1, v_2, v_3 \in V$ i $\lambda_1, \lambda_2 \in \mathbb{R}$, to zachodzi $f(\lambda_1 v_1 + \lambda_2 v_2, v_3) = \lambda_1 f(v_1, v_3) + \lambda_2 f(v_2, v_3)$;
3. jeśli dla jakiegoś $v \in V$ jest tak, że $v \neq 0$, to $f(v, v) > 0$.

W praktyce iloczyn skalarny wektorów $v_1, v_2 \in V$ oznaczany jest jako $\langle v_1, v_2 \rangle$.

Definicja 3.7.2 (Iloczyn skalarny nad $\mathbb{C}$)**.** Niech $\mathbb{K} = (\mathbb{C}, +, \cdot)$ będzie ciałem, a $\mathbb{V} = (V, \oplus, \odot, \mathbb{K})$ będzie przestrzenią wektorową nad tym ciałem.

Wówczas **iloczynem skalarnym** nazwiemy funkcję $f : V \times V \to \mathbb{C}$ taką, że:

1. Jeśli $v_1, v_2 \in V$, to zachodzi $f(u, v) = \overline{f(v, u)}$;
2. jeśli $v_1, v_2, v_3 \in V$ i $\lambda_1, \lambda_2 \in \mathbb{R}$, to zachodzi $f(\lambda_1 v_1 + \lambda_2 v_2, v_3) = \lambda_1 f(v_1, v_3) + \lambda_2 f(v_2, v_3)$;
3. jeśli dla jakiegoś $v \in V$ jest tak, że $v \neq 0$, to $f(v, v) > 0$.

W praktyce iloczyn skalarny wektorów $v_1, v_2 \in V$ oznaczany jest jako $\langle v_1, v_2 \rangle$.

Definicja 3.7.3 (Przestrzeń euklidesowa)**.** Niech $\mathbb{K} = (\mathbb{R}, +, \cdot)$ będzie ciałem, a $\mathbb{V} = (V, \oplus, \odot, \mathbb{K})$ przestrzenią wektorową nad tym ciałem. Ponadto, niech $f : V \times V \to \mathbb{R}$ będzie iloczynem skalarnym nad $\mathbb{R}$.

Wówczas parę $(\mathbb{V}, f)$ nazwiemy **przestrzenią euklidesową**.

Definicja 3.7.4 (Przestrzeń unitarna)**.** Niech $\mathbb{K} = (\mathbb{C}, +, \cdot)$ będzie ciałem, a $\mathbb{V} = (V, \oplus, \odot, \mathbb{K})$ przestrzenią wektorową nad tym ciałem. Ponadto, niech $f : V \times V \to \mathbb{C}$ będzie iloczynem skalarnym nad $\mathbb{C}$.

Wówczas parę $(\mathbb{V}, f)$ nazwiemy **przestrzenią unitarną**.

Przykład 3.7.1. Przykładem przestrzeni euklidesowej $\mathbb{V}, f$ jest przestrzeń gdzie:

1. $\mathbb{V} = (\mathbb{R}^n, \oplus, \odot, \mathbb{R})$ jest przestrzenią wektorową z wektorami $\mathbb{R}^n$ i „standardowymi" operacjami;
2. $f(a, b) = \sum_{i=1}^{n} a_i b_i$

Definicja 3.7.5 (Norma wektora)**.** Niech $\mathbb{V}$ będzie przestrzenią wektorową nad $\mathbb{K}$. Wówczas funkcję $\|\cdot\| : V \to K$ taką, że $\|v\| = \sqrt{\langle v, v \rangle}$ nazwiemy **normą wektorową**.

3.7.2 Własności

Fakt 3.7.1. Niech $v \in \mathbb{V}$. Wówczas $\|\lambda v\| = \lambda \|v\|$.

Dowód.

$$\|\lambda v\| = \sqrt{\langle \lambda v, \lambda v \rangle} = \sqrt{\lambda^2 \langle v, v \rangle} = \lambda \sqrt{\langle v, v \rangle} = \lambda \|v\|$$

□

Fakt 3.7.2. Niech $v \in \mathbb{V}$. Wówczas następujące warunki są równoważne:

1. $\|v\| = 0$
2. $v = 0$

Dowód. Przekształcamy pierwszy warunek równoważnie:

$$\begin{aligned} \|v\| &= 0 \\ \sqrt{\langle v, v \rangle} &= 0 \\ \langle v, v \rangle &= 0 \\ v &= 0 \end{aligned}$$

Ostatnie przejście wynika bezpośrednio z definicji iloczynu skalarnego. □

Fakt 3.7.3.

Fakt 3.7.4. Niech $v_1, v_2 \in \mathbb{V}$. Wówczas $\|v_1 + v_2\| \leq \|v_1\| + \|v_2\|$

Dowód.

$$\|v_1\| + \|v_2\| = \sqrt{\langle v_1 + v_2, v_1 + v_2 \rangle}$$

□

3.8 Macierze ortogonalne.

3.9 Macierze diagonalne i diagonalizacja macierzy.

3.10 Zbiór Π_n wszystkich wielomianów stopnia nie większego niż n jako przestrzeń wektorowa. Podaj wymiar i kilka przykładowych baz.

3.11 Bazy przestrzeni wektorowych - przykłady.

3.12 Wyznacznik jako objętość.

3.13 Opisz metodę ortogonalizacji Grama Schmidta.

Rozdział 4

Matematyka dyskretna

Wykaz definicji

$R^{(k)}(l_1, l_2, l_3, \ldots, l_s; s)$ – ogólna liczba Ramseya; patrz sekcja 4.3.2

Kolorowanie wierzchołkowe grafu – funkcja $c : V \to [n]$ przypisująca każdemu wierzchołkowi grafu $G = (V, E)$ jakąś liczbę naturalną w taki sposób, że dla każdej pary $v_1, v_2 \in V$ mamy $c(v_1) = c(v_2) \implies (v_1, v_2) \notin E$; o liczbie n mówimy wówczas, że jest liczbą kolorów użytą przez kolorowanie. Po ludzku: kolorujemy wierzchołki grafu tak, by wierzchołki o tym samym kolorze nie były połączone.

Kolorowanie krawędziowe grafu – funkcja $c : E \to [n]$ przypisująca każdej krawędzi grafu $G = (V, E$ jakąs liczbę naturalną w ten sposób, że dla każdej pary $e_1 = (v_a, v_b), e_2 = (v_c, v_d) \in E$ mamy $c(e_1) = c(e_2) \implies \{v_a, v_b\} \cap \{v_c, v_d\} = \varnothing$; o liczbie n mówimy wówczas, że jest liczbą kolorów użytą przez kolorowanie. Po ludzku: kolorujemy krawędzie grafu tak, by krawędzie o tym samym kolorze nie wychodziły z jednego wierzchołka.

$\chi(G)$ – liczba chromatyczna grafu G; najmniejsza liczba kolorów potrzebna do wykonania poprawnego kolorowania wierzchołkowego

$\omega(G)$ – liczba klikowa grafu G; rozmiar największej kliki w grafie G

$\mathrm{col}(G)$ – liczba kolorująca grafu G; patrz sekcja 4.7.1

$\delta(G)$ – stopień najmniejszego wierzchołka w grafie G

$\Delta(G)$ – stopień największego wierzchołka w grafie G

$\mathrm{N}(A)$ – zbiór wszystkich sąsiadów danego podzbioru $A \subseteq V$ w jakimś grafie G

Przepływ całkowitoliczbowy, etc. – patrz sekcja 4.9.1

4.1 Zasada włączeń i wyłączeń. Przykłady zastosowań

4.1.1 Dowód zasady

Twierdzenie 4.1.1 (Zasada włączeń i wyłączeń)**.**

$$|\bigcup_{i\in[n]} A_i| = \sum_{\varnothing\neq X\subset[n]} (-1)^{|X|-1} \cdot |\bigcap_{i\in X} A_i| \tag{4.1}$$

Dowód. Weźmy sobie jakieś $x \in \bigcup A_i$ należące dokładnie do k zbiorów (gdzie $k \leq n$). Bez straty ogólności można założyć, że $x \in A_1, A_2, A_3, \ldots, A_k$ oraz $x \notin A_{k+1}, A_{k+2}, A_{k+3}, \ldots, A_n$. Zauważamy, że x w sumie występującej po lewej stronie postulowanej równości zostanie zliczone raz (oczywiste), a po prawej:

1. Sumując $A_1, A_2, A_3, \ldots, A_k$ zliczone zostanie k razy,
2. Sumując $A_1 \cap A_2, \ldots$ zliczone zostanie $\binom{k}{2}$ razy i odjęte od wcześniejszej wartości
3. Sumując l-elementowe przecięcia zbiorów $A_1, A_2, \ldots, A_k$ zostanie odjęte/dodane $\binom{k}{l}$ razy.

Mamy więc, że x jest zliczone $\binom{k}{1} - \binom{k}{2} + \binom{k}{3} - \binom{k}{4} \ldots$ razy. Powołujemy się wtedy na magiczny wzór, który wygląda następująco:

$$\sum_{i=0}^{k} (-1)^{i+1} \binom{k}{i} = -(1-1)^k = 0$$

Skąd mamy

$$\sum_{i=1}^{k} (-1)^{i+1} \binom{k}{i} = 1$$

bo $\binom{k}{0} = 1$. To prowadzi nas do konkluzji, że x po prawej stronie również został zliczony 1 raz, czyli wszystko działa tak jak powinno.

□

4.1.2 Zliczanie cykli Hamiltona

Zanim przejdziemy do twierdzenia, musimy wprowadzić niestety parę definicji. Przez U oznaczamy zbiór wszystkich **spacerów** długości $n+1$ w grafie w którym usiłujemy zliczyć liczbę cyklów Hamiltona (pod względem liczby wierzchołków; przez n oznaczamy oczywiście liczbę wierzchołków w grafie), zaczynających się w jakimś wierzchołku v_0 i również w nim kończących. Przez A_i oznaczamy zbiór wszystkich spacerów z U, które przechodzą przez wierzchołek o numerze i. Ponadto zakładamy, że

$$U = \bigcap_{i\in\varnothing} B$$

gdzie B jest czymkolwiek (w sensie serio czymkolwiek). Dlaczego tak zakładamy? Nie mam pojęcia, ale inaczej dowód by nie zadziałał. Formaliści mogą spojrzeć na definicję czapeczkowania jeśli czapeczkujemy jedynie po elementach zbioru pustego, but, honestly, I don't care. Ważna obserwacja zanim jeszcze przejdziemy do dowodu:

$$2 \cdot |H| = |\bigcap_{i\in[n]} A_i|$$

Gdzie H jest to zbiór zawierający wszystkie cykle Hamiltona. Ta obserwacja w sumie ma sens, bo jeśli jakiś spacer ma długość $n+1$ zaczyna się i kończy w v_0 i jednocześnie przechodzi przez każdy wierzchołek, to musi być cyklem Hamiltona (no bo dwa razy był w v_0, najpierw z niego wychodząc a potem do niego wchodząc, a w pozostałych $n-1$ wierzchołkach musiał być dokładnie raz bo cały spacer ma długość $n-1+2=n+1$). Należy zauważyć że cykle Hamiltona w ten sposób zliczamy zawsze podwójnie, bo cykle w grafach nieskierowanych mają to do siebie że można je obejść na 2 różne sposoby wychodząc z tego samego punktu, idąc „na prawo" lub „na lewo".

Twierdzenie 4.1.2 (O liczbie cyklów Hamiltona w grafie)**.**

$$|\bigcap_{i\in[n]} A_i| = \sum_{X\subset[n]} (-1)^{|X|} \cdot |\bigcap_{i\in X}(U \setminus A_i)| \tag{4.2}$$

Dowód. Po pierwsze należy zauważyć, że

$$|\bigcap_{i\in[n]} A_i| = |U| - |\bigcup_{i\in[n]}(U \setminus A_i)|$$

W sumie jak się to pokontempluje to zaczyna się to robić oczywiste. Jest to obserwacja czysto teoriomnogościowa; jeśli jakiś spacer znajduje się w przecięciu, to znaczy że jest w każdym zbiorze A_i dla dowolnego i, a więc po prawej stronie nie zostanie „usunięty" z uniwersum (tj. zbioru U) bo za każdym razem zostanie usunięty ze zbioru elementów „do wywalenia", a jeśli nie należy do przecięcia to znaczy że spacer ten nie należy do jakiegoś A_k, a więc $U \setminus A_k$ będzie go zawierać, a więc przy sumowaniu elementów „do wywalenia" zostanie on wliczony.

Pamiętacie nasze śmieszne założenie o tym, że U jest równe przecięciu po elementach zbioru pustego? Teraz go użyjemy, bo

$$|U| - |\bigcup_{i\in[n]}(U \setminus A_i)| = |\bigcap_{i\in\varnothing}(U \setminus A_i)| + \sum_{\varnothing\neq X\subset[n]} (-1)^{|X|} \cdot |\bigcap_{i\in X}(U \setminus A_i)|$$

To przejście może wyglądać przerażająco, ale w sumie nic ciekawego się tu nie stało; pierwszy element po lewej stronie (tj. U) rozpisaliśmy korzystając właśnie z założenia, że $U = \bigcap_{i\in\varnothing} B$, gdzie B jest czymkolwiek, a więc w szczególności $U \setminus A_i$. Drugi element z lewej strony rozpisaliśmy stosując po prostu zasadę włączeń i wyłączeń (aczkolwiek zmieniliśmy znak, by żyło się prościej, a więc i skorygowaliśmy potęgę przy (-1) by wszystko się zgadzało).

Teraz możemy sobie popatrzyć na to co nam wyszło, i uświadomić sobie, że teraz to już możemy sobie po prostu radośnie dodać pierwszy i drugi składnik z prawej strony:

$$|\bigcap_{i\in\varnothing}(U \setminus A_i)| + \sum_{\varnothing\neq X\subset[n]} (-1)^{|X|} \cdot |\bigcap_{i\in X}(U \setminus A_i)| = \sum_{X\subset[n]} (-1)^{|X|} \cdot |\bigcap_{i\in X}(U \setminus A_i)|$$

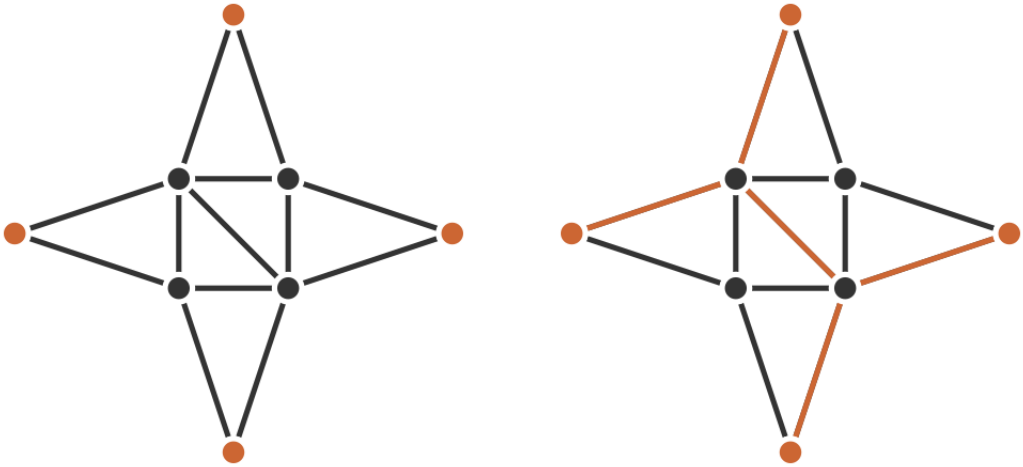
Co prowadzi nas do otrzymania postulowanej równości. □

A jak chcemy zastosować tę równość do zliczania ścieżek Hamiltona? Otóż jak sobie podumamy i oznaczymy $S_i = U \setminus A_i$ (czyli S_i to zbiór spacerów z uniwersum takich, że nie przechodzą przez wierzchołek i), to $|\bigcap_{i\in X} S_i|$ to są wszystkie spacery spełniające warunki bycia w uniwersum, ale na podgrafie indukowanym bez wierzchołków należących do X. A liczbę spacerów w grafie mających długość $n+1$ między dwoma wierzchołkami możemy policzyć, robiąc macierz incydencji wierzchołków i podnosząc ją do potęgi $n+1$. No i fajnie.

4.1.3 Problem drzewa Steinera

To nie jest przyjemny dowód i jest absolutnie koszmarny do sformalizowania. Mimo wszystko spróbuję. W sumie to nawet nie jest dowód, to jest bardziej algorytm postępowania.

Zacznijmy na początku od zdefiniowania problemu. Mamy graf G oraz zbiór wierzchołków, które z jakiegoś powodu nazywamy *terminalami*. Bardzo chcielibyśmy znaleźć najmniejszy (krawędziowo) podgraf, który łączy wszystkie terminale. Oczywiście widzimy, że musi być to drzewo (stąd nazwa), gdyż w przeciwnym razie moglibyśmy pozbyć się jakiejś krawędzi bez naruszania spójności.



Rysunek 4.1: Przykładowy graf z terminalami oraz minimalne drzewo je łączące

Spacerem rozgałęziającym się będziemy nazywać ukorzenione oraz uporządkowane (tzn. dzieci każdego wierzchołka są ponumerowane) drzewo, które „przerzucimy" w dosyć dziwny sposób na graf w którym szukamy drzewa Steinera. Generalnie wierzchołki drzewa mapujemy dowolnie na wierzchołki grafu, z tym że jeśli 2 wierzchołki w drzewie są połączone krawędzią, to po przemapowaniu również muszą być połączone krawędzią. Co ważne, **wiele wierzchołków drzewa może zostać przemapowane na ten sam wierzchołek naszego grafu**. Intuicyjnie odpowiada to wielu równoległym spacerom z tego samego punktu.

Formalniej można o tym myśleć jak o krotce

$$(T, \pi, \varphi)$$

gdzie T jest jakimś drzewem, π permutacją dzięki której mamy porządek dzieci, a φ funkcją mapującą wierzchołki tego drzewa na nasz graf, zgodnie z warunkiem który został tu opisany.

Bardzo chcielibyśmy umieć liczyć sobie, ile jest spacerów rozgałęziających się o danej długości, „zaczynających" się w danym wierzchołku. Okazuje się, że możemy to liczyć z pomocą programowania dynamicznego. Przed $dp[v][i]$ będziemy oznaczać liczbę spacerów rozgałęziających się „wychodzących" z wierzchołka v mających i krawędzi. Mamy, że:

$$dp[v][0] = 1$$

co jest dosyć oczywiste, bo skoro jest 0 krawędzi to ten spacer rozgałęziający się ma po prostu jeden wierzchołek i taki jest jeden. Dosyć prosto też zaobserwować, że

$$dp[v][1] = \deg(v)$$

Bo drzewo które będziemy chcieli zmapować na nasz graf będzie mieć 1 krawędź; tym samym jesteśmy w stanie przemapować jeden wierzchołek zawsze na wierzchołek v, a drugi na któregoś z jego sąsiadów. Mało odkrywcza obserwacja i w sumie to bezużyteczna, bo wprost będzie wynikać ze wzoru który zostanie zaraz podany.

Twierdzenie 4.1.3 (Straszny wzór)**.**

$$dp[u][i] = \sum_{v \in N(u)} \sum_{a+b=i-1} dp[v][a] \cdot dp[u][b] \tag{4.3}$$

Dowód. Rozważamy drzewo, które mapujemy na nasz graf. Ponieważ elementem spaceru rozgałęziającego się jest permutacja, to któreś dziecko naszego korzenia będzie w niej najbardziej „na lewo". To dziecko w swoim poddrzewie ma jakieś a krawędzi, a cała reszta drzewa ma jakieś b krawędzi. Jak się to narysuje to widać, że $a + b = i - 1$, bo krawędź łącząca korzeń z lewym dzieckiem łączyła 2 rozpatrywane przez nas poddrzewa, które teraz rozpatrujemy osobno. Teraz liczba spacerów rozgałęziających się to po prostu suma po wszystkich możliwych wierzchołkach na które mapujemy tamto dziecko i możliwych wartościach a, b i iloczynach spacerów $dp[v][a]$ i $dp[u][b]$ czyli liczbach spacerów rozgałęziających się długości a z tego dziecka i długości b z całej reszty. Wartości te już mamy policzone, więc to jest poprawnie zdefiniowane. Obliczenie wszystkich możliwych tych wartości działa jakoś wielomianowo.

Rysunek 4.2: Przykłady podziału spaceru na dwie części.
Po lewej $a = 2, b = 5$, po prawej $a = 7, b = 0$

□

Z tego wzoru istotnie wynika wcześniejsza obserwacja, bo

$$dp[u][1] = \sum_{v \in N(u)} \sum_{a+b=0} dp[v][a] \cdot dp[u][b] = \sum_{v \in N(u)} dp[v][0] \cdot dp[u][0] = \sum_{v \in N(u)} 1 \cdot 1 = \deg(u)$$

Teraz zaczniemy używać motywów bardzo podobnych do tych występujących w problemie zliczania liczby cyklów Hamiltona w grafie. Mianowicie, przez U oznaczę sobie zbiór wszystkich spacerów rozgałęziających się, wychodzących z jakiegoś terminala t oraz mapujących dokładnie l krawędzi (innymi słowy, drzewo musi mieć l krawędzi, a funkcja przesyłająca musi przesyłać jego korzeń na t). Poza tym, zakładam że w grafie mam jakieś terminale $t_1, t_2, \ldots, t_k$ (gdy

terminal jest tylko jeden, problem znalezienia drzewa Steinera jest dosyć trywialny). Przez A_i oznaczam zbiór wszystkich spacerów rozgałęziających się z U, które mapują jakiś wierzchołek drzewa na terminal t_i.

Zauważmy, że $\bigcap_{i\in[k]} A_i$ da nam zbiór wszystkich spacerów rozgałęziających się, które można „przerobić" do postaci drzewa łączącego wszystkie terminale, a więc jeśli jest niepusty to znaczy to, że istnieje drzewo, które łączy wszystkie terminale i ma l krawędzi (lub mniej, ale to nam w tym problemie nie przeszkadza; chcemy wiedzieć czy istnieje drzewo łączące wszystkie terminale, mające maksymalnie l krawędzi).

Ponadto, żeby żyło się prościej, wprowadzam oznaczenie $S_i = U \setminus A_i$, Innymi słowy, S_i to zbiór wszystkich spacerów rozgałęziających się z U, które nie mapują niczego na t_i. Wykonuję teraz przekształcenia podobne do tych, które robiliśmy przy cyklach Hamiltona:

$$\begin{aligned} |\bigcap_{i\in[k]} A_i| &= |U| - |\bigcup_{i\in[k]} U \setminus A_i| \\ &= |U| - |\bigcup_{i\in[k]} S_i| \\ &= |U| - \sum_{\varnothing\neq X\subset[k]} (-1)^{|X|-1} \cdot |\bigcap_{i\in X} S_i| \end{aligned}$$

Zasadniczo teraz mamy już skończony algorytm obliczania, bo:

1. $|U|$ jesteśmy w stanie obliczyć w czasie wielomianowym, bo to po prostu $dp[t][l]$
2. $|\bigcap_{i\in X} S_i|$ dla danego X również obliczamy w czasie wielomianowym, bo to jest $dp[t][l]$ ale policzone dla grafu indukowanego bez wierzchołków, po których iterujemy się idąc przez X. Innymi słowy, zbiór $S_{i_1} \cap S_{i_2} \cap \cdots \cap S_{i_j}$ jest to zbiór wszystkich spacerów rozgałęziających się w grafie $G[V \setminus \{t_{i_1}, t_{i_2}, \ldots, t_{i_j}\}]$.

4.2 Porządki częściowe, Twierdzenia Dilwortha i Spernera

4.2.1 Twierdzenie Dilwortha

Twierdzenie 4.2.1 (Twierdzenie Dilwortha)**.** Jeśli długość maksymalnego antyłańcucha w posecie wynosi k, poset ten da się pokryć w całości z użyciem k łańcuchów.

Dowód. Robimy indukcję po liczbie elementów posetu; gdy poset P składa się z jednego elementu, twierdzenie zachodzi w trywialny sposób. W dalszej części dowodu, pisząc P będziemy mieli na myśli zbiór, na którym zdefiniowany jest nasz poset (bo formalne poset to tupla).

Załóżmy teraz, że mamy poset zdefiniowany na n elementach. Wiemy, że jego antyłańcuch maksymalny ma długość k. Antyłańcuchów maksymalny spełniający ten warunek nie musi być jeden; oznaczymy zbiór wszystkich antyłańcuchów maksymalnych w tym posecie jako A.

Zdefiniujmy teraz (dla danego antyłańcucha maksymalnego $\alpha \in A$ zbiory U_α i D_α, które będziemy określać odpowiednio jako *upset* i *downset* antyłańcucha α. Do zbioru U_α należą wszystkie elementy P, takie że są (ostro) większe od jakiegokolwiek elementu z α. Do zbioru D_α należą zaś wszystkie elementy P, takie że są mniejsze (ostro) od jakiegokolwiek elementu z α. Bardziej formalnie:

$$U_\alpha = \{\, x \in P \mid \exists_{y\in\alpha}\ y \leq x\} \setminus \alpha$$

$$D_\alpha = \{\, x \in P \mid \exists_{y\in\alpha}\ y \geq x\} \setminus \alpha$$

Pierwsza obserwacja którą należy wykonać, to taka że dla dowolnego $\alpha \in A$ jest tak, że $U_\alpha \cup D_\alpha \cup \alpha = P$. Jest to oczywiste; jeśli istniałby jakiś element z P który nie należałby ani do downsetu α, ani do upsetu α, ani do antyłańcucha maksymalnego α, to z faktu że nie należy ani do downsetu ani do upsetu wynikałoby, że musiałby należeć do antyłańcucha maksymalnego α (bo nie jest porównywalny z żadnym jego elementem).

Druga obserwacja: dla dowolnego $\alpha \in A$ nie istnieje element, który należy jednocześnie do U_α i D_α. Aby dowieść tę obserwację, załóżmy nie wprost, że istnieją jakieś $x, y, z \in P$ takie, że:

1. $y, z \in \alpha$
2. $x \geq y$
3. $x \leq z$

Wówczas otrzymujemy, że $y \leq x \leq z$, a więc z tranzytywności w posetach dostalibyśmy, że $y \leq z$. To prowadziłoby do sprzeczności, bo założyliśmy że $y, z \in \alpha$, a więc znajdują się w jednym antyłańcuchu (i nie mogą być porównywalne).

Udowodnimy teraz szybki lemacik.

Lemat 4.2.1. Następujące warunki są równoważne:

1. Antyłańcuch maksymalny α jest taki, że $D_\alpha = \varnothing$;
2. Antyłańcuch maksymalny α składa się **jedynie** ze wszystkich elementów minimalnych posetu P.

Dowód. 1. (1) $\implies$ (2); stosujemy dowód nie wprost. Załóżmy, że istnieje taki antyłańcuch maksymalny α, że $D_\alpha = \varnothing$, ale do α nie należy jakiś element minimalny z P[1]. Nazwijmy go x. Rozważmy zbiór $\alpha' = \alpha \cup \{x\}$. Jeśli α' jest antyłańcuchem, to znaczy że α nie był antyłańcuchem maksymalnym i otrzymujemy sprzeczność z założeniami. Jeśli α' nie jest antyłańcuchem, to oznacza że istnieje jakieś $y \in \alpha$ takie, że $y \leq x$ lub $y \geq x$.

Nie może być tak, że $y \leq x$, bo x jest elementem minimalnym w P. Jeśli $y \geq x$, to z kolei mamy, że $x \in D_\alpha$, skąd otrzymujemy sprzeczność.

Należy tutaj dodać, że ten dowód nie wprost pokazał jedynie, że α w takim przypadku zawiera wszystkie elementy minimalne z P, ale nie pokazaliśmy że *nie należą do niego* inne elementy z P. Na szczęście, wiedząc że wszystkie elementy minimalne z P znajdują się w α, wiemy że jakikolwiek inny element nie może się tam znaleźć (bo skoro nie jest minimalny to jest porównywalny z jakimś minimum, a więc nie należy do antyłańcucha). To już kończy dowód.

2. (2) $\implies$ (1); element minimalny to taki, że nie istnieje element który byłby od niego mniejszy. Z definicji D_α musi zatem być tak, że $D_\alpha = \varnothing$.

□

Niemal identycznym dowodem można posłużyć się, by dowieść następujący lemat:

Lemat 4.2.2. Następujące warunki są równoważne:

1. Antyłańcuch maksymalny α jest taki, że $U_\alpha = \varnothing$;

[1]Należy również uważać na to, że nie możemy w tym dowodzie założyć *czegokolwiek* o elementach w α – w szczególności *a priori* możliwe jest, że α zawiera jakieś elementy które nie są minimalne w P.

2. Antyłańcuch maksymalny α składa się **jedynie** ze wszystkich elementów maksymalnych posetu P.

Teraz musimy rozpatrzyć trzy przypadki:

1. $\exists_{\alpha \in A}\, U_\alpha = D_\alpha = \varnothing$
   W tym przypadku istnieje antyłańcuch maksymalny, którego upset i downset są puste. Nietrudno pokazać, że jest to jedyny antyłańcuch maksymalny (ale to nie ma znaczenia dla dowodu). Co ma znaczenie dla dowodu to to, że wystarczy z każdego elementu tego antyłańcucha utworzyć jednoelementowy łańcuch zawierający tylko siebie samego. Jako że α ma k elementów, dostajemy podział P na k antyłańcuchów.

2. $\exists_{\alpha \in A}\, U_\alpha \neq \varnothing \wedge D_\alpha \neq \varnothing$
   Rozpatruję sobie posety na zbiorach $B = A \cup U_\alpha$ i $C = A \cup D_\alpha$. Jako, że $U_\alpha \neq \varnothing$ i $D_\alpha \neq \varnothing$, to z pewnością $|B| < |P|$ i $|C| < |P|$. W takim razie, B i C z założenia indukcyjnego da się podzielić na k łańcuchów.

   Ponadto, każdy element α (zarówno w pokryciu łańcuchowym zbioru B, jak i C) należy do łańcucha innego niż jakikolwiek inny element α, jako że 2 elementy z jednego antyłańcucha nie mogą znaleźć się w jednym łańcuchu. W dodatku, z definicji zbiorów U_α i D_α bezpośrednio wynika, że każdy element α jest elementem najmniejszym w odpowiednim łańcuchu z pokrycia łańcuchowego zbioru B, i elementem największym w odpowiednim łańcuchu ze zbioru C. W takim razie po prostu „sklejam" łańcuchy z B i C w danym elemencie z α i mam poprawne pokrycie łańcuchowe całego posetu P.

3. Przypadek przeciwny do dwóch wcześniejszych; $\forall_{\alpha \in A}\, U_\alpha = \varnothing$ **ALBO** $D_\alpha = \varnothing$

   Korzystając z lematów 4.2.1 i 4.2.2, wiemy, że dla każdego $\alpha \in A$ jest tak, że składa się (jedynie) ze wszystkich elementów maksymalnych lub ze wszystkich elementów minimalnych w P. Nawiasem mówiąc, to bezpośrednio implikuje, że w tym przypadku $|A| \leq 2$, ale nie jest to specjalnie ważne.

   Weźmy z P takie x, y, że x jest elementem maksymalnym, a y jest elementem minimalnym w P, przy czym chcemy, by te dwa elementy były porównywalne (tzn. $x \geq y$). Taka para dwóch elementów szczęśliwie zawsze istnieje – jeśli w A istnieje antyłańcuch bez downsetu, to parę tę stanowi dowolne maksimum i jego świadek bycia w upsecie; analogicznie w dualnym przypadku. Rozważmy więc poset $P' = P \setminus \{x, y\}$. Zauważmy, że:

   - $|P'| = |P| - 2$, co pozwala nam zastosować założenie indukcyjne;
   - jako, że każde $\alpha \in A$ zawierało zbiór wszystkich elementów maksymalnych lub minimalnych P, wiemy że długość **wszystkich** antyłańcuchów maksymalnych zmniejszyła się o 1, a więc długość najdłuższego antyłańcucha wynosi $k - 1$.

   To oznacza, że z założenia indukcyjnego P' możemy podzielić na $k - 1$ łańcuchów. Tak więc dodając do P' łańcuch $\{x, y\}$ otrzymujemy podział P na k łańcuchów, a to kończy dowód.

□

4.2.2 Twierdzenie dualne do Dilwortha

Twierdzenie 4.2.2 (Twierdzenie dualne do twierdzenia Dilwortha)**.** Jeśli długość maksymalnego łańcucha w posecie wynosi k, poset ten da się pokryć w całości z użyciem k antyłańcuchów.

Dowód. Zdefiniujmy sobie poset P i funkcję φ idącą z elementów P w liczby naturalne, taką że $\varphi(x)$ jest to moc najdłuższego łańcucha w P, którego maksimum wynosi x. Zauważmy, że φ może przyjmować jedynie wartości w zakresie $1 \dots k$, bo k to długość najdłuższego łańcucha w ogóle. Zauważamy, że wszystkie elementy P które przechodzą na jakąś liczbę m muszą być ze sobą nieporównywalne, a więc formować antyłańcuch. Gdyby tak nie było i istniałyby jakieś elementy x, y, takie że, bez straty ogólności, $x \leq y$ i $\varphi(x) = \varphi(y) = z$, to łańcuch w którym x jest elementem maksymalnym i który ma długość z mógłbym „rozszerzyć" dodając do niego y, które stałoby się nowym elementem maksymalnym; tym samym maksymalna długość łańcucha w którym y byłoby elementem maksymalnym wynosiłaby nie z, a $z + 1$, co prowadziłoby do sprzeczności. W takim razie dla każdej liczby naturalnej w zakresie $1 \dots k$ mam jakiś antyłańcuch i wiem, że te antyłańcuchy w sumie muszą pokrywać cały poset P, co kończy dowód. □

4.2.3 Nierówność LYM

Twierdzenie 4.2.3 (Nierówność LYM (Lubella, Yamamoto, Meshalkina))**.** Przez f_k oznaczmy liczbę elementów k-elementowych w danym antyłańcuchu w kracie B_n. Mamy wówczas:

$$\sum_{k=0}^{n} \frac{f_k}{\binom{n}{k}} \leq 1 \tag{4.4}$$

Dowód. Zdefiniujemy teraz dziwną strukturę, ale obiecuję że ona ma sens. Mamy jakiś antyłańcuch F i definiujemy zbiór C, który dla każdego elementu z antyłańcucha C trzyma wszystkie możliwe pary tego elementu i łańcucha maksymalnego w kracie zbiorów, do którego należy dany element antyłańcucha F. Jeden element z antyłańcucha w zbiorze C może występować w różnych parach (np. jeśli przechodzi przez niego k łańcuchów maksymalnych w kracie zbiorów to wystąpi on k razy).

Ciekawszą obserwacją natomiast jest, że jeden łańcuch maksymalny w zbiorze C może wystąpić maksymalnie raz. Wynika to z faktu, że gdyby wystąpił dwukrotnie, to znaczyłoby że dwa różne elementy z F należą do jednego łańcucha, a więc są ze sobą porównywalne; prowadziłoby to do sprzeczności, bo z założenia te dwa elementy mają ze sobą nie być porównywalne, jako że należą do jednego antyłańcucha.

Warto zauważyć, że liczba łańcuchów maksymalnych w kracie zbiorów B_n wynosi $n!$. Jest to dosyć prosta do udowodnienia obserwacja – bierzemy sobie najwyższy element; elementów o mocy od niego mniejszej o 1 (tzn. takich że są „poziom niżej") zawierających się w nim jest dokładnie n (bo na n sposobów mogę „wyrzucić" z niego jakiś element tak, by powstał jego podzbiór mający moc mniejszą od niego o 1). Gdy mam zbiór mający $n-1$ elementów, mogę uzyskać podzbiór na $n-2$ elementowy na $n-1$ sposobów, i tak dalej. „Wyrzucanie" elementów w ten sposób aż nie otrzymamy zbioru pustego tworzy zawsze jakiś łańcuch maksymalny; oczywistym jest, że łańcuchy te są różne jak i to, że w ten sposób otrzymujemy wszystkie możliwe łańcuchy maksymalne.

Oznacza to, że elementów w zbiorze C jest maksymalnie $n!$, bo każdy łańcuch maksymalny występuje maksymalnie raz. Jednocześnie okazuje się, że możemy zliczyć ile dokładnie razy w zbiorze C występuje dany element z antyłańcucha F. Weźmy sobie element $x \in F$; zauważmy, że przechodzi przez niego dokładnie $|x|! \cdot (n - |x|)!$ łańcuchów maksymalnych. Żeby to pokazać, stosujemy tę samą obserwację co powyżej, ale jakby „zawężamy" kratę zbiorów do tego elementu, tzn. bierzemy taki jej podzbiór że mamy inną kratę zbiorów, gdzie x jest „na górze", a więc łańcuchów które idą „od dołu" do x jest $x!$. Podobne rozumowanie stosujemy

by pokazać, że łańcuchów idących od x „na górę" jest $(n - |x|)!$, a więc wszystkich łańcuchów maksymalnych idących przez x jest dokładnie $|x|! \cdot (n - |x|)!$. Mamy więc, że

$$\sum_{x \in F} |x|! \cdot (n - |x|)! = |C| \leq |n!|$$

A zatem

$$\sum_{x \in F} |x|! \cdot (n - |x|)! \leq |n!|$$

$$\sum_{x \in F} \frac{|x|! \cdot (n - |x|)!}{n!} \leq 1$$

$$\sum_{x \in F} \frac{1}{\binom{n}{|x|}} \leq 1$$

Teraz żeby było ładniej mówimy, że f_k to jest liczba elementów F takich, że mają moc równą dokładnie k, skąd otrzymujemy już

$$\sum_{k=0}^{n} \frac{f_k}{\binom{n}{k}} \leq 1$$

□

4.2.4 Twierdzenie Spernera

Twierdzenie 4.2.4 (Twierdzenie Spernera)**.** Najdłuższy antyłańcuch F w kracie zbiorów B_n ma moc $\binom{n}{\lfloor \frac{n}{2} \rfloor} = \binom{n}{\lceil \frac{n}{2} \rceil}$

Dowód. Jesteśmy w stanie wskazać antyłańcuch takiej długości; jest to po prostu antyłańcuch w „warstwie" $\lfloor n/2 \rfloor$ kraty B_n, tzn. wszystkie podzbiory $[n]$ mocy $\lfloor n/2 \rfloor$. Pokazujemy teraz, że nie istnieje dłuższy antyłańcuch, korzystając z faktu, że:

$$\binom{n}{k} \leq \binom{n}{\lfloor \frac{n}{2} \rfloor} = \binom{n}{\lceil \frac{n}{2} \rceil}$$

a więc z nierówności LYM mamy, że:

$$\sum_{k=0}^{n} \frac{f_k}{\binom{n}{\lfloor n/2 \rfloor}} \leq \sum_{k=0}^{n} \frac{f_k}{\binom{n}{k}} \leq 1$$

co dowodzi tezę bo w takim razie:

$$\sum_{k=0}^{n} \frac{f_k}{\binom{n}{\lfloor n/2 \rfloor}} = |F| \cdot \frac{1}{\binom{n}{\lfloor n/2 \rfloor}} \leq 1$$

więc

$$|F| \leq \binom{n}{\lfloor n/2 \rfloor} = \binom{n}{\lfloor n/2 \rfloor}$$

□

4.3 Twierdzenie Ramseya. Przykłady zastosowań

4.3.1 Definicja

4.3.2 Dowód twierdzenia

> W twierdzeniach ramseyowych nie chodzi o jakieś liczby, a o granice Twojej wyobraźni
>
> *Student TCSu który uwalił egzamin*

Twierdzenie 4.3.1. Liczby

$$R^{(k)}(l_1, l_2, l_3, \ldots, l_s; s)$$

są zdefiniowane poprawnie.

Dowód. Prowadzimy indukcję po k, s i $\sum_{i=1}^{s} l_i$. Sprawdzamy przypadki bazowe:

1. Gdy $k = 1$ poprawność wynika z zasady szufladkowej, $N = (l_1 - 1) + (l_2 - 1) + \cdots + (l_s - 1) + 1$,
2. Gdy $s = 1$ chyba wszyscy widzimy dlaczego to jest trywialne,
3. Trzeci przypadek bazowy polega na tym, że dla jakiegoś j $l_j = k$ (k jest to minimalna wartość którą w ogóle może przyjąć jakiekolwiek l_i, inaczej to by nie miało sensu). W takim razie jest tak, że $R^{(k)}(l_1, l_2, l_3, \ldots, l_j, \ldots, l_s; s) = R^{(k)}(l_1, l_2, l_3, \ldots, l_{j-1}, l_{j+1}, \ldots, l_s; s - 1)$. Wynika to z faktu, że jeśli pokolorujemy jakikolwiek podzbiór k-elementowy na kolor j to wtedy to kolorowanie już jest „załatwione", a więc rozpatrujemy przypadek złośliwego kolorowania gdzie kolor j jest po prostu nieużywany. Legendy mówią, że ktoś fakt ten kiedyś dowiódł formalnie.

Zostajemy obecnie z przypadkiem, gdzie mamy jakieś $k, s \geq 2$ i dla każdego i jest tak, że $l_i > k$. Wprowadzamy oznaczenie:

$$L_i = R^{(k)}(l_1, l_2, l_3, \ldots, l_{i-1}, l_i - 1, l_{i+1}, \ldots l_s; s)$$

I zauważamy, że L_i z założenia indukcyjnego (bo zredukowaliśmy sumę l_i; formaliści mogą sobie podumać nad indukcją po wielu zmiennych i jak działa) jest zdefiniowane poprawnie (w sensie jest jakąś liczbą naturalną). Teraz definiujemy sobie pewną *potężną* liczb służącą jako ograniczenie górne:

$$N = R^{(k-1)}(L_1, L_2, L_3, \ldots, L_s; s) + 1$$

Ponownie, jest ona poprawnie zdefiniowana z założenia indukcyjnego (bo kolorujemy teraz podzbiory $k-1$-elementowe). Po co ta jedynka na końcu? Zaraz się okaże. Przez c oznaczajmy jakieś kolorowanie podzbiorów k-elementowych $[N]$ i zdefiniujmy sobie teraz kolorowanie c', które koloruje podzbiory $k-1$-elementowe $[N]$. c' definiujemy sobie w oparciu o c w niezwykle fascynujący sposób. Wybieramy sobie jakiś wierzchołek x z $[N]$ i mówimy, że kolorowanie c' koloruje jakikolwiek $k-1$-elementowy podzbiór $[N] \setminus \{x\}$ (który nazwę a) na taki sam kolor, na który kolorowanie c pokolorowałoby $a \cup \{x\}$. Pomocne może być tutaj narysowanie tej sytuacji.

W każdym razie, wiemy że mamy jakiś zbiór L_j elementów z N, taki że mamy L_1 lub L_2 lub $\ldots$ lub L_s punktów takich, że każdy ich $k-1$-elementowy podzbiór koloruje się na ten sam kolor (w kolorowaniu c').

A skoro mamy zbiór L_j elementów, to z definicji L_j wiemy, że istnieje tu podzbiór l_1 lub l_2 lub $\ldots$ lub $l_j - 1$ lub $\ldots$ lub l_s elementów taki, że każdy ich k-elementowy podzbiór ma ten sam kolor (w kolorowaniu c). Zauważam że jeśli własność ta zachodzi dla jakiegokolwiek l_i gdzie $i \neq j$, to ta własność nam się przez przypadek właśnie udowodniła (i nawet nie użyliśmy naszego śmiesznego kolorowania). Zostaje przypadek gdy mamy $l_j - 1$ punktów takich, że ich każdy k-elementowy podzbiór jest pokolorowany na kolor j.

Pamiętacie kolorowanie c'? Teraz ono wchodzi do akcji, bo nasze L_j musiało spełniać, że dowolny $k-1$-elementowy podzbiór L_j (nazwijmy go ponownie a) ma ten sam kolor co $a \cup \{x\}$ w kolorowaniu c. Czyli skoro w tym L_j mam $l_j - 1$ punktów, że każdy ich k-elementowy podzbiór ma kolor j w c, to jak dorzucę x do tego podzbioru to mam już zbiór l_j-elementowy i nadal wszystko jest kolorowane tak jak bym chciał żeby było. Sparse'owanie tego co się stało może trochę zająć, ale w sumie to udowodniliśmy twierdzenie Ramseya. Fajnie. □

4.3.3 Twierdzenie Erdősa-Szekeresa

Twierdzenie 4.3.2 (Erdősa-Szekeresa)**.** Dla dowolnego k istnieje takie N, że jeśli N punktów na płaszczyźnie znajduje się w pozycji ogólnej to jakieś k z nich znajduje się w pozycji wypukłej.

Dowód. Po pierwsze musimy zauważyć, że zbiór k punktów jest w pozycji wypukłej wtedy i tylko wtedy gdy dowolny jego podzbiór 4-elementowy również jest w pozycji wypukłej. Wynika to z faktu, że jeśli k punktów nie jest w pozycji wypukłej to mogę striangulować i wskazać 4 takie, że również nie są w pozycji wypukłej, a jeśli k punktów jest w pozycji wypukłej to oczywiste jest, że dowolny ich podzbiór również jest w takiej pozycji.

Po drugie, dla dowolnych 5 punktów jest tak że wśród nich znajdziemy co najmniej 4 takie, że są w pozycji wypukłej. Dowodzimy to w ten sposób, że sobie je rysujemy i bierzemy otoczkę wypukłą; zakładamy że ma ona 3 punkty, bo inaczej teza już się udowodni, po czym puszczamy linię prostą przez 2 punkty które są wewnątrz trójkąta stanowiącego otoczkę wypukłą. Po jednej stronie tej prostej są jakieś 2 punkty z otoczki, które razem z tymi dwoma punktami przez które przeszła prosta są pozycji ogólnej. Fajnie byłoby chyba to rozrysować żeby dowieść.

No i teraz już po prostu robimy sobie ogólnego Ramseya $R^{(4)}(k, 5; 2)$ – kolorujemy czwórki punktów na pierwszy kolor jeśli są w pozycji wypukłej, na drugi jeśli nie. Ponieważ nie ma 5 takich punktów że każdy ich 4-elementowy podzbiór nie jest w pozycji wypukłej (druga obserwacja) to musi być k takich punktów że ich każdy 4-elementowy podzbiór znajduje się w pozycji wypukłej, co daje nam tezę na mocy pierwszej obserwacji. □

4.4 Funkcje tworzące. Wyznaczanie liczb Fibonacciego za pomocą funkcji tworzących

4.4.1 Rozwiązywanie rekurencji liniowych

> I can elaborate: zrobiłam zadanka, zobaczyłam tworzące, stwierdziłam, że chce mi się spać, poszłam sobie
>
> *Studentka TCSu o zadaniach z funkcji tworzących na kolokwium*

4.4.2 Rozkład na ułamki proste

To nie jest formalny dowód ani formalna własność ani nic, bardziej schemat postępowania przy rozkładzie na ułamki proste. Sam dowód tego, że rozkład na ułamki proste istnieje, to *sprowadź do wspólnego mianownika i zobacz co Ci wyszło*. Jeżeli $deg(P(x)) < deg(Q(x))$ i $Q(x) = (x-a)^n \cdot (x-b)^k$ to:

$$\frac{P(x)}{Q(x)} = \frac{P(x)}{(x-a)^n \cdot (x-b)^k} = \frac{A_1}{x-a} + \frac{A_2}{(x-a)^2} + \cdots + \frac{A_n}{(x-a)^n} + \frac{B_1}{x-b} + \frac{B_2}{(x-b)^2} + \cdots + \frac{B_k}{(x-b)^k}$$

Oczywiście ten schemat można rozszerzać na więcej śmiesznych rzeczy w mianowniku, ale chyba widać o co chodzi.

4.4.3 Wzór Bineta

Twierdzenie 4.4.1 (Wzór Bineta)**.**

$$f_n = \frac{1}{\sqrt{5}} \cdot \left(\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right) \tag{4.5}$$

Dowód. Rozpisujemy sobie funkcję tworzącą ciągu f_n:

$$\begin{aligned}
F(x) &= f_0 + f_1 \cdot x + f_2 \cdot x^2 + f_3 \cdot x^3 \cdots = \\
&= f_0 + f_1 \cdot x + (f_0 + f_1) \cdot x^2 + (f_1 + f_2) \cdot x^3 + \cdots = \\
&= f_0 + f_1 \cdot x + f_0 \cdot x^2 + f_1 \cdot x^2 + f_1 \cdot x^3 + f_2 \cdot x^3 + \cdots = \\
&= f_0 + f_1 \cdot x + f_0 \cdot x^2 + f_1 \cdot x^3 + \cdots + f_1 \cdot x^2 + f_2 \cdot x^3 + \cdots = \\
&= f_0 + f_1 \cdot x + x^2 \cdot (f_0 + f_1 \cdot x + \dots) + x \cdot (f_1 \cdot x + f_2 \cdot x^2 + \dots) = \\
&= f_0 + f_1 \cdot x + x^2 \cdot F(x) + x \cdot (F(x) - f_0) = \\
&= 0 + 1 \cdot x + x^2 \cdot F(x) + x \cdot (F(x) - 0) = \\
&= x + x^2 \cdot F(x) + x \cdot F(x)
\end{aligned}$$

W takim razie mamy, że:

$$\begin{aligned}
F(x) &= x + x^2 \cdot F(x) + x \cdot F(x) \\
F(x) - x^2 \cdot F(x) - x \cdot F(x) &= x \\
F(x) \cdot (1 - x^2 - x) &= x \\
F(x) &= \frac{x}{-x^2 - x + 1}
\end{aligned}$$

Mianownik możemy rozbić (za pomocą liczenia jakichś delt czy coś):

$$F(x) = \frac{x}{(-1) \cdot \left(x - \left(-\frac{1+\sqrt{5}}{2} \right) \right) \cdot \left(x - \left(-\frac{1-\sqrt{5}}{2} \right) \right)}$$

Nie no, serio, jeśli ktoś myśli że będę TeXować te przekształcenia to się myli. Powinno wyjść po przekształceniach że:

$$F(x) = \frac{x}{(1-ax) \cdot (1-bx)}$$

gdzie $a = \frac{1+\sqrt{5}}{2}, b = \frac{1-\sqrt{5}}{2}$

Dalej rozbijamy na ułamki proste:

$$F(x) = \frac{A}{1-ax} + \frac{B}{1-bx}$$

A powinno wyjść $\frac{1}{\sqrt{5}}$, B powinno wyjść $-\frac{1}{\sqrt{5}}$.

Odwijamy każdą z tych funkcji tworzących z osobna, korzystając ze wzoru podanego we wcześniejszym rozdziale i otrzymujemy wzór. □

4.5 Skojarzenia w grafach dwudzielnych. Twierdzenie Halla

4.5.1 Twierdzenie Halla

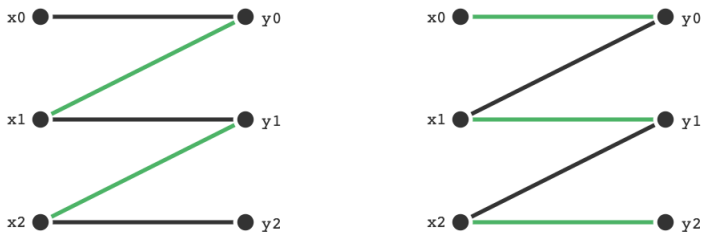
> Dlaczego wysoki odsetek pracowników służby drogowej ma rodziny? Bo dużo Hall'ują.
>
> *Niezwykle suchy żart pewnego studenta*

Twierdzenie 4.5.1 (Halla)**.** Graf dwudzielny $G = (X, Y, E)$, gdzie $|X| = |Y|$ ma dopasowanie doskonałe wtedy i tylko wtedy, gdy dla dowolnego $A \subset X$ zachodzi:

$$|A| \leq |N(A)| \tag{4.6}$$

Dowód. Ponieważ twierdzenie mówi *wtedy i tylko wtedy*, musimy udowadniać w dwie strony. Zacznijmy od tej prostszej strony, czyli pokażmy że gdy graf dwudzielny w którym $|X| = |Y|$ ma dopasowanie doskonałe to $|A| \leq |N(A)|$. Zasadniczo od razu to widać, bo skoro dopasowanie doskonałe istnieje to wystarczy sobie je wziąć. Każdy wierzchołek z $|X|$ ma wówczas jakąś krawędź do wierzchołka z $|Y|$. Zauważamy, że siłą rzeczy w samym dopasowaniu (czyli jakimś podgrafie oryginalnego grafu dwudzielnego, z którego być może „wywalono" jakieś krawędzie) jest tak, że $|A| = |N(A)|$, z czego w szczególności wynika teza. To chyba widać.

W drugą stronę jest ciekawiej, bo po pierwsze musimy sobie wprowadzić instytucję *ścieżki powiększającej*. Nie należy tego mylić ze ścieżką powiększającą z przepływów, bo one mówią o innych rzeczach (ale idea jest ta sama). Generalnie to załóżmy sobie, że mam jakieś dopasowanie M które nie jest doskonałe. Oznacza to, że w X jest jakiś wierzchołek x_0 poza dopasowaniem. Jeśli x_0 łączy się z jakimś wierzchołkiem $y_0 \in Y$ i $y_0 \in M$. y łączy się z jakimś $x_1 \in M$ (bo są razem w dopasowaniu). Teraz jeśli x_1 łączy się z jakimś $y_1 \in Y$ takim, że $y_1 \notin M$ to ja to dopasowanie mogę przerobić: „połączyć" x z y i x_1 z y_1, dorzucając dodatkowy wierzchołek do dopasowania. To jest przykład bardzo krótkiej ścieżki powiększającej, ale generalna idea to jest taki „zygzak" którego można przerobić, żeby dopasowanie powiększyło się o jeden wierzchołek. At this point wszyscy już chyba wiedzą, że zamiłowania do formalizmu to ja nie mam.



Rysunek 4.3: Ścieżka powiększająca przed i po zamianie krawędzi

No dobra, ale co ma ścieżka powiększająca do twierdzenia Halla? Okazuje się że jest ona bardzo wygodnym narzędziem.

Załóżmy sobie nie wprost, że mamy jakiś graf dwudzielny $G = (X, Y, E)$, w którym zachodzi warunek Halla ale nie ma dopasowania doskonałego. W takim razie weźmy dopasowanie maksymalne M. Istnieje jakiś x, który nie należy do tego dopasowania (bo nie jest doskonałe). Z warunku Halla ($|A| \leq |N(A)|$ dla dowolnego $A \subset X$) mam, że x musi mieć jakiegoś sąsiada w Y. Zbiór wszystkich wierzchołków, z którymi połączony jest x (być może jest ich więcej, być może tylko jeden) oznaczam jako B_0. Każdy wierzchołek z B_0 musi należeć do dopasowania M (bo inaczej mógłbym je rozszerzyć, biorąc krawędź między tym wierzchołkiem a x). Wszystkie wierzchołki z X które są razem w dopasowaniu z wierzchołkami z B_0 oznaczam jako A_1. Oczywiście $|A_1| = |B_0|$. Zauważmy, że $|A_1 \cup \{x\}| \geq |B_0|$, a zatem musi istnieć jakiś zbiór wierzchołków B_1 który ma krawędzie do wierzchołków zbioru A_1. Ponownie, wszystkie krawędzie z B_1 muszą być w dopasowaniu, bo inaczej mielibyśmy ścieżkę powiększającą (aha!) od x do jakiegoś wierzchołka z B_1. W takim razie mamy jakiś zbiór A_2 wierzchołków które są w dopasowaniu z wierzchołkami z B_1, ponownie $|A_2| = |B_1|$. $|A_2| + |A_1| + 1 \geq |B_0| + |B_1|$, skąd wierzchołki z A_2 łączą się jeszcze z jakimiś innymi wierzchołkami z Y, ich zbiór nazwiemy B_2. I ponownie, wierzchołki z B_2 muszą być w dopasowaniu, bo inaczej mielibyśmy ścieżkę powiększającą. Korzystamy teraz z faktu który zawsze zakładamy, tj. faktu że grafy są skończone.

I tak dalej, aż do wyczerpania zasobów

Stefan „Siara" Siarzewski do senatora Ferdynanda Lipskiego, „Kilerów-ów 2-óch"

Nietrudno bowiem zauważyć, że w końcu wierzchołki się skończą i albo dostaniemy sprzeczność z założeniem że warunek Halla zachodzi, albo wyjdzie nam ścieżka powiększająca (a dopasowanie miało być maksymalne). Tym samym kończymy dowód. □

4.6 Kolorowanie grafów, twierdzenia Brooksa

Twierdzenie 4.6.1 (Brooks)**.** Jeśli spójny graf G jest kliką lub cyklem nieparzystym to $\chi(G) = \Delta(G) + 1$. W przeciwnym razie $\chi(G) \leq \Delta(G)$

Dowód. Widzimy, że cykl nieparzysty wymaga użycia $3 = \Delta(G) + 1$ kolorów, a w przypadku

kliki sąsiedzi każdego wierzchołka używają $\Delta(G)$ kolorów, a jeszcze jeden potrzebujemy na ten wierzchołek. Przyjmijmy zatem, że nasz graf G nie jest ani cyklem nieparzystym ani kliką.

Jeśli $\Delta(G) \leq 2$ to G jest ścieżką lub cyklem parzystym i widzimy, że G jest dwudzielny.

Niech $\Delta(G) \geq 3$.

Idea dowodu jest taka, że będziemy chcieli jakoś skonstruować kolorowanie używające co najwyżej $\Delta(G)$ kolorów. W związku z tym będziemy inkrementalnie odfiltrowywać grafy, dla których takie kolorowanie stworzymy. Przedstawiam zatem kolejne własności grafu, dla którego będzie się trzeba trochę bardziej nameczyć.

1. G jest Δ-regularny Pokażemy, że w przeciwnym razie $col(G) \leq \Delta$ Jeśli G nie jest Δ-regularny to w G istnieje wierzchołek v o stopniu mniejszym niż Δ. Postawmy ten wierzchołek na końcu permutacji i spójrzmy na graf $G - v$. v miał jakichś sąsiadów, których stopień wynosił co najwyżej Δ. W takim razie po usunięciu v jego byli sąsiedzi na pewno mają teraz stopień mniejszy niż Δ i możemy powtórzyć całe to rozumowanie aż skończą nam się wierzchołki i wygenerujemy całą permutację. Zauważamy, że dzięki konstrukcji każdy wierzchołek ma na lewo mniej niż Δ sąsiadów i dostajemy $col(G) \leq \Delta$

   Wybierzmy dowolny wierzchołek v i oznaczmy $H = G - v$. Sąsiadom v zmniejszyliśmy stopień, zatem z powyższego wywodu wynika, że H jest Δ-kolorowalny. Pokolorujmy zatem H i przejdźmy do kolejnej własności.

2. Sąsiedzi v dostają parami różne kolory w kolorowaniu grafu H W przeciwnym razie któryś z Δ kolorów jest wolny w v i możemy go użyć kończąc kolorowanie.

   Nazwijmy sąsiadów v przez $v_1, ..., v_\Delta$ i niech będą pokolorowani kolorami $1, ..., \Delta$.

   Oznaczmy $C_{ij} = H[\{v \in V \mid c(v) \in \{i, j\}]$ - podgraf indukowany grafu H, który zawiera wszystkie wierzchołki w kolorach i, j

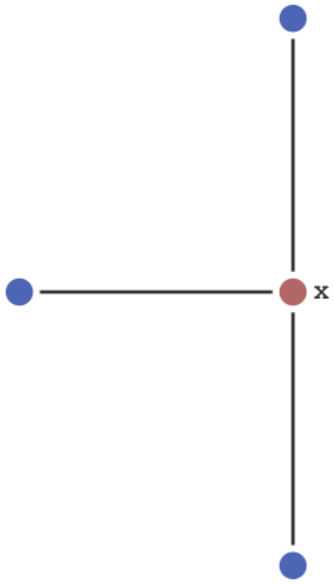
3. Sąsiedzi v_i, v_j leżą w tym samym komponencie grafu C_{ij} W przeciwnym razie możemy wziąć komponent do którego należy v_i i przekolorować go tak, że wierzchołki o kolorze i dostają kolor j, a te o kolorze j dostają kolor i. Tym samym wierzchołki v_i, v_j otrzymują oba kolor j sprowadzając problem do poprzedniego podpunktu.

4. Każdy C_{ij} jest ścieżką. Załóżmy że tak nie jest i weźmy pierwszy licząc od v_i wierzchołek, który ma co najmniej trzech sąsiadów w C_{ij} i nazwijmy go x. Bez straty ogólności powiedzmy, że x ma kolor i. Rozważmy kolory jakie mają sąsiedzi x. Co najmniej trzech sąsiadów ma kolor j, a pozostałych jest $\Delta - 3$ W takim razie sąsiedzi x używają co najwyżej $\Delta - 2$ różnych kolorów. Oczywiście jeden z wolnych kolorów to i, ale możemy teraz przekolorować x na ten drugi. Ponieważ x był pierwszym rozgałęzieniem między v_i a v_j to po przekolorowaniu v_i oraz v_j muszą leżeć w różnych komponentach nowego C_{ij}

5. Każde dwa $C_{ij}, C_{ik}, k \neq j$ przecinają się tylko w v_i W przeciwnym razie istnieje wierzchołek x w kolorze i, który ma dwóch sąsiadów w kolorze j i dwóch sąsiadów w kolorze k. Jak się dobrze policzy to tak jak poprzednio wyjdzie nam, że jakiś kolor jest wolny i możemy zrobić ten sam myk z przekolorowaniem.

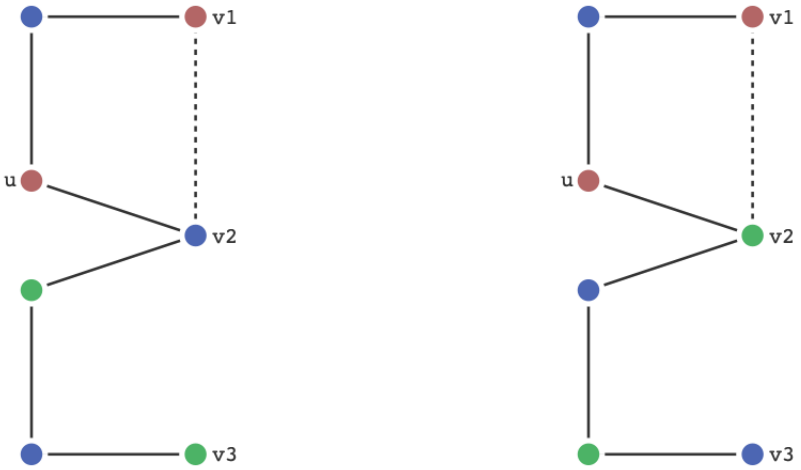
6. Istnieje para v_i, v_j, która nie jest połączoną krawędzią. W przeciwnym razie wierzchołki $v, v_1, ..., v_\Delta$ tworzą klikę, co jest sprzeczne z założeniem.

   Bez straty ogólności powiedzmy, że v_1 i v_2 nie są połączone krawędzią. Jednak z własności (5) musi istnieć ścieżka między nimi. Niech więc u to będzie pierwszy wierzchołek w kolorze 1 na ścieżce od v_2 do v_1.



Rysunek 4.4: Wierzchołek x ma trzech sąsiadów w kolorze j

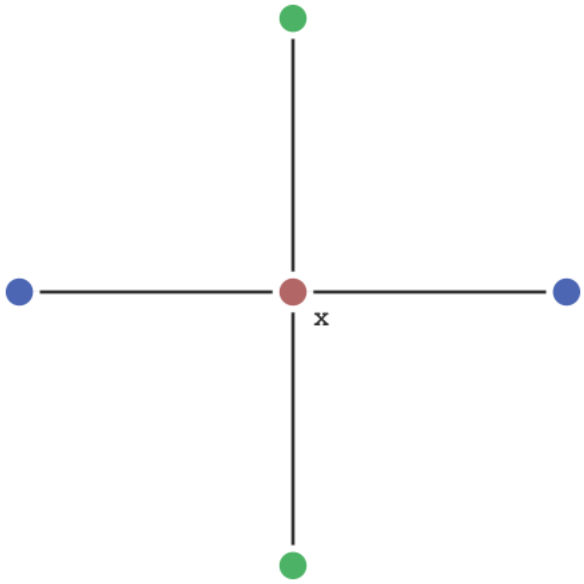
Teraz dzieje się magia. W C_{23} zamieniamy kolory 2 i 3 i takie kolorowanie przepuszczamy przez warunki (2)−(5). Jeśli w którymś miejscu udało nam się stworzyć dobre kolorowanie to super, a jeśli nie to ups. Na szczęście zauważamy teraz fajną rzecz. Otóż wierzchołek u nadal jest połączony ścieżką w kolorach 1 i 2 z wierzchołkiem v_1, zatem należy do komponentu C_{12}, ale z drugiej strony jest połączony krawędzią z wierzchołkiem v_2, który ma teraz kolor 3 zatem należy też do komponentu C_{13}. W takim razie nowe kolorowanie narusza warunek (5) co już umiemy rozwiązać.



Rysunek 4.6: Wierzchołki v_1 i v_2 przed i po przekolorowaniu komponentu C_{23}

To tyle, nie ma więcej warunków, które musimy rozważać. Fajnie.

□



Rysunek 4.5: Wierzchołek x ma dwóch sąsiadów w kolorze j i dwóch w kolorze k

4.7 Liczba chromatyczna a liczba kolorująca grafów

4.7.1 Liczba kolorująca

Liczba kolorująca jest źródłem konfuzji dla niezliczonych studentów. Co ona w ogóle robi, po co jest i dlaczego nie ma jej w żadnej literaturze? Na to ostatnie pytanie nie odpowiem, ale liczba kolorująca, oznaczana również jako $col(G)$ jest użyteczna, bo, jak za niedługo pokażemy, ogranicza od góry $\chi(G)$ oraz da się ją obliczyć w czasie wielomianowym (na wypadek gdyby ktoś się jeszcze nie zorientował, kolorowanie grafów jest problemem klasy NP-przykrej). Poza tym jestem zdania, że nazwanie tego *liczba kolorująca* to fatalna decyzja, bo dosyć ładnie kamufluje co to w ogóle jest; w tym momencie chciałbym napisać coś śmiesznego o matematykach i dziwnych decyzjach, ale nic zabawnego nie przychodzi mi do głowy.

4.7.1.1 Definicja

Rozpatrzmy wszystkie permutacje wierzchołków grafu G. Zdefiniujmy funkcję f, która dla każdego wierzchołka w obrębie danej permutacji przypisuje liczbę jego sąsiadów, którzy wystąpili „przed nim" w permutacji. Zdefiniujmy funkcję g, która dla danej permutacji wynosi maksymalnej wartości funkcji f policzonej dla wszystkich wierzchołków z tej permutacji i dodaje do tego jeden. Skąd wytrzasnęła się ta jedynka zaraz się dowiemy, obiecuję że to nie jest kolejny arbitralny wymysł matematyków którzy wypili za dużo kawy (tak jak nazwa tego bytu).

Fajnie teraz zauważyć, że g dla danej permutacji oszacowuje z góry jak bardzo może „skopać" kolorowanie algorytm First-Fit, jeśli pokoloruje wierzchołki zgodnie z tą permutacją; w najgorszym przypadku, gdy istnieje jakiś wierzchołek mający d sąsiadów „na lewo" i wszyscy mają różne kolory, to FF da mu kolor $d+1$. Stąd maksymalna liczba sąsiadów na lewo wśród wierzchołków $(+1)$ daje nam górne oszacowanie na to, jak First-Fit może popsuć kolorowanie *jeśli będzie kolorować według tej permutacji.*

Nikogo teraz chyba nie zdziwi fakt, że $col(G)$ to jest po prostu minimum po funkcjach g dla wszystkich permutacji wierzchołków grafu G.

4.7.2 Relacja z liczbą chromatyczną

4.7.2.1 Ograniczenie liczby chromatycznej przez liczbę kolorującą

Twierdzenie 4.7.1 (Relacja liczby kolorującej z liczbą chromatyczną)**.**

$$\chi(G) \leq col(G) \tag{4.7}$$

Dowód. Gdy tak nie było, to istniałaby taka permutacja wierzchołków grafu że First-Fit pokolorowałby graf lepiej niż $\chi(G)$ mówi, że da się pokolorować. Koniec dowodu. Serio. □

4.7.2.2 Ograniczenie liczby kolorującej przez jakąś funkcję liczby chromatycznej

Nie da się. Znaczy da się, ale tylko w pewnych klasach grafów. W ogólnej klasie grafów istnieją takie grafy, że liczba kolorująca leci do nieskończoności, a $\chi(G) = 2$. Grafem takim jest na przykład klika dwudzielna $K_{n,n}$. Swoją drogą to jest protip: jeśli potrzebujesz udowodnić że $col(G)$ nie da się ograniczyć w jakiejś klasie grafów przez funkcję $\chi(G)$, spróbuj pokazać że da się tam skonstruować klikę dwudzielną. Nie ma za co.

4.7.3 Algorytm obliczania liczby kolorującej

Twierdzenie 4.7.2 (Magiczny wzór na liczbę kolorującą)**.**

$$col(G) = \max\{\delta(H) : H \subset_{ind.} G\} + 1 \tag{4.8}$$

Dowód. Ten wzór wygląda na początku dziwnie, ale w sumie ma sens. Dowodzić to będziemy trochę dziwnie, bo zamiast pokazywać od razu równość, pokażemy ograniczenie z góry i z dołu (skąd będziemy mieć, że istotnie zachodzi równość).

Pokażmy zatem nierówność w pierwszą stronę:

$$col(G) \geq \max\{\delta(H) : H \subset_{ind.} G\} + 1$$

Dowód jest dosyć prosty; jeśli $col(G) = k$ dla jakiegoś k, to znaczy to że istnieje jakaś permutacja wierzchołków G taka, że każdy wierzchołek „na lewo" ma co najwyżej $k-1$ sąsiadów. Teraz dla każdego podgrafu indukowanego H jest tak, że gdzieś w tej permutacji jest „ostatni" wierzchołek należący do tego podgrafu indukowanego. Nie ma on w ogóle krawędzi „na prawo" i ma same krawędzie „na lewo". Minimalnie może ich mieć $\delta(H)$, czyli w takim razie $\delta(H) \leq k-1$, skąd mamy że $k \geq \delta(H) + 1$ dla dowolnego podgrafu indukowanego (skąd mamy już tezę).

W drugą stronę dowód przy okazji pokazuje nam wielomianowy algorytm obliczania $col(G)$. Nie ukrywam że jest on bardzo fajny.

$$col(G) \leq \max\{\delta(H) : H \subset_{ind.} G\} + 1$$

Konstruujemy sobie permutację wierzchołków grafu G. W jaki sposób? Bierzemy wierzchołek o najmniejszym stopniu i wrzucamy go *na koniec* permutacji. Następnie rozpatrujemy podgraf indukowany H, bez tamtego wierzchołka o najmniejszym stopniu. H znowu ma jakiś wierzchołek o minimalnym stopniu, więc dorzucam go na koniec permutacji (przed wcześniejszym wierzchołkiem o minimalnym stopniu) i kontynuuję „obgryzanie". Nietrudno zauważyć, że każdy wierzchołek ma „na lewo" od siebie jakieś $\delta(H)$ sąsiadów (gdzie H jest jakimś podgrafem indukowanym). Tym samym col jest z pewnością mniejszy lub równy niż maksymalny minimalny stopień wierzchołka w jakimś podgrafie indukowanym (+1).

Pokazaliśmy więc, że nasz „algorytm" generuje optymalną permutację, bo pokazaliśmy wcześniej że lepiej się nie da (pokazując ograniczenie dolne na $col(G)$). Jednocześnie dowodzi to równości. □

4.8 Kolorowania krawędziowe grafów. Twierdzenie Vizinga

4.8.1 Twierdzenie Vizinga

Jest to dowód, który najłatwiej zrozumieć samemu rozrysowując sobie proces na kartce. Niemniej, z pomocą rysunków spróbuję go Wam przybliżyć.

Twierdzenie 4.8.1 (Vizing)**.**

$$\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1$$

Dowód. Ograniczenie dolne widzimy od razu – Δ krawędzi spotykających się w jednym wierzchołku musi dostać różne kolory.

Ograniczenie górne pokazujemy indukcją po liczbie pokolorowanych krawędzi. Jedną krawędź umiemy pokolorować bez problemu. Weźmy zatem częściowe kolorowanie i powiedzmy, że chcemy pokolorować krawędź (x, y).

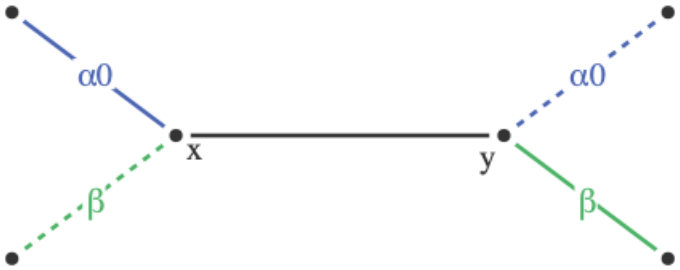
Skoro mamy do dyspozycji $\Delta + 1$ kolorów to znaczy, że każdy wierzchołek ma jakiś kolor wolny (nie wychodzi z niego krawędź w tym kolorze).

Obserwacja, z której będziemy dużo korzystać: jeśli dowolne wierzchołki x, y połączone krawędzią mają wolny ten sam kolor β to krawędź między nimi możemy pokolorować na tenże kolor. Kolory wolne dla danego wierzchołka będziemy oznaczać linią przerywaną. dotykającą tego wierzchołka.

Załóżmy więc, że mamy pecha i wierzchołki x, y nie mają wspólnych wolnych kolorów tj. jeśli x ma wolny kolor β to y ma ten kolor zajęty i vice versa. Dodatkowo niech α_0 będzie wolnym kolorem wierzchołka y.

Od tego momentu będziemy tak kombinować, żeby x zwolnić α_0, być może zajmując przy tym β.

Rysunek 4.7: prosty przypadek; x i y mają wolny ten sam kolor β

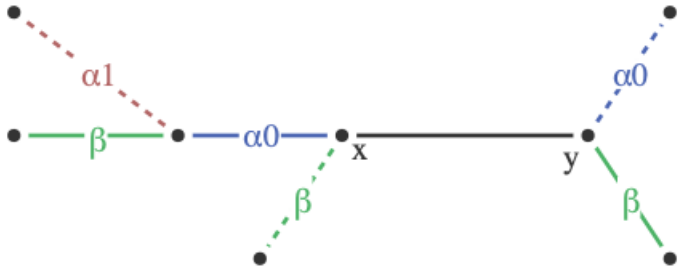


Rysunek 4.8: x i y nie mają wspólnych wolnych kolorów

Niech x_0 będzie taki, że krawędź (x, x_0) ma kolor α_0. Jeśli x_0 ma wolny kolor β to krawędź

(x, x_0) możemy przekolorować na kolor β, tym samym sprawiając, że x ma wolne α_0. Ale w takiej sytuacji możemy pokolorować (x, y) na kolor α_0.

Zatem sytuacja ma się teraz tak:



Rysunek 4.9: x i y mają wolne różne kolory, x_0 ma zajętą β i wolne α_0

Jeżeli teraz x ma wolny kolor α_1 to krawędź (x, x_0) możemy przekolorować na α_1, a krawędź (x, y) na α_0. Niech więc (x, x_1) będzie w kolorze α_1.

Jeśli x_1 miałby wolny kolor β to możemy przekolorować (x, x_1) na β, wtedy x zwalnia się α_1 a z tym wiemy co robić. No to niech w x_1 β będzie zajęta, a wolny będzie kolor α_2. Poniżej ilustracja:

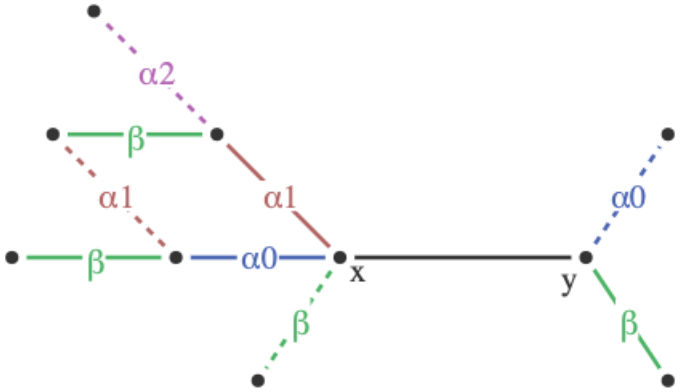
Podobnie jak wcześniej stwierdzamy, że z x wychodzi krawędź w kolorze α_2, bo inaczej przekolorujemy (x, x_1) na α_2. Kontynuujemy to rozumowanie aż napotkamy wierzchołek x_k o wolnym kolorze α_j, który już znajduje się wśród kolorów $\alpha_0, ..., \alpha_{k-1}$

Niestety nie możemy wykonać tej samej sztuczki z przekolorowaniem co wcześniej, ale to nic nie szkodzi bo zrobimy co innego. Otóż wyjdźmy z wierzchołka x_k i pójdźmy ścieżką w kolorach na przemian β i α_j. Oczywiście kiedyś skończą nam się krawędzie i wylądujemy w jakimś wierzchołku v.

Rozważmy sobie teraz przypadki czym ten wierzchołek v jest.

1. $v \notin \{x, x_0, x_1, ..., x_k\}$ Najfajniejszy przypadek - ścieżka kończy się w niezbyt istotnym miejscu. Zamieniamy kolory na ścieżce miejscami. Możemy tak zrobić, bo wewnętrznym wierzchołkom się nic nie zmienia, a na końcach odpowiedni kolor jest wolny.

   Teraz, począwszy od krawędzi (x, x_k) przekolorujemy wachlarz. Dzięki przekolorowaniu, x_k ma teraz wolną β tak jak x, zatem (x, x_k) możemy dać kolor β W takim razie x ma

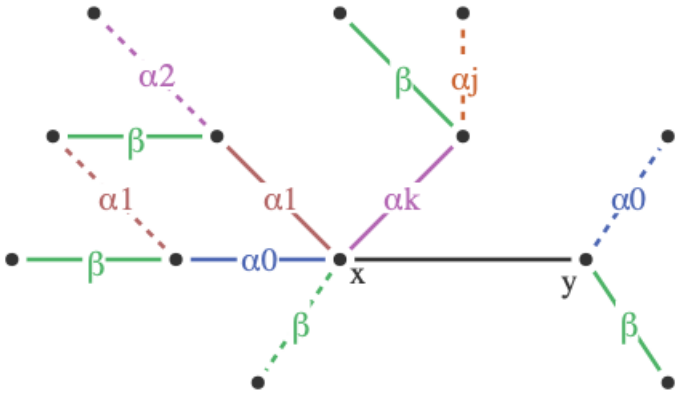


Rysunek 4.10: x_1 ma zajętą β a wolne α_2

teraz wolny kolor α_k tak jak x_{k-1}, zatem (x, x_{k-1}) dostanie kolor α_k. Podobnie (x, x_{k-2}) dostanie kolor α_{k-1}.

W końcu dojdziemy do (x, x_0), które dostanie kolor α_1. Sprawiliśmy, że x ma wolne α_0, więc z czystym sumieniem kolorujemy (x, y) na α_0.

Rysunek 4.12: Przekolorowanie wachlarza gdy ścieżka z x_k kończy się w poza wierzchołkami $x, x_0, ..., x_k$



Rysunek 4.11: x_k ma wolny kolor α_j, który już widzieliśmy.

2. $v = x_{j-1}$ Taka sytuacja niestety może zajść, bo x_{j-1} ma wolny kolor x_j i zajętą β. Zauważmy, że nie możemy zrobić tego co w przypadku pierwszym, bo przekolorowanie ścieżki sprawia, że x_{j-1} ma kolor α_j zajęty, a taki kolor by otrzymał przy poprawianiu wachlarza. Zrobimy zatem co innego.

   Tak jak wcześniej przekolorujemy ścieżkę, ale zamiast przekolorowywać krawędź (x, x_k) na β przekolorujemy (x, x_{j-1}) na β. Dalej możemy kontynuować tak jak poprzednio: (x, x_{j-1}) dostanie kolor α_{j-2} itd. (Musieliśmy przyjść do v krawędzią o kolorze β bo inaczej nie byłby to koniec ścieżki)

Rysunek 4.13: Przekolorowanie gdy ścieżka z x_k kończy się w x_{j-1}

□

4.9 Przepływy w sieciach. Twierdzenie o maksymalnym przepływie i minimalnym przekroju

4.9.1 Definicje

Źródło to wyróżniony wierzchołek, z którego krawędzie mogą jedynie „wychodzić" (graf jest skierowany).

Ujście to wyróżniony wierzchołek, do którego krawędzie mogą jedynie „wchodzić".

c jest to funkcja przepustowości, idąca ze zbioru krawędzi w liczby naturalne > 0 (tak po ludzku to mówiona jaką przepustowość ma dana krawędź).

Przepływem całkowitoliczbowym nazywamy funkcję f przyporządkowującą każdej krawędzi jakąś liczbę naturalną mniejszą lub równą jej przepustowości. Dla każdego wierzchołka który nie jest źródłem lub przepływem jest tak, że suma po funkcjach przepływów krawędzi do niego wchodzących jest równa sumie po funkcjach przepływów do niej wchodzących (co w sumie jest dosyć oczywiste, studenci zafiksowani na fizyce gadają tam o jakichś prawach kirchoffa i nie wiem o co im chodzi; studiuję TCS, nie prawo).

Przekrojem sieci przepływowej S nazywamy jej „podział" na 2 zbiory wierzchołków jej grafu (S, T), taki, że:

1. $S \cup T = V$
2. $S \cap T = \varnothing$
3. Źródło jest w S
4. Ujście jest w T

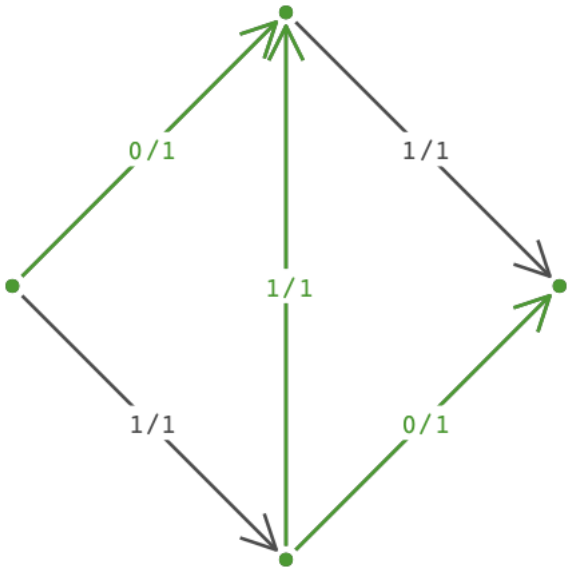
Przepustowością przekroju, również oznaczaną przez c (co generuje bałagan w oznaczeniach, ale to nie ja je wymyślałem) nazywamy sumę po przepustowościach wszystkich krawędzi wychodzących z S.

Przepływem przez przekrój nazywamy sumę po wartości funkcji f dla wszystkich krawędzi które „wychodzą" z S pomniejszoną o sumę po wartości funkcji f dla wszystkich krawędzi, które „wchodzą" do S (co jest dosyć intuicyjne, po prostu to co wypływa z S minus to co wpływa).

Przekrojem minimalnym nazywamy przekrój o minimalnej przepustowości. Od razu można też zauważyć, że przepustowość sieci (w sensie to ile maksymalnie może wpłynąć do ujścia) jest mniejsza lub równa od przepustowości przekroju minimalnego. Dowodzi się to za pomocą dowodu *to widać*. Formaliści mogą sobie nad tym podumać.

Ścieżką powiększającą w przepływie nazywamy zbiór krawędzi który będzie stanowić ścieżkę jeśli pominiemy ich skierowanie; generalnie to działa tak że musi ona iść od jakiegoś wierzchołka v do innego wierzchołka u; zazwyczaj idzie ona od źródła do ujścia. Jeśli dana krawędź idzie „do przodu" to przepływ na niej nie może być równy przepustowości; jeśli idzie „do tyłu" musi być niezerowy. Dlaczego to tak nazywamy? Bo jak do wszystkich krawędzi idących „do przodu" dodamy 1 a od wszystkich idących „do tyłu" odejmiemy 1 (a ścieżka szła od źródła do ujścia) to nadal mamy poprawny przepływ, w którym dodatku do ujścia wchodzi jedna jednostka więcej.

Przez $val(f)$ czasem będziemy oznaczać to, ile w danym przepływie wpływa do ujścia (i będziemy starali się maksymalizować tę wartość).



Rysunek 4.14: Przykład ścieżki powiększającej w sieci przepływowej

4.9.2 Twierdzenie o maksymalnym przepływie i minimalnym przekroju

Twierdzenie 4.9.1 (Oczywiste twierdzenie o przekrojach)**.** Dla dowolnego przepływu w sieci przepływowej f i dla dowolnego jej przekroju (S, T) jest tak, że przepływ przez przekrój jest słabo mniejszy niż jego przepustowość (tzn. $f(S, T) \leq c(S, T)$).

Dowód. To widać. W sensie serio, gdyby suma po przepływach krawędzi wychodzących z S była większa niż suma po ich przepustowościach, to znaczyłoby że coś gdzieś poszło bardzo mocno nie tak. □

Twierdzenie 4.9.2 (Mniej oczywiste twierdzenie o przekrojach)**.** Przepływ przez każdy przekrój sieci jest taki sam i wynosi $val(f)$.

Dowód. Indukcja po liczbie wierzchołków w części S przekroju (tej do której należy źródło). Przypadek bazowy gdy $|S| = 1$ trywialny. W przypadku ogólnym mamy sobie jakiś przekrój (S, T). Weźmy teraz jakiś $x \in S$ (różny od źródła, na pewno taki jest bo dla przypadku gdzie jest tylko jeden wierzchołek w S mamy to już udowodnione) i wrzućmy go do T, otrzymując alternatywny przekrój (S', T'). Teraz jest śmiesznie, bo wiemy z założenia indukcyjnego że przepływ przez $(S', T') = val(f)$, bo S' ma mniejszą moc od S. Rozpiszmy sobie teraz $f(S', T')$ oraz $f(S, T)$ (przez v_s etc. oznaczam wierzchołek należący do S lub innych zbiorów):

$$f(S,T) = \sum_{(v_s,v_t)\in E, v_s\neq x} f(v_s, v_t) + \sum_{(x,v_t)\in E} f(x, v_t) - \sum_{(v_t,v_s)\in E, v_s\neq x} f(v_t, v_s) - \sum_{(v_t,x)\in E} f(v_t, x)$$

Jako że jeśli punkt $v \in S$ oraz $v \neq x$, to $v \in S'$, możemy to uprościć i zapisać jako:

$$f(S,T) = \sum_{(v_{s'},v_t)\in E} f(v_{s'}, v_t) + \sum_{(x,v_t)\in E} f(x, v_t) - \sum_{(v_t,v_{s'})\in E} f(v_t, v_{s'}) - \sum_{(v_t,x)\in E} f(v_t, x)$$

$$f(S',T') = \sum_{(v_{s'},v_{t'})\in E, v_{t'}\neq x} f(v_{s'}, v_{t'}) + \sum_{(v_{s'},x)\in E} f(v_{s'}, x) - \sum_{(v_{t'},v_{s'})\in E, v_{t'}\neq x} f(v_{t'}, v_{s'}) - \sum_{(x,v_{s'})\in E} f(x, v_{s'})$$

Jako, że jeśli punkt $v \in T'$ oraz $v \neq x$, to wiemy że $v \in T$:

$$f(S',T') = \sum_{(v_{s'},v_t)\in E} f(v_{s'}, v_t) + \sum_{(v_{s'},x)\in E} f(v_{s'}, x) - \sum_{(v_t,v_{s'})\in E} f(v_t, v_{s'}) - \sum_{(x,v_{s'})\in E} f(x, v_{s'})$$

To wygląda przerażająco, ale w sumie wynika wprost z definicji przepływu, po prostu dodajemy te wszystkie krawędzie do siebie. Zasadniczo nic ciekawego. Co teraz jest fajne to to, że możemy policzyć $f(S, T) = f(S', T')$ (zwróćmy uwagę, że pierwszy i trzeci składnik się uproszczą (co w sumie ma sens, bo jedyne co zmienialiśmy to pozycja x, więc to co przepływa w reszcie grafu się nie zmieniło – ależ plot twist) , zostawiając tylko składniki z x).

$$f(S,T) - f(S',T') = \sum_{(x,v_t)\in E} f(x, v_t) - \sum_{(v_{s'},x)\in E} f(v_{s'}, x) - \sum_{(v_t,x)\in E} f(v_t, x) + \sum_{(x,v_{s'})\in E} f(x, v_{s'})$$

Jako, że do x wpływa tyle samo co wypływa:

$$\begin{aligned} f(S,T) - f(S',T') &= \sum_{(x,v_t)\in E} f(x, v_t) - \sum_{(v_t,x)\in E} f(v_t, x) + \sum_{(x,v_{s'})\in E} f(x, v_{s'}) - \sum_{(v_{s'},x)\in E} f(v_{s'}, x) \\ &= 0 + 0 = 0 \end{aligned}$$

Skąd mamy szokujące odkrycie, że $f(S, T) = f(S', T')$. Ale przecież $f(S', T')$ było równe $val(f)$, czyli wszystko się zgadza. □

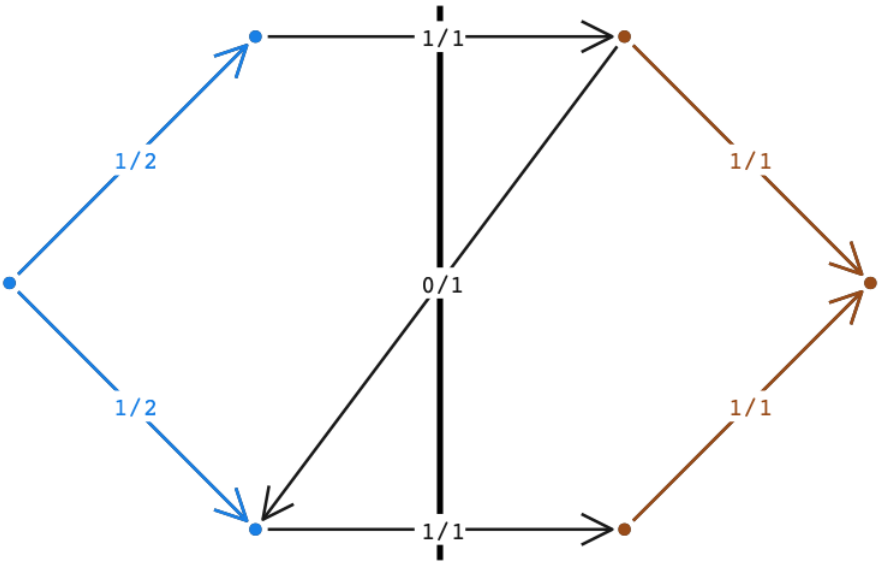
4.9.3 Twierdzenie Forda-Fulkersona

Twierdzenie 4.9.3. Następujące warunki są równoważne:

1. f jest przepływem maksymalnym
2. W sieci przepływowej nie istnieje ścieżka powiększająca od źródła do ujścia
3. Przekrój (S, T) taki, że S zawiera wszystkie wierzchołki do których istnieje ścieżka powiększająca od źródła, jest poprawnie zdefiniowanym przekrojem, spełniającym warunek $f(S, T) = c(S, T)$

Dowód. Trzeci warunek brzmi strasznie, ale tak naprawdę taki nie jest. Zajmijmy się poszczególnymi implikacjami celem udowodnienia równoważności:

1. (1) $\implies$ (2): Trywialne do udowodnienia, gdyby f było przepływem maksymalnym a istniałaby ścieżka powiększająca od źródła do ujścia to mógłbym zwiększyć wartość przepływu o 1 za jej pomocą, a więc przepływ ten nie byłby maksymalny.
2. (2) $\implies$ (3): Ładnie się to dowodzi, stosując dowód przez rysowanie. Idea generalnie jest taka, że poprawność przekroju podanego w (3) wynika nam z tego, że z (2) mamy że nie istnieje ścieżka powiększająca od źródła do ujścia, a więc ujście na pewno będzie T. Źródło na pewno będzie w S, oczywiście. Pozostałe warunki dla przekroju oczywiście będą spełnione, więc mamy że (S, T) to poprawnie zdefiniowany przekrój. Jak teraz sobie spojrzymy na wszystkie krawędzie wchodzące i wychodzące z S to odkrywamy, że skoro nie ma ścieżki powiększającej która wychodziłaby poza S, to wszystkie krawędzie wychodzące muszą być w pełni wysycone, a wszystkie wchodzące wyzerowane (inaczej moglibyśmy dorzucić wierzchołek z T do S, zgodnie z definicją naszego przekroju). No a skoro tak jest, to mamy że $f(S, T) = c(S, T)$; innymi słowy przepływ przez przekrój jest równy jego przepustowości.



Rysunek 4.15: Przekrój (S, T) taki, że do S należą wszystkie wierzchołki, do których idzie ścieżka powiększająca (zaznaczone na niebiesko). Nie jest możliwe poprowadzenie ścieżki powiększającej dalej, a więc krawędzie albo są maksymalnie wysycone (jeśli idą z S do T) albo wyzerowane (jeśli idą z T do S) (zaznaczone na czarno).

3. (3) $\implies$ (1): Ponieważ $val(f)$ musi być mniejsze niż przepustowość dowolnego przekroju, co zauważyliśmy już przy etapie definicji pojęć (a formaliści mam nadzieję że już dowiedli, o ile nadal nie zastanawiają się co to jest liczba parzysta), a z (3) mamy, że przekrój (S, T) spełnia $f(S, T) = c(S, T)$ to mamy że przepływ ten jest maksymalny (no w sensie większego przepływu nam się nie uda zrobić, skoro właśnie osiągnęliśmy limit).

Z powyższego dowodu oczywiście również wynika, że przepustowość minimalnego przekroju jest równa $val(f)$. □

Rozdział 5

MPI

5.1 Łańcuchy Markowa na przykładzie analizy randomizowanego algorytmu dla problemu 2-SAT

5.1.1 Definicje

Definicja 5.1.1. Procesem stochastycznym nazywamy dowolny zbiór zmiennych losowych $\{X(t) : t \in T\}$. Zwykle t oznacza moment w czasie, a $X(t)$ jest **stanem** tego procesu w czasie t i zapisujemy X_t
Mówimy, że proces jest **skończony** jeśli zmienne X_t przyjmują skończenie wiele wartości.

Definicja 5.1.2. Procesem Markowa (czasu homogenicznego) nazywamy taki proces stochastyczny $X_0, X_1, X_2, \ldots$ w którym dla dowolnego t zachodzi

$$P(X_t = a_t \mid X_{t-1} = a_{t-1}, X_{t-2} = a_{t-2}, \ldots X_0 = a_0) = P(X_t = a_t \mid X_{t-1} = a_{t-1})$$

Innymi słowy aby dostać rozkład zmiennej X_t wystarczy, że znamy rozkład zmiennej X_{t-1} tzn. łańcuch Markowa jest bez pamięci.

Warto zauważyć, że **nie oznacza to**, że X_t jest niezależne od $X_{t-2}, X_{t-3}, \ldots$ – jest zależne, ale cała ta zależność jest zawarta w zależności od stanu X_{t-1}.

Definicja 5.1.3. Łańcuch Markowa jest **czasu homogenicznego** jeśli $P(X_t = a_t \mid X_{t-1} = a_{t-1} \wedge t = t_0) = P(X_t = a_t \mid X_{t-1} = a_{t-1})$

Mniej formalnie - na rozwój wydarzeń ma jedynie wpływ stan łańcucha, a nie czas w którym ten stan ma miejsce.

5.1.2 Randomizowany 2-SAT

2-SATa nikomu nie trzeba przedstawiać – mamy n zmiennych i k klauzul postaci $a \vee b$, gdzie $a = x_i$ albo $a = \bar{x}_i$.

Przykładem instancji problemu 2-SAT jest na przykład taka formuła:

$$(x_1 \vee x_3) \wedge (\neg x_1 \vee x_2) \wedge (x_3 \vee \neg x_1)$$

Znamy algorytm rozwiązujący ten problem w czasie $\mathcal{O}(n + k)$ za pomocą silnie spójnych składowych, ale tutaj pokażemy **wolniejszy** algorytm.

Niech $\lambda \in \mathbb{N}$ będzie parametrem algorytmu (stałą czasu działania).

1. Wylosuj dowolne wartościowanie zmniennych $x_1, \ldots, x_n$.
2. Powtarzaj maksymalnie $2\lambda n^2$ razy lub do znalezienia rozwiązania:
   (a) Wylosuj niespełnioną klazulę.
   (b) Wylosuj literał z tej klauzuli i odwróć wartość zmiennej tego literału.
3. Jeśli znajdziemy jakieś wartościowanie które spełnia tę formułę to je zwracamy.
4. W przeciwnym razie orzekamy, że formuła jest niespełnialna.

5.1.3 Własności algorytmu

Twierdzenie 5.1.1 (Lemat 7.1 P&C)**.** Jeśli dana jest formuła spełnialna, oraz pozwalamy działać algorytmowi dowolnie długo to oczekiwana liczba kroków wynosi co najwyżej n^2.

Dowód. Będziemy modelować zachowanie algorytmu jako łańcuch Markowa (a jakże). Wybierzmy sobie dowolne wartościowanie S, które spełnia formułę. Oznaczmy wartościowanie stworzone przez algorytm w i-tym kroku przez A_i.

Niech X_i oznacza liczbę zmiennych, które mają to samo wartościowanie w S oraz w A_i.

Mamy zatem taki proces $X_0, X_1, \ldots$, który niestety nie jest łańcuchem Markowa, bo tracimy informacje o tym, które zmienne mają jakie wartościowanie, a prawdopodobieństwo przejścia z X_i do X_{i+1} jest zadane wartościowaniem A_i, które nie jest częścią łańcucha.

Zrobimy zatem sztuczkę i rozważymy proces $Y_0, Y_1, \ldots$, który będzie łańcuchem Markowa i jednocześnie będzie pesymistyczną sytuacją naszego procesu.

Zauważmy, że jeśli wybieramy klauzulę, która nie jest spełniona, to wartościowania A_i oraz S wartościują którąś ze zmiennych (być może obie) inaczej. W takim razie

$$P(X_{i+1} = j + 1 \mid X_i = j) \geq \frac{1}{2}$$

$$P(X_{i+1} = j - 1 \mid X_i = j) \leq \frac{1}{2}$$

Zatem w pesymistycznej sytuacji, którą modeluje nasz Y_i mamy:

$$Y_0 = X_0$$

$$P(Y_{i+1} = j + 1 \mid Y_i = j) = \frac{1}{2}$$

$$P(Y_{i+1} = j - 1 \mid Y_i = j) = \frac{1}{2}$$

Przy czym (z uwagi na to, że mamy tylko jedną opcję)

$$P(Y_{i+1} = 1 \mid Y_i = 0) = 1$$

Niech Z_i oznacza liczbę kroków potrzebną do pierwszego dotarcia do stanu n zaczynając w stanie i. Mamy

$$h_i = \mathbb{E}[Z_i] = \mathbb{E}\left[1 + \frac{Z_{i-1}}{2} + \frac{Z_{i+1}}{2}\right]$$

Dostajemy zatem układ $n+1$ równań

$$\begin{cases} h_0 = h_1 + 1 \\ h_i = 1 + \frac{1}{2}(h_{i-1} + h_{i+1}) \\ h_n = 0 \end{cases}$$

Przekształcamy środkową zależność do postaci

$$h_{i+1} = 2h_i - h_{i-1} - 2$$

Rozwiązując indukcyjnie dostajemy

$$\begin{aligned} h_0 = h_1 + 1 &\iff h_1 = h_0 - 1 \\ h_1 = 1 + \frac{1}{2}(h_0 + h_2) &\iff h_2 = 2h_1 - h_0 - 2 = h_1 - 3 = h_0 - 1 - 3 \\ &\quad h_3 = 2h_2 - h_1 - 2 = h_2 - 5 = h_0 - 1 - 3 - 5 \\ &\quad \dots \\ &\quad h_i = h_0 - i^2 \end{aligned}$$

W takim razie

$$h_n = 0 = h_0 - n^2$$

czyli

$$h_0 = n^2$$

Start w stanie 0 jest najbardziej pesymistyczny, a pokazaliśmy, że nawet dla takiego stanu w oczekiwaniu po n^2 krokach znajdziemy wartościowanie, co kończy dowód ograniczenia górnego. □

Twierdzenie 5.1.2 (Lemat 7.2 P&C)**.** Jeśli formuła jest spełnialna, to algorytm z sekcji 5.1.2 myli się z prawdopodobieństwem co najwyżej $2^{-\lambda}$.

Dowód. Dzielimy wykonanie algorytmu na bloki długości $2n^2$. Niech Z_i oznacza liczbę kroków wykonaną od początku i-tego bloku (zakładając, że nie znaleźliśmy wcześniej wartościowania).

Pokazaliśmy przed chwilą, że $\mathbb{E}[Z_i] \leq n^2$, a zatem z nierówności Markowa:

$$\mathrm{P}(Z_i > 2n^2) \leq \frac{n^2}{2n^2} = \frac{1}{2}$$

W takim razie prawdopodobieństwo, że po wykonaniu λ bloków nie znaleźliśmy wartościowania wynosi

$$\left(\frac{1}{2}\right)^{\lambda}$$

□

5.2 Problem kul i urn ze wzmocnionym feedbackiem (jako ilustracja zastosowania rozkładu wykładniczego)

5.2.1 Ciągłość w probabilistyce

Wiem, wiem, pytanie o kulach i urnach a ja tu z tym jak wygląda ciągłość w probabilistyce, ale w żadnym pytaniu wcześniej sobie tego nie zdefiniowaliśmy, więc brzmi to jak niezły moment by to teraz zrobić.

Jeśli jakaś zmienna losowa jest *ciągła*, to musimy zacząć posługiwać się dystrybuantami[1], tzn. takimi funkcjami $F : \mathbb{R} \to \mathbb{R}$, że:

$$F_X(x) = P(x \geq X)$$

Dystrybuantę można generalizować na wiele zmiennych losowych jednocześnie:

$$F_{XY}(x, y) = \mathrm{P}(x \geq X \wedge y \geq Y)$$

Jeśli X i Y są niezależne to ich wspólna dystrybuanta jest iloczynem ich „indywidualnych" dystrybuant, tzn.

$$F_{XY}(x, y) = F_X(x) \cdot F_Y(y)$$

Ponadto definiujemy gęstość rozkładu $f(x)$ jako pierwszą pochodną dystrybuanty, tzn.

$$f(x) = F'(x)$$

I analogicznie możemy zdefiniować wspólną gęstość rozkładu $f_{XY}(x, y)$ jako pochodną po współrzędnej x i y (tzn. $\frac{\partial^2}{\partial x \partial y} F_{XY}(x, y)$) dystrybuanty tychże dwóch zmiennych.

Fakt 5.2.1. Jeśli zmienne losowe X i Y są niezależne, to $f_{XY}(x, y) = f_X(x) \cdot f_Y(y)$.

Dowód. Skoro X i Y są niezależne, to:

$$F_{XY}(x, y) = F_X(x) \cdot F_Y(y)$$

a zatem

$$\frac{\partial}{\partial x} F_{XY}(x, y) = \frac{\partial}{\partial x}(F_X(x) \cdot F_Y(y)) = F_Y(y) \cdot f_X(x)$$

więc

$$\frac{\partial^2}{\partial x \partial y} F_{XY}(x, y) = \frac{\partial}{\partial y}(F_Y(x) \cdot f_X(x)) = f_Y(y) \cdot f_X(x)$$

□

[1] To wynika z faktu, że gdy dopuszczamy nieprzeliczalnie wiele zdarzeń to prawdopodobieństwo wystąpienia części z nich wyniesie 0 (mimo że mogą się zdarzyć) i bardziej nas interesuje prawdopodobieństwo tego, że zmienna wyląduje w jakimś przedziale.

Po co nam to wszystko? Bo teraz możemy sobie radośnie obliczać prawdopodobieństwo, że zmienna losowa przyjmie wartość w jakimś przedziale:

$$\mathrm{P}(a \leq x \leq b) = \int_a^b f(x)\, dx$$

Podobnie możemy zrobić by policzyć prawdopodobieństwa zachowań wielu różnych zmiennych losowych, korzystając z ich wspólnych gęstości, na przykład:

$$\mathrm{P}(a \leq x \leq b \wedge c \leq y \leq d) = \int_{x=a}^b \int_{y=c}^d f_{XY}(x,y)\, dydx$$

Co w przypadku gdy są one niezależne od siebie (co będzie częste w naszych rozważaniach) będzie można zapisać tak:

$$\mathrm{P}(a \leq x \leq b \wedge c \leq y \leq d) = \int_{x=a}^b f_X(x) \int_{y=c}^d f_Y(y)\, dydx$$

A jak w tym przezabawnym probabilistycznym świecie wygląda definicja wartości oczekiwanej zmiennej losowej? Ano tak:

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} f(x) \cdot x\, dx$$

No i oczekiwana jakiejkolwiek zmiennej losowej od X zależnej, danej jakąś funkcją $g(X)$ może zostać obliczona w taki sposób:

$$\mathbb{E}[g(X)] = \int_{-\infty}^{\infty} f_X(x) \cdot g(x)\, dx$$

5.2.2 Rozkład wykładniczy

Definicja 5.2.1. Rozkładem wykładniczym z parametrem λ nazywamy rozkład zadany dystrybuantą

$$F(x) = \begin{cases} 1 - e^{-\lambda x} & \text{dla } x \geq 0 \\ 0 & \text{dla } x < 0 \end{cases}$$

Gęstość rozkładu wykładniczego $f = F'$ wynosi natomiast $f(x) = \lambda e^{-\lambda x}$.

Lemat 5.2.1. Wartość oczekiwana rozkładu wykładniczego o parametrze λ wynosi $\frac{1}{\lambda}$

Dowód.

$$\int_0^{\infty} \lambda x e^{-\lambda x}\, dx = \lambda \int_0^{\infty} x e^{-\lambda x}\, dx$$

Lecimy całkując przez części:

- $u = x$
- $v' = e^{-\lambda x}$

więc:

- $u' = 1$
- $v = \frac{-1}{\lambda} e^{-\lambda x}$

$$\int x e^{-\lambda x}\, dx = \frac{-x}{\lambda} e^{-\lambda x} - \int \frac{-1}{\lambda} e^{-\lambda x}$$

z kolei:

$$\int \frac{-1}{\lambda} e^{-\lambda x} = \frac{-1}{\lambda} \int e^{-\lambda x} = \frac{-1}{\lambda} \left(\frac{-1}{\lambda} e^{-\lambda x} + c \right) = \frac{1}{\lambda^2} e^{-\lambda x} - \frac{1}{\lambda}$$

a zatem

$$\lambda \int_0^\infty x e^{-\lambda x}\, dx = \lambda \Big|_0^\infty \left(\frac{-x}{\lambda} e^{-\lambda x} - \frac{1}{\lambda^2} e^{-\lambda x} + \frac{1}{\lambda} \right) = \lambda \cdot \frac{1}{\lambda^2} = \frac{1}{\lambda}$$

□

Twierdzenie 5.2.1 (Lemat 8.4 P&C)**.** Rozkład wykładniczy jest **bez pamięci** tzn.

$$P(X > s + t \mid X > t) = P(X > s)$$

Dowód.

$$\begin{aligned} P(X > s + t \mid X > t) &= \frac{P(X > s + t)}{P(X > t)} \\ &= \frac{1 - P(X \leq s + t)}{1 - P(X \leq t)} \\ &= \frac{\exp(-\lambda(s + t))}{\exp(-\lambda t)} \\ &= e^{-\lambda s} = P(X > s) \end{aligned}$$

□

Jest to bardzo przydatna własność, bowiem sprawia, że możemy „resetować" zmienną o której wiemy, że ma większą wartość niż ustalona.

Twierdzenie 5.2.2 (Lemat 8.5 P&C)**.** Jeśli X_1, X_2 są **niezależnymi** zmiennymi losowymi o rozkładzie geometrycznym o parametrach (odpowiednio) λ_1 i λ_2 to zmienna losowa Y będąca ich minimum jest zmienną o rozkładzie geometrycznym o parametrze $\lambda_1 + \lambda_2$.

Dodatkowo, prawdopodobieństwo że $X_1 = Y$ (w sensie: że to X_1 będzie mniejsze) wynosi $\frac{\lambda_1}{\lambda_1 + \lambda_2}$. Analogicznie, dla X_2 prawdopodobieństwo to wyniesie $\frac{\lambda_2}{\lambda_1 + \lambda_2}$.

Dowód. Policzmy sobie dystrybuantę zmiennej losowej Y (w nieco śmieszny sposób):

$$1 - F_Y(x) = \mathrm{P}(x \leq Y) = \mathrm{P}(x \leq X_1 \wedge x \leq X_2) = \mathrm{P}(x \leq X_1) \cdot \mathrm{P}(x \leq X_2)$$

$$P(x \leq X_1) \cdot P(x \leq X_2) = (1 - F_{X_1}(x)) \cdot (1 - F_{X_2}(x))$$

Jako, że dla zmiennych X rozkładu wykładniczego $F_X(x) = 1 - e^{-\lambda x}$ to $1 - F_X(x) = e^{-\lambda x}$. Zatem:

$$(1 - F_{X_1}(x)) \cdot (1 - F_{X_2}(x)) = e^{-\lambda_1 x} \cdot e^{-\lambda_2 x} = e^{-(\lambda_1+\lambda_2)x} = 1 - F_Y(x)$$

czyli

$$F_Y(x) = 1 - e^{-(\lambda_1+\lambda_2)x}$$

skąd Y jest zmienną rozkładu wykładniczego z parametrem $\lambda_1 + \lambda_2$. Teraz pozostaje pokazać, że:

$$P(X_1 \leq X_2) = \frac{\lambda_1}{\lambda_1 + \lambda_2}$$

Zatem liczymy:

$$\begin{aligned}
\mathrm{P}(X_1 \leq X_2) = \int_{x_2=-\infty}^{\infty} \int_{x_1=-\infty}^{x_2} f_{X_1X_2}(x_1, x_2)\, dx_1 dx_2 = \\
\int_{x_2=-\infty}^{\infty} f_{X_2}(x_2) \int_{x_1=-\infty}^{x_2} f_{X_1}(x_1)\, dx_1 dx_2 = \\
\int_{x_2=0}^{\infty} \lambda_2 e^{-\lambda_2 x_2} \int_{x_1=0}^{x_2} \lambda_1 e^{-\lambda_1 x_1}\, dx_1 dx_2 = \\
\lambda_1\lambda_2 \int_{x_2=0}^{\infty} e^{-\lambda_2 x_2} \int_{x_1=0}^{x_2} e^{-\lambda_1 x_1}\, dx_1 dx_2 = \\
\lambda_1\lambda_2 \int_{x_2=0}^{\infty} e^{-\lambda_2 x_2} \left(\Big|_0^{x_2} \frac{-1}{\lambda_1} e^{-\lambda_1 x_1} \right) dx_2 = \\
\lambda_1\lambda_2 \int_{x_2=0}^{\infty} e^{-\lambda_2 x_2} \left(\frac{-1}{\lambda_1} e^{-\lambda_1 x_2} - \frac{-1}{\lambda_1} e^{0} \right) dx_2 = \\
\lambda_1\lambda_2 \int_{x_2=0}^{\infty} e^{-\lambda_2 x_2} \frac{-1}{\lambda_1} \left(e^{-\lambda_1 x_2} - 1 \right) dx_2 = \\
-\lambda_2 \int_{x_2=0}^{\infty} e^{-\lambda_2 x_2} \left(e^{-\lambda_1 x_2} - 1 \right) dx_2 = \\
-\lambda_2 \int_{x_2=0}^{\infty} e^{-\lambda_2 x_2 - \lambda_1 x_2} - e^{-\lambda_2 x_2}\, dx_2 = \\
-\lambda_2 \int_{x_2=0}^{\infty} e^{-x_2(\lambda_2+\lambda_1)} - e^{-\lambda_2 x_2}\, dx_2 = \\
-\lambda_2 \left(\int_{x_2=0}^{\infty} e^{-x_2(\lambda_2+\lambda_1)}\, dx_2 - \int_{x_2=0}^{\infty} e^{-\lambda_2 x_2}\, dx_2 \right) = \\
-\lambda_2 \left(\left(\Big|_{x_2=0}^{\infty} \frac{-1}{\lambda_1 + \lambda_2} e^{-x_2(\lambda_1+\lambda_2)} \right) - \left(\Big|_{x_2=0}^{\infty} \frac{-1}{\lambda_2} e^{-\lambda_2 x_2} \right) \right) = \\
-\lambda_2 \left(\frac{1}{\lambda_1 + \lambda_2} - \frac{1}{\lambda_2} \right) = \\
-\lambda_2 \left(\frac{\lambda_2}{\lambda_2 \cdot (\lambda_1 + \lambda_2)} - \frac{\lambda_1 + \lambda_2}{\lambda_2 \cdot (\lambda_1 + \lambda_2)} \right) = \\
(-\lambda_2) \cdot \frac{-\lambda_1}{\lambda_2 \cdot (\lambda_1 + \lambda_2)} = \\
\frac{\lambda_1}{\lambda_1 + \lambda_2}
\end{aligned}$$

□

5.2.3 Problem kul i urn z feedbackiem

Jak zwykle, zanim zaczniemy to pokażemy pomocniczy lemat:

Lemat 5.2.2. Niech X będzie dowolną zmienną losową ze skończoną wartością oczekiwaną, tj. $\mathbb{E}[X] \in \mathbb{R}$. Wtedy

$$P(X < \infty) = 1$$

Dowód. Korzystamy z nierówności Markowa

$$P(X \geq n) \leq \frac{\mathbb{E}[X]}{n}$$

Zatem

$$\lim_{n\to\infty} P(X \geq n) \leq \lim_{n\to\infty} \frac{\mathbb{E}[X]}{n} = 0$$

□

Kule i urny jakie są każdy widzi. Rozważmy sobie jednak zabawny model, w którym mamy tylko dwie urny ale z takim twistem, że im więcej kul jest w urnie, tym większa szansa na to, że wrzucimy tam kolejną kulę.

Konkretniej - jeśli w pierwszej urnie jest x kul a w drugiej y to prawdopodobieństwo, że kolejna kula trafi do pierwszej urny wynosi

$$\frac{x^p}{x^p + y^p}$$

a do drugiej

$$\frac{y^p}{x^p + y^p}$$

dla ustalonego p.

Będziemy się zajmować $p > 1$ tzn. więcej kul dostaje „cięższa" urna.

Twierdzenie 5.2.3. Dla dowolnego $p > 1$ oraz dowolnych warunków początkowych, z prawdopodobieństwem 1 od pewnego momentu kule wpadają tylko do jednej urny.

Dowód. Przyjmijmy, że w obu urnach jest po jednej kuli, uprości to dowód, a rozumowanie pozostaje takie same.

Rozważmy inny, choć podobny, proces. Każda urna dostaje własny, niezależny licznik, który odlicza czas do przyjścia kolejnej kuli do tej konkretnej urny.

Jeśli w pierwszej urnie jest x kul to czas oczekiwania na kolejną wynosi T_x, które ma rozkład wykładniczy z parametrem x^p.

Podobnie dla drugiej urny – jeśli jest w niej y kul to mamy zmienną U_y z parametrem y^p.

Zauważamy teraz fajną rzecz – prawdopodobieństwo, że kolejna kula ląduje w pierwszej urnie wynosi dokładnie (co wynika z twierdzenia 5.2.2):

$$\frac{x^p}{x^p + y^p}$$

a w drugiej:

$$\frac{y^p}{x^p + y^p}$$

Czyli nasz nowy proces jest taki sam jak oryginalny, cóż za zbieg okoliczności.

Definiujemy „czasy nasycenia", mające opisywać po jakim czasie liczba kul w urnach będzie dowolnie duża.

$$X = \sum_{i=1}^{\infty} T_i$$

$$Y = \sum_{i=1}^{\infty} U_i$$

gdzie X jest czasem nasycenia dla pierwszej urny, a Y jest czasem nasycenia dla drugiej urny.

Lemat 5.2.3. Zmienne losowe X i Y, opisujące czasy nasycenia, mają skończoną wartość oczekiwaną.

Dowód.

$$\mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^{\infty} T_i\right] = \sum_{i=1}^{\infty} \mathbb{E}[T_i]$$

(drugie przejście jest legalne dla nieskończonej sumy pod warunkiem, że $\sum_{i=1}^{\infty} \mathbb{E}[|T_i|]$ jest zbieżny, ale to zaraz pokażemy)

i analogicznie:

$$\mathbb{E}[Y] = \mathbb{E}\left[\sum_{i=1}^{\infty} U_i\right] = \sum_{i=1}^{\infty} \mathbb{E}[U_i]$$

Zarówno T_i jak i U_i to zmienne rozkładu wykładniczego o parametrze i^p, a zatem

$$\mathbb{E}[T_i] = \mathbb{E}[U_i] = \frac{1}{i^p}$$

Zauważamy teraz[2], że:

$$\sum_{i=1}^{\infty} \mathbb{E}[|T_i|] = \sum_{i=1}^{\infty} \mathbb{E}[T_i]$$

A szereg

$$\sum_{i=1}^{\infty} \mathbb{E}[T_i] = \sum_{i=1}^{\infty} \frac{1}{i^p}$$

jest bezdyskusyjnie zbieżny z kryterium zbieżności dla szeregów harmonicznych. To po pierwsze pokazuje że nasze przejście na początku dowodu było legalne, a przy okazji pokazuje że $\mathbb{E}[X]$ jest skończona (analogiczne dla Y).

□

[2](to jest mało szokujące odkrycie)

Lemat 5.2.4. Czasy nasycenia z prawdopodobieństwem 1 są skończone.

Dowód. Tego P&C nie mówi, ale na to trzeba formalnie uważać (bo a priori nie wiemy, że jeśli zmienna ma skończoną oczekiwaną to z prawdopodobieństwem 1 *zmienna* przyjmuje skończoną wartość.)

Szczęśliwie dowiedliśmy wcześniej lemat 5.2.2 i dzięki temu wiemy że w istocie X, Y z prawdopodobieństwem 1 są skończone. □

Lemat 5.2.5. Z prawdopodobieństwem 1 czasy nasycenia są różne (tzn. $X \neq Y$).

Dowód. Rozpiszmy sobie warunek na to, że $X = Y$. Wtedy

$$\sum_{i=1}^{\infty} T_i = \sum_{i=1}^{\infty} U_i$$

czyli

$$T_1 = \sum_{i=1}^{\infty} U_i - \sum_{i=2}^{\infty} T_i$$

możemy sobie wyobrazić sytuację, że „wylosowaliśmy" już wszystkie wartości w eksperymencie poza T_1; wtedy z powyższego wzoru dostaniemy ile **dokładnie** musi wynieść T_1 by ten warunek zaszedł. Ale zaraz chwilunia, przecież T_1 jest zmienną rozkładu wykładniczego, zmienną ciągłą, prawdopodobieństwo że przyjmie jakąś konkretną wartość wynosi 0. Czyli z prawdopodobieństwem 1 ten warunek nie zachodzi. □

Bogaci w wiedzę że z prawdopodobieństwem 1 $X \neq Y$ możemy bez straty ogólności założyć, że $X > Y$. Oznacza to, że dla pewnego n

$$\sum_{i=1}^{n} T_i < Y < \sum_{i=1}^{n+1} T_i$$

a to z kolei oznacza, że:

$$\exists_{M_0 \in \mathbb{N}} \forall_{m > M_0} : \sum_{i=1}^{n} T_i < \sum_{i=1}^{m} U_i < \sum_{i=1}^{n+1} T_i$$

W takim razie, dla odpowiednio dużych m pierwsza urna zawiera m kul, a druga urna zawiera jedynie n kul, czyli z prawdopodobieństwem 1 druga urna „utknęła" na posiadaniu n kul, a to jest to co chcieliśmy pokazać. □

5.3 Wykorzystaj liniowość wartości oczekiwanej i oblicz oczekiwaną liczbę wykonanych porównań w algorytmie sortowania Quicksort. Możesz przyjąć, że sortujemy tablicę parami różnych elementów.

5.3.1 Liniowość wartości oczekiwanej

Dla porządku przytoczymy definicję wartości oczekiwanej i dowód liniowości wartości oczekiwanej.

Definicja 5.3.1. Wartość oczekiwaną zmiennej losowej X definiujemy jako

$$\mathbb{E}[X] = \sum_{x \in \operatorname{im} X} x \cdot P(X = x)$$

Twierdzenie 5.3.1. Jeśli X i Y są zmiennymi losowymi, to $\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$.

Dowód.

$$\begin{aligned}
\mathbb{E}[X+Y] = \sum_{x\in X}\sum_{y\in Y} \mathrm{P}(X = x \wedge Y = y) \cdot (x + y) = \\
\sum_{x\in X}\sum_{y\in Y} \mathrm{P}(X = x \wedge Y = y) \cdot x + \sum_{x\in X}\sum_{y\in Y} \mathrm{P}(X = x \wedge Y = y) \cdot y = \\
\sum_{x\in X} x \sum_{y\in Y} \mathrm{P}(X = x \wedge Y = y) + \sum_{y\in Y} y \sum_{x\in X} \mathrm{P}(X = x \wedge Y = y) = \\
\sum_{x\in X} x \cdot \mathrm{P}(X = x) + \sum_{y\in Y} y \cdot \mathrm{P}(Y = y) = \\
\mathbb{E}[X] + \mathbb{E}[Y]
\end{aligned}$$

□

5.3.2 Oczekiwana liczba porównań

Quicksort jaki jest każdy widzi – pamiętamy z ASD, że jego złożoność to pesymistycznie $\mathcal{O}(n^2)$, ale w losowym przypadku $\Theta(n \lg n)$.

Twierdzenie 5.3.2 (2.11 P&C)**.** Rozważmy standardowy algorytm Quicksort, w którym pivota wybieramy losowo, niezależnie i jednostajnie. Wtedy oczekiwana liczba porównań wynosi $2n \ln n + \mathcal{O}(n)$.

Dowód. Niech $x_1, \ldots, x_n$ będzie wejściowym ciągiem n różnych liczb. Niech $y_1, \ldots y_n$ będzie posortowaną permutacją tych wartości.

Definiujemy indykatory dla $i < j$; niech

$$X_{i,j} = \begin{cases} 1 & \text{jeśli } y_i, y_j \text{ zostały porównane chociaż raz} \\ 0 & \text{wpp.} \end{cases}$$

Łączna liczba porównań X wynosi $X = \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} X_{i,j}$ Oczekiwana liczba porównań wynosi zatem

$$\mathbb{E}[X] = \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} \mathbb{E}[X_{i,j}]$$

Zastanówmy się kiedy elementy y_i, y_j są porównywane. Na pewno któryś z nich musi zostać wybrany jako pivot. Ale ponadto muszą być w momencie tego wyboru na jednej liście, która jest aktualnie sortowana. Niech $Y^{i,j} = \{y_i, \ldots, y_j\}$.

Jeśli wybrany zostanie pivot który leży poza tą listą, to nie dojdzie do „rozspójnienia" tej listy i kiedyś będzie mogło nadal dojść do porównania y_i z y_j.

Jeśli wybrany zostanie pivot z tej listy różny od y_i oraz y_j, to te 2 elementy już nigdy nie zostaną ze sobą porównane, jako że będą znajdywać się na 2 oddzielnych listach.

W takim razie $X_{i,j} = 1$ wtedy i tylko wtedy, gdy pierwszym pivotem wybranym ze zbioru $Y^{i,j}$ jest element y_i lub element y_j.

Jako, że losowanie jest jednostajne i w ogóle, to każdy element z listy ma dokładnie takie same szanse na „zostanie pivotem". Jako, że elementów na liście jest $j-i+1$, to prawdopodobieństwo, że wybierzemy y_1 lub y_j wynosi $\frac{2}{j-i+1}$, czyli $\mathbb{E}[X_{i,j}] = \frac{2}{j-i+1}$.

Aby policzyć ostateczny wynik sumujemy się po wszystkich parach $i < j$:

$$\begin{aligned}
\mathbb{E}[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= 2 \sum_{i=1}^{n-1} \sum_{k=2}^{n-i+1} \frac{1}{k} \\
&= 2 \sum_{k=2}^{n} \sum_{i=1}^{n-k+1} \frac{1}{k} \\
&= 2 \sum_{k=2}^{n} \frac{n+1-k}{k} \\
&= 2 \left((n+1) \sum_{k=2}^{n} \frac{1}{k} \right) - 2(n-1) \\
&= 2 \left((n+1) \left(\sum_{k=1}^{n} \frac{1}{k} \right) - (n+1) \right) - 2(n-1)
\end{aligned}$$

Teraz korzystamy z faktu, że $\sum_{k=1}^{n} \frac{1}{k} = H_n = \ln n + \Theta(1)$ i dostajemy

$$\begin{aligned}
\mathbb{E}[X] &= 2(n+1) \cdot H_n - \Theta(n) \\
&= 2(n+1) \cdot (\ln n + \Theta(1)) - \Theta(n) \\
&= 2n \ln n + \Theta(n)
\end{aligned}$$

□

5.4 Igłę długości l rzucono na podłogę z desek o szerokości d, przy czym $l \leq d$. Jakie jest prawdopodobieństwo, że igła przetnie krawędź deski?

5.4.1 Rozkład jednostajny

Believe it or not, ale w tym rozdziale będziemy używać rozkładu jednostajnego (a jeszcze go nie zdefiniowaliśmy). Pozwolimy sobie zatem przytoczyć tutaj podstawowe definicje, a potem przejść do odpowiedzi na właściwe pytanie.

Definicja 5.4.1. Mówimy, że zmienna losowa X ma **rozkład jednostajny** na przedziale $[a, b]$ jeśli dystrybuanta tej zmiennej zadana jest przez funkcję

$$F(x) = \begin{cases} 0 & \text{gdy } x < a \\ \frac{x-a}{b-a} & \text{gdy } a \leq x \leq b \\ 1 & \text{gdy } x > b \end{cases}$$

Łatwo zauważyć, że gęstość takiej zmiennej wynosi $\frac{1}{b-a}$ w $[a, b]$ oraz 0 wszędzie indziej.

5.4.2 Wstęp

W problemie Igły Buffona rzucamy „igłą" na podłogę złożoną z „desek". Pytamy się o prawdopodobieństwo przecięcia igły z krawędzią deski.

Bardziej formalnie, mamy płaszczyznę na której są zaznaczone proste, które są parami równoległe. Odstęp między kolejnymi prostymi jest ustalony i wynosi on d - szerokość naszej deski.

Następnie losujemy odcinek o długości l - długość igły, gdzieś na tej płaszczyźnie i pytamy się o prawdopodobieństwo, że ten odcinek przecina jedną z naszych prostych.

5.4.3 Rozwiązanie

Twierdzenie 5.4.1. Prawdopodobieństwo że igła przetnie którąś z krawędzi wynosi $\frac{2l}{4\pi}$

Dowód. Niech X oznacz odległość od środka igły do najbliższej krawędzi deski. Oznaczmy θ kąt ostry pomiędzy naszą igłą a jedną z wielu równoległych prostych.

Oczywiście obie z tych zmiennych mają rozkład jednostajny.

$$f_X(x) = \begin{cases} \frac{2}{d} : & 0 \leq x \leq \frac{d}{2} \\ 0 : & \text{wpw} \end{cases}$$

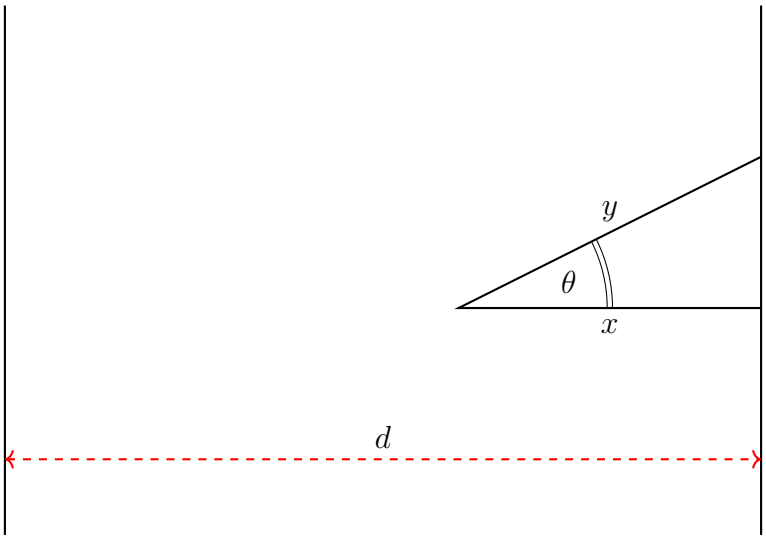
$$f_\theta(\theta) = \begin{cases} \frac{2}{\pi} : & 0 \leq \theta \leq \frac{\pi}{2} \\ 0 : & \text{wpw} \end{cases}$$

Powyżej korzystamy z faktu, że jest bijekcja między sytuacją gdy środek igły jest bliżej lewej a sytuacją gdy jest bliżej prawej krawędzi (bo rozumowanie jest to samo). Analogicznie zakładamy z kątem „odchylenia" igły. Tu też pojawia się miejsce w którym użyjemy założenia, że $l \leq d$: dzięki temu wiemy, że igła zawsze przetnie maksymalnie jedną krawędź[3].

[3]Modulo sytuacja gdzie wypadnie dokładnie na środku i kąt będzie zerowy, ale takie zdarzenie ma miarę zero i nie wpłynie na wynik naszego prawdopodobieństwa; tym samym równie dobrze możemy założyć, że $l < d$.

Zmienne te są w oczywisty sposób niezależne, więc ich wspólny rozkład prawdopodobieństwa będzie ich iloczynem.

$$f_{X\theta}(x,\theta) = \begin{cases} \frac{4}{d\pi} : & 0 \le x \le \frac{d}{2}, 0 \le \theta \le \frac{\pi}{2} \\ 0 : & \text{wpw} \end{cases}$$



Igła będzie przecinać krawędź, gdy $\frac{l}{2} \ge y$. Jednocześnie:

$$\cos\theta = \frac{x}{y} \iff y = \frac{x}{\cos\theta}$$

Więc aby igła przecinała krawędź musimy mieć:

$$l \ge 2y = \frac{2x}{\cos\theta}$$

czyli w takim razie musi być tak, że:

$$x \le \frac{l\cos\theta}{2}$$

Liczymy zatem nasze piękne prawdopodobieństwo:

$$\begin{aligned} \int_{\theta=0}^{\frac{\pi}{2}} \int_{x=0}^{\frac{l}{2}\cos\theta} \frac{4}{d\pi}\, dx\, d\theta = \frac{4}{d\pi} \int_{\theta=0}^{\frac{\pi}{2}} \int_{x=0}^{\frac{l}{2}\cos\theta} 1\, dx\, d\theta = \\ \frac{4}{d\pi} \int_{\theta=0}^{\frac{\pi}{2}} \frac{l}{2}\cos\theta\, d\theta = \\ \frac{2l}{d\pi} \int_{\theta=0}^{\frac{\pi}{2}} \cos\theta\, d\theta = \\ \frac{2l}{d\pi}\Big|_{\theta=0}^{\frac{\pi}{2}} \sin\theta = \\ \frac{2l}{d\pi} \cdot \left(\sin\frac{\pi}{2} - \sin 0\right) = \\ \frac{2l}{d\pi} \end{aligned}$$

□

5.5 Losowy spacer na płaszczyźnie

5.5.1 Pełne sformułowanie pytania

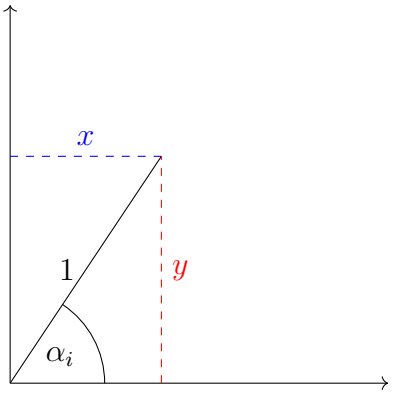
W tym przypadku treść pytania była zbyt długa by ją wpisać jako tytuł rozdziału, więc w pełnej formie przytoczymy ją dopiero tutaj.

Pytanie (Losowy spacer na płaszczyźnie)**.** Pionek znajduje się w punkcie $(0, 0)$ na płaszczyźnie. W i-tym kroku ($i > 1$) losowany jest kąt α_i jednostajnie na przedziale $[0, 2\pi]$ (i niezależnie od poprzednich losowań), a pionek przesuwa się ze swojej aktualnej pozycji o wektor jednostkowy wyznaczony przez kąt α_i (z osią OX). Jaki jest oczekiwany kwadrat odległości od punktu $(0, 0)$ po n krokach?

5.5.2 Rozwiązanie

Zamodelujemy nasz spacer nieco inaczej. Zamiast losować kąt z $[0, 2\pi]$, będziemy losować go z przedziału $[0, \frac{\pi}{2}]$ a następnie będziemy z prawdopodobieństwem $\frac{1}{2}$ „odbijać" nasz wektor jednostkowy w osi OY (i niezależnie od tego z prawdopodobieństwem $\frac{1}{2}$ „odbijać" tenże wektor jednostkowy w osi OX).

Wyliczmy na początek o ile przesuwa się pionek w osi Y jak i osi X, jeśli wylosowaliśmy kąt $\alpha_i \in [0, \frac{\pi}{2}]$. Sytuacja ma się wtedy następująco:



Wobec tego mamy, że:

$$\sin \alpha_i = \frac{y}{1} = y$$

$$\cos \alpha_i = \frac{x}{1} = x$$

Aby wyrazić formalnie dystans który przebywamy przez nas w i-tym kroku w osi OX (i OY) zdefiniujemy sobie 2 niezależne od siebie zmienne losowe:

1. D_{X_i}, która przyjmie 1 jeśli w i-tym kroku idziemy w prawo i -1 jeśli idziemy w lewo. Oczywiście, $\mathrm{P}(D_X = 1) = \frac{1}{2}$ i $\mathrm{P}(D_X = -1) = \frac{1}{2}$.
2. D_{Y_i}, którą definiujemy analogicznie jak D_{X_i}, ale dla i-tego kroku osi OY.

Możemy teraz policzyć dystans przebywany przez nas w i-tym kroku w osiach OX i OY:

$$X_i = D_{X_i} \cdot \cos \alpha_i$$

$$Y_i = D_{Y_i} \cdot \sin \alpha_i$$

Przypomnijmy sobie, że w pytaniu nas pytają o kwadrat odległości od punktu $(0, 0)$, czyli o wartość wyrażenia:

$$\left(\sum_{i=1}^{n} X_i\right)^2 + \left(\sum_{i=1}^{n} Y_i\right)^2$$

Rozpisujemy:

$$\begin{aligned}
\left(\sum_{i=1}^{n} X_i\right)^2 + \left(\sum_{i=1}^{n} Y_i\right)^2 &= \sum_{i=1}^{n} X_i^2 + \sum_{i<j} X_i X_j + \sum_{i=1}^{n} Y_i^2 + \sum_{i<j} Y_i Y_j \\
&= \sum_{i=1}^{n} D_{X_i}^2 \cos^2 \alpha_i + \sum_{i<j} D_{X_i} D_{X_j} \cos \alpha_i \cos \alpha_j + \\
+ \sum_{i=1}^{n} D_{Y_i}^2 \sin^2 \alpha_i + \sum_{i<j} D_{Y_i} D_{Y_j} \sin \alpha_i \sin \alpha_j &= \sum_{i=1}^{n} D_{X_i}^2 \cos^2 \alpha_i + D_{Y_i}^2 \sin^2 \alpha_i + \\
+ \sum_{i<j} D_{X_i} D_{X_j} \cos \alpha_i \cos \alpha_j + \sum_{i<j} D_{Y_i} D_{Y_j} \sin \alpha_i \sin \alpha_j &= \sum_{i=1}^{n} \cos^2 \alpha_i + \sin^2 \alpha_i + \\
+ \sum_{i<j} D_{X_i} D_{X_j} \cos \alpha_i \cos \alpha_j + \sum_{i<j} D_{Y_i} D_{Y_j} \sin \alpha_i \sin \alpha_j &
\end{aligned}$$

Jako, że znaną tożsamością trygonometryczną jest to, że $\sin^2 \alpha + \cos^2 \alpha = 1$, to pierwszy składnik musi wynosić n.

Pytamy się zatem o:

$$\mathbb{E}\left[n + \sum_{i<j} D_{X_i} D_{X_j} \cos \alpha_i \cos \alpha_j + \sum_{i<j} D_{Y_i} D_{Y_j} \sin \alpha_i \sin \alpha_j\right]$$

co możemy zapisać jako:

$$\mathbb{E}[n] + \mathbb{E}\left[\sum_{i<j} D_{X_i} D_{X_j} \cos \alpha_i \cos \alpha_j\right] + \mathbb{E}\left[\sum_{i<j} D_{Y_i} D_{Y_j} \sin \alpha_i \sin \alpha_j\right]$$

Korzystając z tego, że α_i są losowane niezależnie od siebie, podobnie jak D_{X_i} możemy to zapisać jako[4]:

$$n + \sum_{i<j} \mathbb{E}[D_{X_i}]\mathbb{E}\left[D_{X_j}\right]\mathbb{E}[\cos \alpha_i]\mathbb{E}[\cos \alpha_j] + \sum_{i<j} \mathbb{E}[D_{Y_i}]\mathbb{E}\left[D_{Y_j}\right]\mathbb{E}[\sin \alpha_i]\mathbb{E}[\sin \alpha_j]$$

Zauważamy, że $\mathbb{E}[D_{X_i}] = \mathbb{E}[D_{Y_i}] = \frac{1}{2} \cdot (-1) + \frac{1}{2} \cdot 1 = 0$

[4]Jeśli X i Y są niezależne, to $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$

Co upraszcza nasze wyrażenie do:

$$n + \sum_{i<j} 0 \cdot 0 \cdot \mathbb{E}[\cos \alpha_i]\mathbb{E}[\cos \alpha_j] + \sum_{i<j} 0 \cdot 0 \cdot \mathbb{E}[\sin \alpha_i]\mathbb{E}[\sin \alpha_j]$$

czyli w oczekiwaniu jesteśmy oddaleni od punktu $(0, 0)$ o n. Ale super.

5.6 Proces Poissona; definicja i potrzebne własności.

5.6.1 Pełna treść pytania

Dla porządku – podajmy pełną treść pytania, bo była zbyt długa by ją dać do nazwy sekcji.

Pytanie (Proces Poissona)**.** Definicja i potrzebne własności aby wykazać co następuje. Niech $(N(t), t_0)$ będzie procesem Poissona o parametrze λ. Wykazać, że jeśli w przedziale czasowym $(0, t]$ zaszło dokładnie jedno zdarzenie, czyli $N(t) = 1$ to czas zajścia tego zdarzenia X_1 ma rozkład jednostajny na przedziale $(0, t]$ (w szczególności nie zależy od λ).

5.6.2 Podstawowe własności procesu Poissona

Definicja 5.6.1. Procesem Poissona z parametrem λ nazywamy proces stochastyczny

$$\{N(t) \mid t \in \mathbb{R}, t \geq 0\}$$

(intuicyjnie: $N(t)$ mówi ile *jakichś* zdarzeń zaszło od momentu rozpoczęcia procesu do jakiejś chwili t.

taki, że:

1. $N(0) = 0$
2. Rozłączne przedziały są niezależne tj. zmienne
   $N(a) - N(b)$ i $N(c) - N(d)$ są niezależne dla $[b, a] \cap [d, c] = \varnothing$
3. Liczba zdarzeń na przedziałach jest stacjonarna tj.
   $N(t + s) - N(s)$ ma taki sam rozkład jak $N(t)$
4. Prawdopodobieństwo jednego zdarzenia w małym przedziale długości t zbiega do λ

   $$\lim_{t \to 0} \frac{P(N(t) = 1)}{t} = \lambda$$

5. Prawdopodobieństwo więcej niż jednego zdarzenia w małym przedziale zbiega do zera

   $$\lim_{t \to 0} \frac{P(N(t) > 1)}{t} = 0$$

Powyższa definicja nie jest jedyną możliwą definicją procesu Poissona. Okazuje się, że możemy skorzystać też z nieco wygodniejszej definicji bez tych dwóch limesów, ale za to korzystającej z rozkładu Poissona.

Pokażemy teraz dwa lematy, które dadzą nam równoważność między dwoma definicjami.

Twierdzenie 5.6.1 (Twierdzenie 8.7 P&C)**.** Niech $\{N(t) \mid t \geq 0\}$ będzie procesem Poissona z parametrem λ. Wtedy dla dowolnego $t \geq 0$ oraz $n \in \mathbb{N}$

$$P_n(t) = P(N(t) = n) = e^{-\lambda t}\frac{(\lambda t)^n}{n!}$$

Dowód. Zaczynamy od policzenia $P_0(t)$; dowód będzie indukcyjny.

Zauważamy, że z niezależności rozłącznych przedziałów mamy

$$P_0(t+h) = P_0(t) \cdot P_0(h)$$

Robimy więc pierwszą rzecz, która nam przychodzi do głowy tj. liczymy pochodną $P_0(t)$, a co.

$$\begin{aligned}
P_0'(t) &= \lim_{h\to 0} \frac{P_0(t+h) - P_0(t)}{h} \\
&= \lim_{h\to 0} P_0(t) \cdot \frac{P_0(h) - 1}{h} \\
&= \lim_{h\to 0} P_0(t) \cdot \frac{1 - P(N(h) = 1) - P(N(h) > 1) - 1}{h} \\
&= \lim_{h\to 0} \left(P_0(t) \cdot \left(\frac{-P(N(h) = 1)}{h} - \frac{P(N(h) > 1)}{h} \right) \right) \\
&= P_0(t) \cdot \left(-\lim_{h\to 0} \frac{P(N(h) = 1)}{h} - \lim_{h\to 0} \frac{P(N(h) > 1)}{h} \right) \\
&= P_0(t) \cdot (-\lambda - 0) \\
&= -\lambda P_0(t)
\end{aligned}$$

Wyniki poszczególnych limesów biorą się z własności 4 i 5 procesu Poissona.

Mamy zatem równanie różniczkowe

$$P_0'(t) = -\lambda P_0(t)$$

$$\frac{P_0'(t)}{P_0(t)} = -\lambda$$

Całkujemy po t i dostajemy

$$\ln P_0(t) = -\lambda t + C$$

$$P_0(t) = e^{-\lambda t + C}$$

Ponieważ $P_0(0) = 1$ to $C = 0$, czyli $P_0(t) = e^{-\lambda t}$. Tym samym bazę indukcji mamy udowodnioną.

Podobnie zabawny motyw dzieje się gdy obliczamy kolejne $P_n(t)$. Na początek zaobserwujmy jednak jedną rzecz.

Fakt 5.6.1.

$$P_n(t+h) = \sum_{k=0}^{n} P_{n-k}(t) \cdot P_k(h)$$

Dowód. Jeśli wiemy, że w czasie $t+h$ zaistniało n zdarzeń, to wiemy, że jakieś k (być może 0) zdarzeń musiało zaistnieć w czasie h, a więc $n-k$ zdarzeń zaistniało w czasie t. Aby policzyć prawdopodobieństwo takiej sytuacji wystarczy wymnożyć 2 takie prawdopodobieństwa (bo niezależność) a z racji tego że kolejne składniki sumy opisują zdarzenia które są rozłączne to zsumowanie jest legalne. □

Korzystając z wyżej wymienionego faktu, mamy:

$$\begin{aligned} P_n(t+h) &= \sum_{k=0}^{n} P_{n-k}(t) \cdot P_k(h) \\ &= P_n(t) \cdot P_0(h) + P_{n-1}(t) \cdot P_1(h) + \sum_{k=2}^{n} P_{n-k}(t) \cdot P(N(h)=k) \end{aligned}$$

Zrobiliśmy tu bardzo sprytną rzecz – mianowicie rozbiliśmy sumę na trzy części tak, aby przy liczeniu pochodnych wszystko nam się ładnie zwinęło.

$$\begin{aligned} P_n'(t) &= \lim_{h\to 0} \frac{P_n(t+h) - P_n(t)}{h} \\ &= \lim_{h\to 0} \left(\frac{P_n(t) \cdot P_0(h) + P_{n-1}(t) \cdot P_1(h) + \sum_{k=2}^{n} (P_{n-k}(t) \cdot P(N(h)=k)) - P_n(t)}{h} \right) \\ &= \lim_{h\to 0} \left(\frac{P_n(t) \cdot (P_0(h)-1) + P_{n-1}(t) \cdot P(N(h)=1) + \sum_{k=2}^{n} P_{n-k}(t) \cdot P(N(h)=k)}{h} \right) \\ &= \lim_{h\to 0} \left(\frac{P_n(t) \cdot (P_0(h)-1)}{h} + \frac{P_{n-1}(t) \cdot P(N(h)=1)}{h} + \sum_{k=2}^{n} P_{n-k}(t) \cdot \frac{P(N(h)=k)}{h} \right) \\ &= P_n(t) \lim_{h\to 0} \left(\frac{P_0(h)-1}{h} \right) + P_{n-1}(t) \lim_{h\to 0} \left(\frac{P(N(h)=1)}{h} \right) + \sum_{k=2}^{n} P_{n-k}(t) \cdot \lim_{h\to 0} \left(\frac{P(N(h)=k)}{h} \right) \\ &= P_n(t) \lim_{h\to 0} \left(\frac{1 - \mathrm{P}(N(h)=1) - \mathrm{P}(N(h)=2) - 1}{h} \right) + P_{n-1}(t) \cdot \lambda + \sum_{k=2}^{n} P_{n-k}(t) \cdot 0 \\ &= P_n(t) \left(- \lim_{h\to 0} \frac{\mathrm{P}(N(h)=1)}{h} - \lim_{h\to 0} \frac{\mathrm{P}(N(h)=2)}{h} \right) + \lambda P_{n-1}(t) \\ &= -\lambda P_n(t) + \lambda P_{n-1}(t) \end{aligned}$$

Znowu dostajemy równanie różniczkowe

$$\begin{aligned} P_n'(t) &= -\lambda P_n(t) + \lambda P_{n-1}(t) \\ P_n'(t) + \lambda P_n(t) &= \lambda P_{n-1}(t) \\ e^{\lambda t}(P_n'(t) + \lambda P_n(t)) &= \lambda e^{\lambda t} P_{n-1}(t) \\ e^{\lambda t} P_n'(t) + e^{\lambda t} \lambda P_n(t) &= \lambda e^{\lambda t} P_{n-1}(t) \\ \frac{d}{dt}\big(e^{\lambda t} \cdot P_n(t)\big) &= \lambda e^{\lambda t} P_{n-1}(t) \end{aligned}$$

I z założenia indukcyjnego:

$$\frac{d}{dt}\big(e^{\lambda t} \cdot P_n(t)\big) = \lambda e^{\lambda t} \cdot e^{-\lambda t} \cdot \frac{(\lambda t)^{n-1}}{(n-1)!} = \frac{\lambda^n \cdot t^{n-1}}{(n-1)!}$$

Całkujemy obustronnie:

$$\int \frac{d}{dt}\big(e^{\lambda t} P_n(t)\big)\, dt = e^{\lambda t} P_n(t) + C_1$$

$$\int \frac{\lambda^n \cdot t^{n-1}}{(n-1)!}\, dt = \frac{\lambda^n}{(n-1)!} \cdot \int t^{n-1}\, dt = \frac{\lambda^n}{(n-1)!} \cdot \frac{t^n}{n} + C_2 = \frac{\lambda^n t^n}{n!} + C_2$$

Definiujemy $C = C_2 - C_1$ by musieć mniej myśleć o stałych:

$$\begin{aligned} e^{\lambda t}P_n(t) + C_1 &= \frac{\lambda^n t^n}{n!} + C_2 \\ e^{\lambda t}P_n(t) &= \frac{\lambda^n t^n}{n!} + C_2 - C_1 \\ e^{\lambda t}P_n(t) &= \frac{\lambda^n t^n}{n!} + C \\ P_n(t) &= e^{-\lambda t}\frac{\lambda^n t^n}{n!} + Ce^{-\lambda t} \end{aligned}$$

Wiemy, że $P_n(0) = 0$, zatem $C = 0$. W takim razie:

$$P_n(t) = e^{-\lambda t}\frac{\lambda^n t^n}{n!} = e^{-\lambda t}\frac{(\lambda t)^n}{n!}$$

□

5.6.3 Warunkowe czasy zdarzeń

Możemy teraz przejść do udowadniania tego, co zostało nam nakazane w pytaniu (ale super)!

Wyobraźmy sobie zatem sytuację, gdzie w przedziale czasowym $(0, t]$ doszło do jakiegoś zdarzenia. Policzmy sobie prawdopodobieństwo, że w przedziale $(0, s]$ (dla jakiegoś $s \leq t$) *nie* doszło do tego zdarzenia.

Uwaga techniczna: jak będziemy teraz notować rzeczy typu $N(s)$ lub $N(t)$ będziemy mieli na myśli *ten sam proces*, a jako że przedziały których „dotyczą" te zmienne losowe się pokrywają to będą one od siebie zależne.

No to jedziemy z tematem.

$$\begin{aligned} \mathrm{P}(N(s) = 0 | N(t) = 1) &= \frac{\mathrm{P}(N(s) = 0 \wedge N(t) = 1)}{\mathrm{P}(N(t) = 1)} \\ &= \frac{\mathrm{P}(N(s) = 0 \wedge N(t-s) = 1)}{\mathrm{P}(N(t) = 1)} \end{aligned}$$

Tutaj uwaga: $N(s)$ i $N(t-s)$ będą już od siebie niezależne, bo te zmienne mają na celu opisywać niezależne od siebie przedziały. Tym samym możemy kontynuować nasze obliczenia:

$$\begin{aligned}\frac{\mathrm{P}(N(s)=0 \wedge N(t-s)=1)}{\mathrm{P}(N(t)=1)} &= \frac{\mathrm{P}(N(s)=0)\cdot \mathrm{P}(N(t-s)=1)}{\mathrm{P}(N(t)=1)} \\ &= \frac{e^{-\lambda s}\cdot \lambda(t-s)e^{-\lambda(t-s)}}{\mathrm{P}(N(t)=1)} \\ &= \frac{e^{-\lambda s}\lambda(t-s)e^{-\lambda t+\lambda s}}{\mathrm{P}(N(t)=1)} \\ &= \frac{\lambda(t-s)e^{-\lambda t}}{\mathrm{P}(N(t)=1)} \\ &= \frac{\lambda t e^{-\lambda t}-\lambda s e^{-\lambda t}}{\mathrm{P}(N(t)=1)} \\ &= \frac{\lambda t e^{-\lambda t}-\lambda s e^{-\lambda t}}{\lambda t e^{-\lambda t}} \\ &= 1-\frac{\lambda s e^{-\lambda t}}{\lambda t e^{-\lambda t}} \\ &= 1-\frac{s}{t}\end{aligned}$$

Czyli całkiem ewidentnie mamy tutaj do czynienia z rozkładem jednostajnym; w szczególności całkowicie obojętny jest mu parametr λ (co było do przewidzenia bo to rozkład jednostajny, ya know).

Żeby być całkowicie formalnymi w wykazaniu że to jest rozkład jednostajny to zdefiniujmy sobie zmienną losową X która jako wartość przyjmuje dokładny czas zdarzenia.

Widzimy, że dla $a \in (0,t]$:

$$\mathrm{P}(X<a) = \mathrm{P}(N(a)=0|N(t)=1) = 1-\frac{s}{t}$$

Zatem dystrybuanta jest taka:

$$F_X(a) = \mathrm{P}(X \geq a) = 1-\mathrm{P}(X<a) = 1-1+\frac{s}{t} = \frac{s}{t}$$

Czyli dystrybuanta zmiennej X jest taka sama jak dystrybuanta zmiennej losowej rozkładu jednostajnego, a to kończy dowód.

5.7 Centralne Twierdzenie Graniczne. Warianty mocniejszych wypowiedzi.

5.7.1 Rozkład normalny

Definicja 5.7.1. Definiujemy rozkład normalny (uogólniony) jako rozkład o następującej funkcji gęstości prawdopodobieństwa:

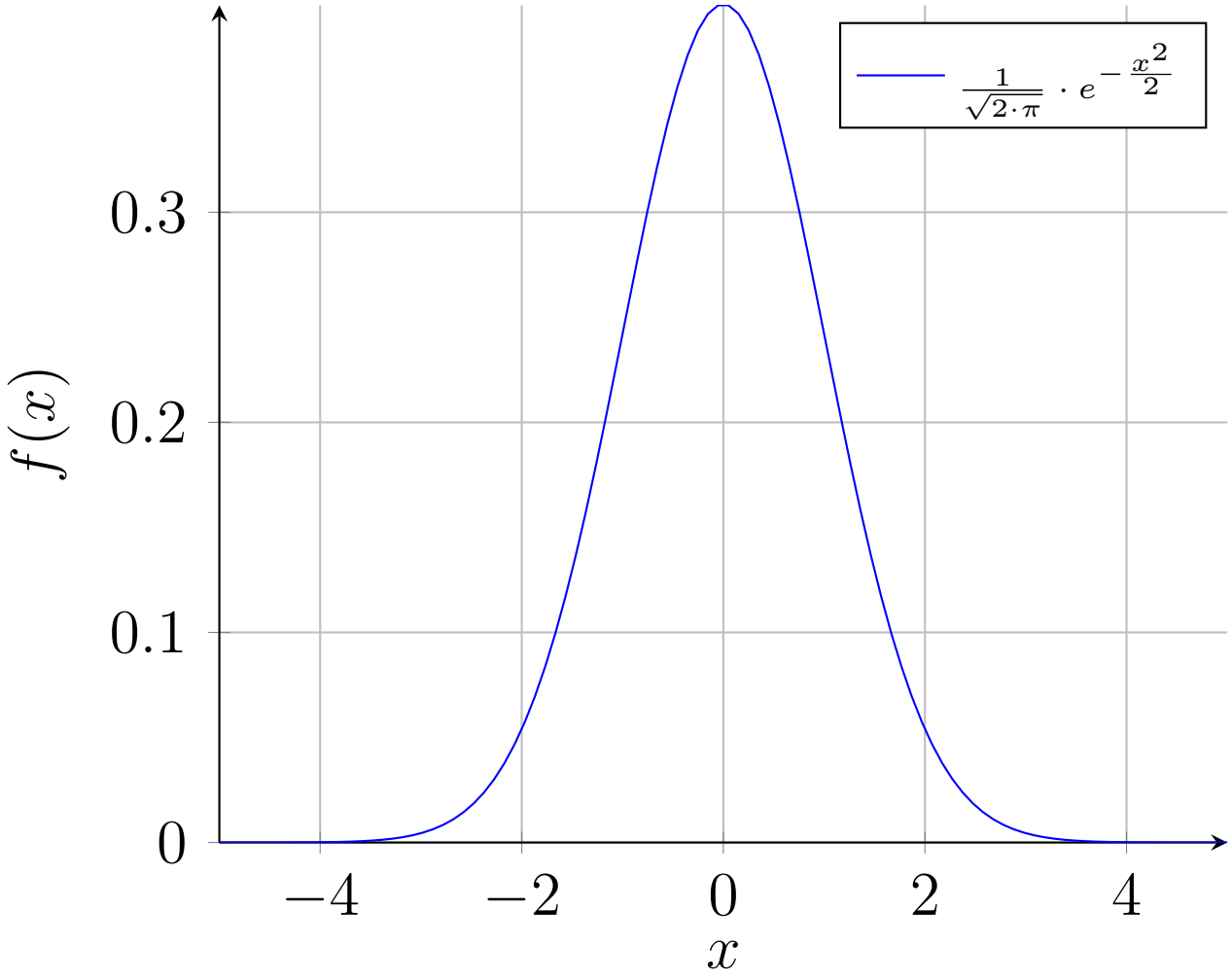
$$f_Z(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-((z-\mu)/\sigma)^2/2}$$

Definicja 5.7.2. Rozkład normalny o parametrach μ i σ^2 oznaczamy jako $N(\mu, \sigma^2)$.

Definicja 5.7.3. Standardowy rozkład Normalny to rozkład normalny o parametrach $\mu = 0$ i $\sigma^2 = 1$; oznaczamy go (bez większego szoku) jako $N(0, 1)$.

Definicja 5.7.4. Dystrybuantę standardowego rozkładu normalnego oznaczamy jako Φ.

Funkcja gęstości prawdopodobieństwa standardowego rozkładu normalnego wygląda jak ~~dzban~~ dzwon.



5.7.2 Centralne Twierdzenie Graniczne

Intuicyjnie: Centralne Twierdzenie Graniczne mówi, że jak mamy jakieś niezależne zmienne losowe (niekoniecznie o takim samym rozkładzie) to rozkład średnich tych wylosowanych wartości będzie zbiegać do rozkładu normalnego po wykonaniu pewnej, dużej liczby prób. Twierdzenie to uzasadnia występowanie w naturze rozkładu normalnego.

Definicja 5.7.5. Ciąg dystrybuant $F_1, F_2, ...$ zbiega w dystrybuancie do dystrybuanty F, co oznaczamy jako $F_n \to F$, jeśli dla każdego $a \in \mathbb{R}$ w którym F jest ciągła zachodzi:

$$\lim_{n\to\infty} F_n(a) = F(a)$$

Twierdzenie 5.7.1 (Centralne Twierdzenie Graniczne)**.** Niech $X_1, X_2, ...X_n$ będą niezależnymi zmiennymi losowymi o takim samym rozkładzie, średniej μ i wariancji σ^2. Niech $\overline{X_n} =$

$\frac{1}{n}\sum_{i=1}^{n} X_i$. Wówczas dla dowolnych a, b

$$\lim_{n\to\infty} \mathrm{P}\left(a \leq \frac{\overline{X_n} - \mu}{\sigma} \cdot \sqrt{n} \leq b\right) \to \Phi(b) - \Phi(a)$$

Dowód. Aby dowieść CTG, będziemy musieli przytoczyć *pomocne twierdzonko*, którego (mamy nadzieję) nikt nie będzie musiał dowodzić:

Twierdzenie 5.7.2 (Lévy-Cramér)**.** Niech $Y_1, Y_2, \ldots$ będzie sekwencją zmiennych losowych, gdzie Y_i ma dystrybuantę F_i i funkcję tworzącą momentów M_i. Niech Y będzie zmienną losową o dystrybuancie F i funkcji tworzącej momenty M. Jeżeli dla każdego t zachodzi:

$$\lim_{n\to\infty} M_n(t) = M(t)$$

to $F_n \to F$ dla wszystkich t w których $F(t)$ jest ciągła.

Dowód. Mitzenmacher przytacza to twierdzenie bez dowodu; na wykładzie go również nie było, a więc i my udowodnimy je poprzez założenie go jako aksjomat (haha). □

Przystępujemy teraz do dowodzenia CTG.

Definiujemy $Z_i = (X_i - \mu)/\sigma$. Wówczas Z_i są to niezależne zmienne losowe oraz:

$$\mathbb{E}[Z_i] = \mathbb{E}\left[\frac{X_i - \mu}{\sigma}\right] = \frac{1}{\sigma} \cdot (\mathbb{E}[X_i] - \mathbb{E}[\mu]) = \frac{1}{\sigma} \cdot (\mu - \mu) = 0$$

$$\mathrm{Var}[Z_i] = \mathrm{Var}\left[\frac{X_i - \mu}{\sigma}\right] = \frac{1}{\sigma} \cdot (\mathrm{Var}[X_i - \mu]) = \frac{1}{\sigma} \cdot (\mathrm{Var}[X_i] - \mathrm{Var}[mu]) = \frac{1}{\sigma^2} \cdot \left(\sigma^2 - 0\right) = 1$$

(gdzie korzystamy z faktu, że dla stałej c $\mathrm{Var}[cX] = c^2 \mathrm{Var}[X]$).

Ponadto mamy, że:

$$\frac{\bar{X_n} - \mu}{\sigma} \cdot \sqrt{n} = \frac{\sqrt{n}}{n} \sum_{i=1}^{n} \frac{X_i - \mu}{\sigma} = \frac{\sqrt{n}}{n} \sum_{i=1}^{n} Z_i = \frac{\sum_{i=1}^{n} Z_i}{\sqrt{n}}$$

Żeby zastosować teraz przywołane przez nas twierdzenie Levy'ego i tego drugiego musimy pokazać, że MGF[5] zmiennych losowych postaci

$$Y_n = \frac{\sum_{i=1}^{n} Z_i}{\sqrt{n}}$$

zbiega do MGF zmiennej losowej o standardowym rozkładzie normalnym. Po zastosowaniu tego twierdzenia dostalibyśmy już tezę Centralnego Twierdzenia Granicznego.

W takim razie, chcemy w tym celu pokazać coś takiego:

$$\lim_{n\to\infty} M_{Y_n}(t) = \lim_{n\to\infty} \mathbb{E}\left[e^{t\sum_{i=1}^{n} Z_i/\sqrt{n}}\right] = e^{t^2/2}$$

[5]Moment Generating Function

Niech $M_{Z_i}(t) = \mathbb{E}\left[e^{tZ_i}\right]$ będzie funkcją tworzącą momenty zmiennej Z_i. Zauważamy, że wówczas MGF zmiennej losowej $Z_i/\sqrt{n}$ wynosi:

$$M_{Z_i/\sqrt{n}}(t) = \mathbb{E}\left[e^{t\cdot Z_i/\sqrt{n}}\right] = M_{Z_i}\left(\frac{t}{\sqrt{n}}\right)$$

Ponieważ Z_i są niezależne i mają ten sam rozkład:

$$M_{Y_n}(t) = M_{\sum_{i=1}^n Z_i/\sqrt{n}}(t) = \left(M_{Z_i/\sqrt{n}}(t)\right)^n = \left(M_{Z_i}\left(\frac{t}{\sqrt{n}}\right)\right)^n$$

Teraz wykonujemy *magiczne założenie*. Zdefiniujmy sobie, **for no reason at all**, funkcję L, taką że:

$$L(t) = \ln M_{Z_i}(t)$$

Dodatkowo, *również bez jakiejkolwiek przyczyny*, policzmy sobie pierwszą i drugą pochodną $L(0)$.

Zacznijmy od trywialnych obserwacji:

$$M_{Z_i}(0) = 1 \implies L(0) = 0$$

$$L'(0) = (\ln M_{Z_i}(0))' = \frac{1}{M_{Z_i}(0)} \cdot M'_{Z_i}(0) = \frac{M'_{Z_i}(0)}{M_{Z_i}(0)} = \frac{\mathbb{E}[Z_i]}{1} = \mathbb{E}[Z_i] = 0$$

$$L''(0) = \frac{M_{Z_i}(0)M''_{Z_i}(0) - (M'_{Z_i}(0))^2}{(M_{Z_i}(0))^2} = \frac{M''_{Z_i}(0) - 0}{1} = \mathbb{E}\left[Z_i^2\right] = 1$$

W ostatnim przejściu korzystamy z faktu, że $\mathbb{E}[Z_i^2] = 1$. Wynika to z faktu, że $\mathbb{E}[Z_i^2] - \mathbb{E}[Z_i]^2 = \sigma^2 = 1$.

Przypomnijmy, że chcieliśmy pokazać, że:

$$\lim_{n\to\infty} M_{Y_n}(t) \to e^{t^2/2}$$

lub równoważnie:

$$\lim_{n\to\infty} \left(M_{Z_i}\left(\frac{t}{\sqrt{n}}\right)\right)^n \to e^{t^2/2}$$

po zlogarytmowaniu stronami:

$$\lim_{n\to\infty} nL\left(\frac{t}{\sqrt{n}}\right) \to \frac{t^2}{2}$$

Pytanie teraz co musimy zrobić by wykazać, że ta granica tyle wynosi.

Jak wszyscy wiemy, kiedy nie wiadomo jak policzyć granicę, to liczymy ją L'Hôpitalem. Zapiszmy więc sobie ten limit tak, byśmy mogli użyć tego twierdzenia (czyli żeby pojawił się symbol nieoznaczony $\frac{0}{0}$).

$$\lim_{n\to\infty} \frac{L\left(\frac{t}{\sqrt{n}}\right)}{n^{-1}}$$

No i lecimy z pochodnymi!

$$\begin{aligned}
\lim_{n\to\infty} \frac{L\left(\frac{t}{\sqrt{n}}\right)}{n^{-1}} &= \lim_{n\to\infty} \frac{-L'\left(\frac{t}{\sqrt{n}}\right) t n^{-\frac{3}{2}}}{-2n^{-2}} \\
&= \lim_{n\to\infty} \frac{L'\left(\frac{t}{\sqrt{n}}\right) t}{2n^{-\frac{1}{2}}} \\
&= \lim_{n\to\infty} \frac{-L''\left(\frac{t}{\sqrt{n}}\right) t^2 n^{-\frac{3}{2}}}{-2n^{-\frac{3}{2}}} \\
&= \lim_{n\to\infty} \frac{t^2 \cdot L''\left(\frac{t}{\sqrt{n}}\right)}{2} \\
&= \lim_{n\to\infty} \frac{t^2 \cdot 1}{2} \\
&= \frac{t^2}{2}
\end{aligned}$$

I w sumie to mieliśmy dowieść. Ale fajnie. □

5.7.3 Mocniejsze warianty CTG

Rozdział 6

Systemy operacyjne

6.1 Opisz mechanizmy komunikacji międzyprocesowej standardu POSIX.

Pisanie do plików - najprostszy mechanizm. Brak synchronizacji narzuconej przez system - trzeba dobrowolnie używać advisory file locking. Operacje odczytu i zapisu mogą być powolne. Można również zmapować dany plik do pamięci operacyjnej.

Named pipes - specjalny rodzaj pliku pozwalający na nawiązanie połączenia między procesami. Procesy otwierają taki plik w trybie do odczytu lub do zapisu i w ten sposób się komunikują (FIFO). Odczyt i zapis jest synchronizowany przez system. Gdy jest wielu czytających, to sytuacja jest nieokreślona. Może być wielu zapisujących - ich zapisy nie będą rodzielane, o ile nie przekraczają rozmiaru bufora systemu. Tak samo działają anonymous pipes, które nie są reprezentowane jako pliki na dysku, tylko są bezpośrednio tworzone jako pary deskryptorów plików.

Współdzielona pamięć - najszybszy mechanizm. Wiele procesów mających dostęp do tego samego fragmentu pamięci operacyjnej. Trzeba synchronizować zapis za pomocą mutexów lub innych mechanizmów. Procesy-dzieci współdzielą pamięć z procesami-rodzicami. Istnieje również mechanizm, za pomocą którego niezależne procesy mogą uzgodnić współdzieloną pamięć.

POSIX threads - W systemie posixowym, każdy proces może składać się z wielu podprocesów - wątków. Wątki domyślnie współdzielą całą pamięć procesu. Do komunikacji między nimi można używać tych samych mechanizmów co do komunikacji pomiędzy procesami, ale jest też dostępne kilka innych.

Sygnały - wysyłane do procesów o znanych PID lub PGID. Przerywają działanie procesów, które zazwyczaj mogą jakoś obsłużyć dany sygnał. Sygnały nie są kolejkowane, tylko ustawiają odpowiednie bity w tablicy jeszcze nieobsłużonych sygnałów. Można blokować sygnały, czyli odkładać ich obsługę na później, w szczególności można blokować sygnały na czas samej ich obsługi.

Unix sockets - używa się ich tak jak protokołów sieciowych, ale komunikacja odbywa się tylko w obrębie systemu. Jako adres jest używany specjalny plik. Działają one dwukierunkowo.

6.2 Porównaj mechanizmy szeregowania zadań na przykładzie serwera przetwarzającego zadania w trybie wsadowym, oraz systemu interaktywnego.

6.2.1 Serwer w trybie wsadowym

W trybie wsadowym zadania są wykonywane po kolei, w pewnej ustalonej kolejności.

Najprostszy sposób to **first-come first-served** - zwykłe kolejkowanie zadań. Zablokowane procesy przechodzą na koniec kolejki.

Można też próbować minimalizować sumaryczny turnaround time, czyli czas między przyjściem a zakończeniem zadania:

- **Shortest job first** - szacowanie czasów wykonywania zadań i wybieranie najkrótszego. Optymalny względem sumarycznego turnaround time, ale tylko w wersji offline.
- **Shortest remaining time next** - wybieranie zadania, które zakończy się najwcześniej. Pozwalamy na wywłaszczanie procesów. Optymalny względem sumarycznego turnaround time w wersji online, o ile nie uwzględniamy czasu na zmianę kontekstu. Turnaround time pojedynczego procesu może być potencjalnie bardzo duży.

6.2.2 System interaktywny

W systemie interaktywnym liczy się szybka reakcja na bodźce, więc, z punktu widzenia użytkownika, procesy są wykonywane w tym samym czasie.

Każdy proces dostaje **kwant** czasu (zazwyczaj kilkadziesiąt milisekund), w obrębie którego może nieprzerwanie wykonywać obliczenia. W trybie **round robin** procesy są ustawione w kolejce i wykonywany jest pierwszy proces w kolejce aż do wyczerpania swojego kwantu czasu. Po wykorzystaniu kwantu czasu, proces trafia na koniec kolejki i dostaje nowy kwant. Zablokowany proces nie traci swojego kwantu, a po odblokowaniu trafia na początek kolejki z pozostałym czasem.

Jeśli kwant czasu jest zbyt mały, to procesor traci zbyt dużo czasu na zmianę kontekstów, a jeśli jest zbyt duży, to czas reakcji systemu może być za długi.

W podstawowej wersji wszystkie procesy są traktowane równo. Można jednak wprowadzić **priority scheduling**, na przykład przydzielając różne kwanty czasu w zależności od priorytetów procesów, albo używając wielu kolejek round-robin dla różnych priorytetów. Priorytety mogą być przydzielane dynamicznie w zależności od tego, jak aktywny jest dany proces.

Fair share scheduling - wybieranie procesów nie po kolei, ale w taki sposób, żeby każdy proces lub użytkownik miał gwarantowany odpowiedni czas procesora.

6.3 Wyjaśnij pojęcie deadlock. Opisz metody wykrywania i zapobiegania powstawaniu deadlocku w kontekście współdzielonych zasobów.

Zbiór procesów jest zakleszczony (**deadlocked**), jeśli każdy proces w tym zbiorze czeka na zdarzenie, które może być spowodowane tylko przez jeden z procesów w tym zbiorze.

Jeśli przedstawimy procesy jako graf skierowany, gdzie krawędź oznacza czekanie na inny proces, oraz każdy proces czeka na co najwyżej jeden inny proces, to deadlock jest po prostu cyklem. Ogólniej, jeśli dany proces może czekać na dowolny z kilku innych procesów, to deadlock jest izolowaną silnie spójną składową.

Przykładowo, jeden proces blokuje dostęp do zasobu A i czeka na dostęp do zasobu B, podczas gdy drugi proces blokuje dostęp do B i czeka na A.

Wykrywanie deadlocków – sprawdzanie co jakiś czas grafu przydziału zasobów i wykrywanie cykli. Po wykryciu cyklu jeden z procesów w tym cyklu jest przerywany i restartowany.

Metody zapobiegania deadlocków:

- pozwalanie na żądanie wielu zasobów równocześnie i nie pozwalanie na żądanie kolejnych zasobów, gdy jakieś już są przydzielone
- pozwalanie na wywłaszczanie przydzielonych zasobów
- ponumerowanie zasobów i pozwalanie na żądanie ich tylko w odpowiedniej kolejności (przykład structural prevention)
- wymaganie od procesów uprzedniej deklaracji potencjalnie używanych zasobów i odpowiednie planowanie przydziału na podstawie tych informacji

6.4 Wyjaśnij mechanizm spooling, podaj przykłady kiedy należy oraz kiedy nie da się używać spoolingu jako mechanizmu racjonalizującego dostęp do zasobu.

Spooling (simultaneous peripheral operations on-line) - polega na nie używaniu danego zasobu bezpośrednio, tylko na przekazywaniu zadań dla tego zasobu do specjalnego procesu (**spooler**), który kolejkuje i wykonuje te zadania. Sposób ten polega na tym, że urządzenie nie udostępnia biblioteki procedur, za pomocą których można komunikować się z urządzeniem, ale zamiast tego tworzony jest specjalny proces działający w tle (**daemon**) oraz dedykowany folder (nie musi to być folder w systemie pilków) - **spooling directory**. Procesy użytkownika umieszczają dane w tym folderze (w kolejności FIFO), a daemon decyduje w jaki sposób zlecić je do "wykonaniańa urządzeniu.

Typowym przykładem jest drukarka, która działa zbyt wolno, żeby dany program używał jej bezpośrednio. Dokumenty są więc przekazywane do spoolera, który po kolei je drukuje. Dzięki temu programy mogą wykonywać inne czynności w trakcie drukowania.

Ogólnie, należy używać spoolingu w przypadku powolnych zasobów wyjściowych. Na pewno nie da się używać spoolingu w przypadku zasobów wejściowych i interaktywnych, wobec których programy muszą czekać na dane wejściowe.

6.5 Segmentacja i stronicowanie - porównaj mechanizmy. Opisz jak te mechanizmy są wykorzystywane na przykładzie wybranego systemu operacyjnego.

W przeciwieństwie do kernela, normalne procesy nie używają bezpośredniego, fizycznego adresowania pamięci, tylko wirtualną przestrzeń adresową, za pomocą której adresy w instrukcjach

są tłumaczone na adresy fizyczne przez podjednostkę CPU zwaną MMU (Memory Management Unit). Każdy proces ma swoją własną, odizolowaną przestrzeń adresową mapowaną do przestrzeni fizycznej.

6.5.1 Segmentacja

Segmentacja polega na podziale wirtualnej przestrzeni adresowej na segmenty przeznaczone do różnych zastosowań, np. segmenty na kod, stos i dane. Programy używające segmentacji muszą do każdego adresu dołączyć informację, do którego segmentu należy. Segmenty są mapowane do offsetów pamięci fizycznej, do których są dodawane używane adresy.

Segmentacja jest obecnie uznawana ze przestarzały mechanizm zastąpiony przez stronicowanie - w architekturze amd64 segmentacja nie jest w ogóle wspierana. Prawie wszystkie rozwiązania, które zapewnia segmentacja można dość łatwo częściowo zapewnić używając stronicowania. Na przykład, jeśli chcemy kilka łatwo rozszerzalnych liniowych przestrzeni adresowych, po prostu alokujemy bloki pamięci tych przestrzeni bardzo daleko od siebie w gigantycznej 64 bitowej wirtualnej przestrzeni adresowej.

6.5.2 Stronicowanie

Stronicowanie jest prostszym mechanizmem, w którym nie ma podziału na segmenty i jest tylko jedna wirtualna przestrzeń adresowa. Pamięć fizyczna jest dzielona na strony o rozmiarze zazwyczaj kilku kilobajtów, które są mapowane do niektórych stron w pamięci wirtualnej. Wobec tego danemu procesowi wydaje się, że jest więcej dostępnej pamięci niż fizycznie obecnej. Programy nie są świadome stronicowania i nie muszą dołączać do adresów dodatkowych informacji.

6.5.3 Segmentacja + Stronicowanie

Można również połączyć mechanizmy segmentacji i stronicowania – segmentacja nie tłumaczy wtedy bezpośrednio na adres fizyczny, tylko na adres, który jest w drugiej kolejności tłumaczony przez stronicowanie.

Ponieważ wirtualna przestrzeń adresowa jest bardzo duża, mapowanie stron jest zapisywane za pomocą tabel na kilku poziomach. Wirtualny adres jest dzielony na kilka części, z których każda określa numer wpisu w tabeli na odpowiednim poziomie lub numer bajtu w obrębie strony. Dla każdego adresu wirtualnego MMU odczytuje po kolei wpisy w odpowiednich tabelach i w ten sposób oblicza adres fizyczny. Żeby przyśpieszyć to tłumaczenie, MMU może cache'ować numery stron do TLB (Translation Lookaside Buffer).

Każdy proces ma swoją własną strukturę tabel stron, więc przy zmianie kontekstu jest zmieniany specjalny rejestr trzymający adres tej struktury. Zmiana tego rejestru wymaga też unieważnienia TLB, co negatywnie wpływa na wydajność.

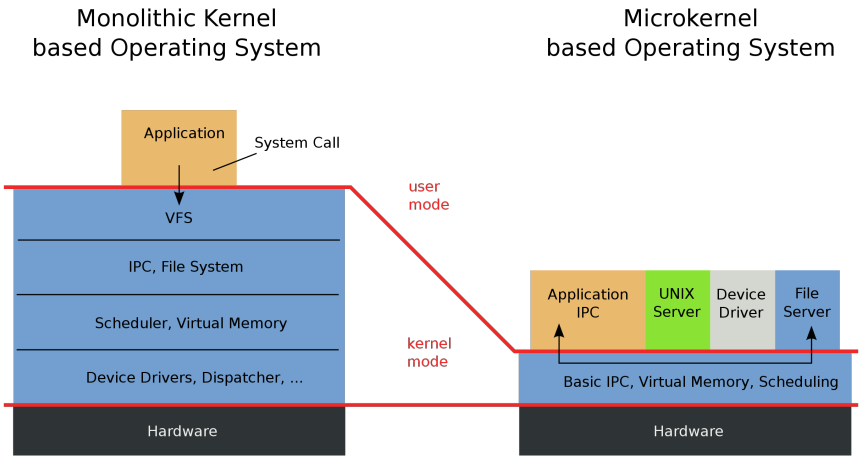
W Linuxie strony są mapowane na żądanie w następujący sposób:

- Stronicowanie jest używane do zaimplementowania pamięci wirtualnej. Każda strona pamięci może być też zapisana na dysku w pliku swap, albo partycji swap
- Współdzielenie pamięci pomiędzy procesami jest zaimplementowane używając stronicowania. Kod bibliotek współdzielonych jest ładowany do pamięci raz, a potem współdzielony pomiędzy wszystkimi procesami używając stronicowania

- Stronicowanie w Linuxie jest używane do wielu optymalizacji. Na przykład, gdy proces jest forkowany, to tylko jego tablica stron jest kopiowana, a nie cała pamięć
- Gdy proces próbuje uzyskać dostęp do adresu, który nie jest zmapowany w tabeli stron, to procesor generuje błąd stronicowania (page fault), który powoduje przekazanie sterowania kernelowi
- Kernel sprawdza, czy adres, do którego chciał uzyskać dostęp proces, jest poprawny.
- Jeśli adres nie jest poprawny, to kernel wysyła do procesu sygnał o błędzie, który domyślnie zatrzymuje proces
- Jeśli adres jest poprawny, to kernel uzupełnia odpowiednie informacje w tabeli stron i oddaje sterowanie procesowi w taki sposób, żeby ponownie wykonał tę instrukcję, która wcześniej spowodowała błąd stronicowania

6.6 Porównaj monolityczną architekturę systemu operacyjnego z architekturą opartą na mikro jądrze.

W każdym momencie procesor działa albo w trybie użytkownika, albo w trybie uprzywilejowanym. W trybie uprzywilejowanym dostępne są instrukcje dające pełną kontrolę nad komputerem, a w trybie użytkownika dostęp ten jest ograniczony.



Rysunek 6.1: Public domain, Wikimedia Commons; Model komunikacji z aplikacjami monolitu i mikro jądra

6.6.1 Monolit

W monolitycznej architekturze cały system operacyjny działa w trybie uprzywilejowanym. W szczególności, wszystkie sterowniki urządzeń działają w trybie uprzywilejowanym, więc usterka w dowolnym z nich psuje działanie całego systemu operacyjnego. Proces użytkownika wykonuje żądanie systemowe poprzez bezpośrednie przejście do trybu uprzywilejowanego, w którym odpowiedni kod kernela obsługuje to żądanie.

6.6.2 Mikro jądro

W architekturze opartej na mikro jądrze dąży się, aby tylko niezbędne minimum kodu było wykonywane w trybie uprzywilejowanym. Wiele zadań systemu operacyjnego, takie jak zarzą-

dzanie procesami czy obsługa systemów plików jest wykonywana przez serwery, czyli specjalne procesy, które działają w trybie użytkownika, ale mogą prosić kernel o wykonywanie dla nich odpowiednich czynności. W szczególności, sterowniki urządzeń są tylko zwykłymi serwerami. Dzięki podziale zadań na różne serwery, cały system operacyjny staje się bardziej modularny, a usterka w pojedynczym serwerze nie koniecznie powoduje zepsucia działania całego systemu. Procesy użytkownika i serwery nigdy nie działają w trybie uprzywilejowanym. Procesy użytkownika wykonują żądania systemowe poprzez przekazywanie wiadomości do odpowiednich serwerów, a serwery realizują te żądania poprzez przekazywanie odpowiednich wiadomości do kernela. Tylko kernel działa w trybie uprzywilejowanym i musi być odpowiedzialny przynajmniej za przekazywanie wiadomości między procesami, mapowanie wirtualnej pamięci, przełączanie procesów, oraz bezpośrednią komunikację z urządzeniami.

6.7 Przedstaw mechanizm współdzielenia bibliotek programistycznych (w systemie Linux), uwzględniając odpowiednie metody adresowania

Biblioteki programistyczne mogą być używane albo w sposób statyczny, albo dynamiczny. **Linkowanie statyczne** powoduje załączenie kodu biblioteki do samego programu w trakcie kompilacji. **Linkowanie dynamiczne** powoduje ładowanie kodu biblioteki dopiero w trakcie działania programu, odczytując z pliku, który może być również wykorzystywany przez inne programy.

Aby biblioteka mogła być linkowana dynamicznie, musi używać adresowania względnego, czyli być skompilowana do **position independent code**. Dzięki temu może być zmapowana do dowolnego miejsca w pamięci wirtualnej danego procesu, oraz różne procesy mogą wybierać dla niej różne miejsca.

Dzięki linkowaniu dynamicznemu jest oszczędzana przestrzeń dyskowa oraz pamięć operacyjna, ponieważ ten sam kod jest wykorzystywany przez wiele programów. Kernel może zmapować ten sam fragment pamięci fizycznej zawierający kod biblioteki do adresów wirtualnych wielu procesów.

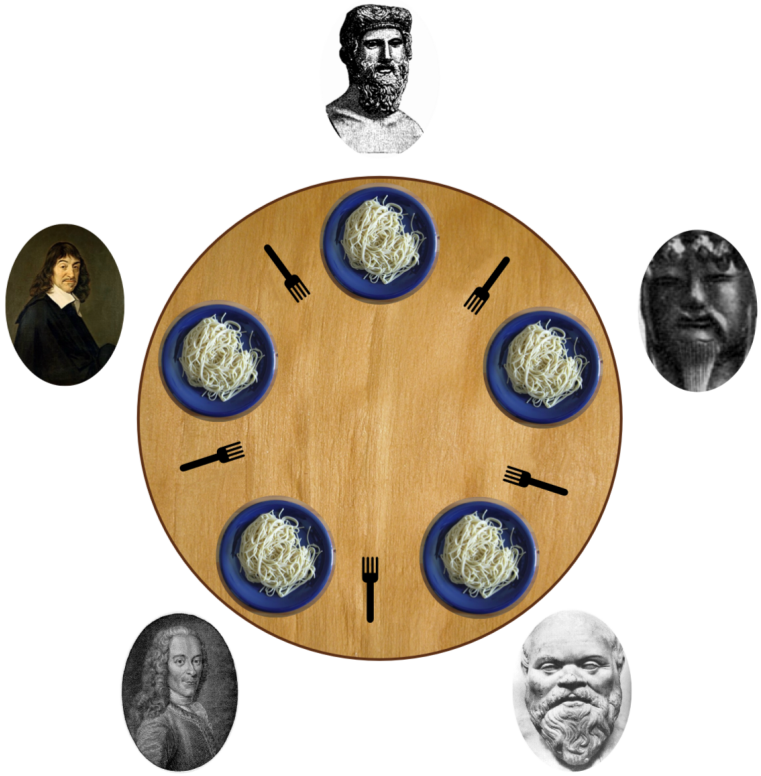
Na Linuxie, programy używające biblioteki dynamiczne są uruchamiane za pomocą specjalnego programu *ld-linux.so*, który poprzedza ich uruchomienie poprzez znalezienie i załadowanie potrzebnych bibliotek.

Biblioteki dynamiczne mogą być ładowane w sposób leniwy – funkcja z biblioteki jest początkowo przekierowaniem do kodu dynamic linkera, który zastępuje cel tego przekierowania odpowiednim załadowanym kodem biblioteki. Jest to osiągane poprzez **Procedure Linkage Table**. Dla każdej importowanej funkcji tworzone są odpowiednie informacje w tabeli PLT powiązane z **Global Offset Table** (GOT).

W GOT dane publiczne (eksportowane) traktowane są, jak zewnętrzne. Przestrzeń dla nich jest alokowana na stercie, nie zależą od IP, a więc znajdują się w nieznanych miejscach w pamięci. Aby dostęp do nich był możliwy, tworzona jest tablica z adresami danych zewnętrznych. Jest ona uzupełniana przez dynamic-linkera i znajduje się w sekcji danych biblioteki.

6.8 Na przykładzie problemu ucztujących filozofów przedyskutuj pojęcia poprawności pod względem bezpieczeństwa i żywotności. Zaproponuj rozwiązanie spełniające oba te warunki.

W problemie ucztujących filozofów pewna liczba filozofów siedzi przy okrągłym stole i każdy z nich na zmianę albo myśli, albo je. Między filozofami na stole są położone widelce, po jednym między każdą parą filzofów. Aby jeść, filozof musi najpierw podnieść dwa widelce z obu stron. Każdy widelec może być w danym momencie podniesiony lub używany przez co najwyżej jednego filozofa. Problem polega na zaprojektowaniu takiej procedury postępowania dla każdego filozofa (czyli algorytmu rozproszonego), żeby mógł on w nieskończoność na zmianę myśleć i jeść.



Rysunek 6.2: bdesham, User:Belbury, CC BY 3.0, `https://creativecommons.org/licenses/by/3.0`, Wikimedia Commons; Ilustracja problemu filozofów

Algorytm może być niepoprawny pod względem **bezpieczeństwa**, jeśli może doprowadzić do deadlocka, na przykład gdy w tym samym momencie każdy filozof podniesie lewy widelec i zacznie czekać na dostępność prawego widelca.

Algorytm może być niepoprawny pod względem **żywotności**, jeśli może dojść do takiej sytuacji, że któryś filozof nigdy nie doczeka się obu widelców. Rozwiązanie polegające na numeracji widelców i podnoszeniu ich tylko w odpowiedniej kolejności rozwiązuje problem deadlocków, ale nie gwarantuje żywotności, ponieważ widelec o najmniejszym numerze jest podnoszony w pierwszej kolejności przez dwóch filozofów.

Rozwiązanie:

- Każdy widelec jest w każdym momencie przez kogoś trzymany.
- Początkowy przydział widelców nie jest symetryczny, czyli nie jest tak, że każdy trzyma prawy widelec lub każdy trzyma lewy widelec.
- Każdy filozof po zakończeniu jedzenia przekazuje równocześnie oba widelce do obu sąsiadów

Utrzymujemy niezmiennik, że przydział widelców nie jest symetryczny. Przydział nie jest symetryczny wtedy i tylko wtedy, gdy przynajmniej jeden filozof trzyma oba widelce. Po zjedzeniu, filozof oddaje równocześnie oba widelce, więc nowy przydział też nie jest symetryczny.

Skoro przydział nigdy nie jest symetryczny, to zawsze przynajmniej jeden filozof może jeść, więc deadlock jest niemożliwy.

Załóżmy nie wprost, że któryś filozof X nigdy nie doczeka się widelca od sąsiada Y. Zatem Y trzyma widelec po stronie X i sam nigdy nie doczeka się widelca z drugiej strony, bo gdyby się doczekał, to w końcu by zjadł i oddał widelec sąsiadowi X. Kontynuując w ten sposób można pokazać indukcyjnie że przydział jest symetryczny, co jest sprzeczne z niezmiennikiem. Zatem każdy filozof kiedyś doczeka się obu widelców.

Rozdział 7

Modele Obliczeń

7.1 Omów hierarchię Chomsky'ego języków, ilustrując klasy i zależności między nimi odpowiednimi przykładami

7.1.1 Hierarchia Chomsky'ego

7.1.1.1 Języki regularne

Definicja 7.1.1. Język REG_Σ definiujemy jako najmniejszy język nad alfabetem $\Sigma' = \Sigma \cup \{+, \cdot, ^*, 0, 1, (,)\}$, który spełnia następujące warunki:

- $0, 1 \in \mathrm{REG}_\Sigma$
- $\forall_{a \in \Sigma} : a \in \mathrm{REG}_\Sigma$
- $\forall_{d_1, d_2 \in \mathrm{REG}_\Sigma} : (d_1 + d_2) \in \mathrm{REG}_\Sigma$
- $\forall_{d_1, d_2 \in \mathrm{REG}_\Sigma} : (d_1 \cdot d_2) \in \mathrm{REG}_\Sigma$
- $\forall_{d \in \mathrm{REG}_\Sigma} : d^* \in \mathrm{REG}_\Sigma$

Definicja 7.1.2. Wyrażenie regularne nad alfabetem Σ definiujemy jako element języka REG_Σ.

Warto zaznaczyć, że operujemy jedynie na napisach bez żadnej intepretacji – symbole są jedynie symbolami, które będziemy zaraz interpretować odpowiednio.

Możemy teraz sobie dobrać do tego interpretację $L : \mathrm{REG}_\Sigma \to \mathcal{P}(\Sigma^*)$

- $L(0) = \varnothing$
- $L(1) = \{\varepsilon\}$
- $L(a) = \{a\}$
- $L(\alpha + \beta) = L(\alpha) \cup L(\beta)$
- $L(\alpha \cdot \beta) = L(\alpha) L(\beta)$
- $L(\alpha^*) = L(\alpha)^*$

Stosując tę dosyć naturalną interpretację będziemy opisywać sobie języki. **Języki które można opisać za pomocą wyrażenia regularnego nazywamy językami regularnymi**.

7.1.1.2 Języki bezkontekstowe

Definicja 7.1.3. Gramatyka bezkontekstowa (Context-Free Grammar) to czwórka $G = (N, \Sigma, P, S)$ gdzie

- N to skończony zbiór zmiennych (nieterminale)
- Σ - alfabet (terminale)
- P - produkcje $P \subseteq N \times (N \cup \Sigma)^*$
- $S \in N$ - symbol startowy

Definicja 7.1.4. Forma zdaniowa to dowolne słowo nad $(N \cup \Sigma)^*$

Intuicyjnie – gramatyka bezkontekstowa definiuje zasady na podstawie których możemy tworzyć nowe słowa nad alfabetem Σ. Nieterminale N są takimi stanami pośrednimi, które możemy podmieniać używając produkcji z P. Każda produkcja jest postaci

$$\text{nieterminal} \to \text{forma zdaniowa}$$

Definicja 7.1.5. Dla gramatyki G definiujemy relację $\to_G$ na formach zdaniowych. Mówimy, że

$$\alpha \to_G \beta$$

wtedy i tylko wtedy gdy:

$$\exists_{\alpha_1,\alpha_2,\gamma \in (N \cup \Sigma)^*} : \exists_{A \in N} : (A, \gamma) \in P \wedge \alpha = \alpha_1 A \alpha_2 \wedge \beta = \alpha_1 \gamma \alpha_2$$

Bardziej po ludzku – formę zdaniową β możemy uzyskać (w jednym kroku) z formy zdaniowej α o ile w α jest jakieś wystąpienie nieterminala A, które możemy zgodnie z produkcjami podmienić na formę zdaniową γ aby uzyskać β.

Definicja 7.1.6. $\to_G^*$ to zwrotne i przechodnie domknięcie $\to_G$

Z α możemy uzyskać β w dowolnej liczbie kroków, jeśli istnieje ciąg przekształceń $\alpha \to_G \alpha_1 \to_G \cdots \to_G \gamma$

Definicja 7.1.7. Język generowany przez gramatykę G to

$$L(G) = \{w \in \Sigma^* \mid S \to_G^* w\}$$

podobnie dla $A \in N$

$$L(G, A) = \{w \in \Sigma^* \mid A \to_G^* w\}$$

Definicja 7.1.8. Język jest bezkontekstowy (Context-Free Language) jeśli jest generowany przez jakąś gramatykę bezkontekstową.

7.1.1.3 Języki kontekstowe

Definicja 7.1.9. Gramatyka kontekstowa (Context-Sensitive Grammar) to czwórka $G = (N, \Sigma, P, S)$ gdzie

- N to skończony zbiór zmiennych (nieterminale)
- Σ - alfabet (terminale)
- P - produkcje $P \subseteq (N \cup \Sigma)^* N (N \cup \Sigma)^* \times (N \cup \Sigma)^*$

- $S \in N$ - symbol startowy

gdzie każda produkcja jest postaci:

$$\alpha_1 A \alpha_2 \to_G \alpha_1 \gamma \alpha_2$$

gdzie $\alpha_1, \alpha_2, \gamma \in (N \cup \Sigma)^*$ oraz $A \in N$ oraz $|\gamma| \geq 1$
Praktycznie wszystko co poznaliśmy z bezkontekstowych znajduje się też tutaj.

Definicja 7.1.10. Język generowany przez gramatykę G to oczywiście

$$L(G) = \{w \in \Sigma^* \mid S \to_G^* w\}$$

Definicja 7.1.11. Język jest kontekstowy (Context-Sensitive Language) jeśli jest generowany przez jakąś gramatykę kontekstową.

7.1.1.4 Maszyna Turinga

Definicja 7.1.12. Maszyna Turinga to tupla

$$MT = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$$

gdzie

- Q – skończony zbiór stanów
- $\Sigma \subseteq \Gamma$ – skończony alfabet wejściowy
- Γ – skończony alfabet taśmowy
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{-1, +1\}$ – stan, litera $\to$ nowy stan, zmiana litery, ruch głowicą (lewo, prawo)
- q_0 – stan startowy
- $\sqcup \in \Gamma \setminus \Sigma$ – pusty symbol / zero (domyślny symbol taśmy)
- F – zbiór stanów akceptujących

Ponadto zakładamy, że $Q \cap \Gamma = \varnothing$.

Definicja 7.1.13. Opis chwilowy (konfiguracja) Maszyny Turinga to słowo nad językiem $\Gamma^* q \Gamma \Gamma^*$.

Definicja 7.1.14. Konfiguracja startowa Maszyny Turinga to słowo postaci $q_0 w \sqcup$.

Definicja 7.1.15. Definiujemy relację $\vdash_M$ na konfiguracjach MT.

Przejście w lewo „wewnątrz taśmy”:

$$\alpha A q B \beta \vdash_M \alpha q' A B' \beta$$

gdzie $\alpha, \beta \in \Gamma^*$, $A, B \in \Gamma$, $q, q' \in Q$, oraz $\delta(q, B) = (q', B', -1)$.

Analogicznie definiujemy przejście w prawo „wewnątrz taśmy”:

$$\alpha q B \beta \vdash_M \alpha B' q' \beta$$

gdzie $\alpha, \beta \in \Gamma^*$, $B \in \Gamma$, $q, q' \in Q$, oraz $\delta(q, B) = (q', B', 1)$.

Definiujemy również „skrajne" przejścia – przejście w prawo „na skraju taśmy":

$$qB \vdash_M B'q'\sqcup$$

gdzie $B, B' \in \Gamma$, $q, q' \in Q$, oraz $\delta(q, B) = (q', B', 1)$

Przejście w lewo „na skraju taśmy":

$$qB \vdash_M q' \sqcup B'$$

gdzie $B, B' \in \Gamma$, $q, q' \in Q$, oraz $\delta(q, B) = (q', B', -1)$

Definicja 7.1.16. Język akceptowany przez Maszynę Turinga definiujemy jako

$$L(M) = \left\{w \in \Sigma^* \mid \exists_{q_f \in F} \exists_{\alpha,\beta \in \Gamma^*} q_0 w \sqcup \vdash_M^* \alpha q_f \beta\right\}$$

7.1.1.5 Języki rekurencyjnie przeliczalne

Definicja 7.1.17. Mówimy że język L jest **rekurencyjnie przeliczalny** jeśli istnieje Maszyna Turinga M taka że $L(M) = L$.

Zbiór języków rekurencyjnie przeliczalnych oznaczamy przez RE.

7.1.1.6 Języki rekurencyjne

Uwaga. Przynajmniej według Wikipedii, języki rekurencyjne nie wchodzą do hierarchii Chomsky'ego. Języki rekurencyjnie przeliczalne już za to tak. Proszę na to uważać, żeby ktoś się nie wykopał na egzaminie.

Definicja 7.1.18. Mówimy, że MT M ma **własność stopu** jeśli po skończonej liczbie kroków trafia do jednego z wyróżnionych stanów q_{acc}, q_{rej} z których jedyne przejścia prowadzą z powrotem do nich samych.

Powyższa definicja działa zarówno w przypadku deterministycznym (w którym mamy jedną ścieżkę obliczeń) jak i niedeterministycznym (wtedy chcemy żeby każda ścieżka obliczeń miała taką własność). Niestety formalnie Maszyna Turinga się nie zatrzymuje, ale przyjmujemy że jak już wpadniemy w stan q_{acc} to akceptujemy słowo i się zatrzymujemy a gdy wpadniemy w stan q_{rej} to odrzucamy słowo i się zatrzymujemy.

Definicja 7.1.19. Mówimy że język L jest **rekurencyjny** albo **rozstrzygalny** jeśli istnieje Maszyna Turinga z własnością stopu M taka że $L(M) = L$.

Zbiór języków rekurencyjnych oznaczamy przez R.

7.1.2 Zawierania

7.1.2.1 Języki regularne i bezkontekstowe

Twierdzenie 7.1.1. Zbiór języków regularnych nad danym alfabetem jest istotnym podzbiorem zbioru języków bezkonstekstowych nad tym samym alfabetem, tzn. $\mathrm{REG}_\Sigma \subsetneq \mathrm{CFL}_\Sigma$

Dowód. Dowodzić tej tezy możemy na 2 sposoby:

1. Skorzystać z faktu, że ewidentnie da się symulować DFA przy pomocy PDA, a dowodziliśmy na modelach że jeśli istnieje DFA dla języka to ten jest regularny (i że jeśli istnieje PDA dla języka to jest on bezkontekstowy);

2. dowieść to bez korzystania z tego faktu.

Istotność zawierania w obu przypadkach będzie wynikać z przykładu, który znajdzie się w sekcji 7.1.3.2.

W tym dowodzie podejmiemy się tego drugiego, bo nie ma tak prosto. Dowodzić będziemy stosując indukcję strukturalną po wyrażeniu regularnym i pokazując, że można je „skonwertować" w gramatykę bezkontekstową.

Na początek rozpatrujemy przypadek bazowy – jeśli mamy wyrażenie regularne α postaci a dla jakiegoś $a \in \Sigma$, to, oczywiście, $L(\alpha) = a$. Konstrukcja takiej CFG która rozpozna ten język jest trywialna – wystarczy mieć jedną produkcję postaci $S \to a$.

Teraz wykonujemy krok indukcyjny – załóżmy, że mamy regexa postaci:

1. $\alpha_1 + \alpha_2$ – z założenia indukcyjnego istnieją jakieś G_1, G_2 takie, że $L(G_1) = L(\alpha_1)$ i $L(G_2) = L(\alpha_2)$. Gramatyki G_1, G_2 mają jakieś symbole startowe S_1, S_2 – robimy gramatykę G_s z dwoma produkcjami:
   (a) $S \to S_1$
   (b) $S \to S_2$

   gdzie S to symbol startowy gramatyki G_s. Zauważamy, że $L(G_s) = L(\alpha_1 + \alpha_2)$, czyli jest super.
2. $\alpha_1\alpha_2$ – takie same założenia jak wcześniej, przy czym teraz gramatyka G_s wygląda tak:
   (a) $S \to S_1 S_2$
3. α_1^* – rozumowanie jak wcześniej, ale produkcje gramatyki G_s wyglądają tak:
   (a) $S \to S_1 S$
   (b) $S \to \varepsilon$

No i w sumie to będzie na tyle. Robimy gramatykę z regexa i super. □

7.1.2.2 Języki bezkontekstowe i kontekstowe

Twierdzenie 7.1.2. Zbiór języków bezkontekstowych nad określonym alfabetem jest istotnym podzbiorem zbioru języków kontekstowych nad tym samym alfabetem, tzn. $\mathrm{CFL}_\Sigma \subsetneq \mathrm{CSL}_\Sigma$.

Dowód. Istotność zawierania będzie wynikać z sekcji 7.1.3.3, gdzie pokażemy przykład języka kontekstowego który nie jest bezkontekstowy. Samo natomiast zawieranie wynika z faktu, że każdą gramatykę bezkontekstowa jest również gramatyką kontekstową, bo każdą produkcję z gramatyki bezkontekstowej postaci:

$$A \to \beta$$

można zapisać w gramatyce kontekstowej jako:

$$\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$$

gdzie $\alpha_1 = \alpha_2 = \varepsilon$. □

7.1.2.3 Języki kontekstowe i rekurencyjne

Twierdzenie 7.1.3. Zbiór języków kontekstowych nad określonym alfabetem jest istotnym podzbiorem zbioru języków rekurencyjnych nad tym samym alfabetem, tzn. $\mathrm{CSL}_\Sigma \subsetneq \mathrm{R}_\Sigma$.

Dowód. LBA to w szczególności Maszyna Turinga, ale z limitacją na rozmiar taśmy. W takim razie jeśli LBA rozpozna jakiś język, to MT też go rozpozna[1].

Przykład języka, który jest rozpoznawany przez MT ale nie LBA prezentujemy w sekcji 7.1.3.4 □

7.1.2.4 Języki rekurencyjne i rekurencyjnie przeliczalne

Twierdzenie 7.1.4. Zbiór języków rekurencyjnych nad określonym alfabetem jest istotnym podzbiorem zbioru języków rekurencyjnie przeliczalnych nad tym samym alfabetem, tzn. $\mathrm{R} \subsetneq \mathrm{RE}$.

Dowód. Każda Maszyna Turinga z własnością stopu jest Maszyną Turinga, więc ten. Natomiast język stopu jest przykładem języka który jest rozpoznwany jedynie przez Maszyny Turinga bez własności stopu. Więcej na ten temat jest w sekcji 7.8.2. □

7.1.3 Przykłady języków

7.1.3.1 Przykład języka regularnego

Twierdzenie 7.1.5. Język $\{a^n : n \in \mathbb{N}\}$ jest regularny.

Dowód. Tworzymy wyrażenie regularne postaci a^*. □

7.1.3.2 Przykład języka bezkontekstowego

Twierdzenie 7.1.6. Język $\{a^n b^n : n \in \mathbb{N}\}$ jest bezkontekstowy.

Dowód. Tworzymy gramatykę bezkontekstową z następującymi produkcjami:

1. $S \to aSb$
2. $S \to \varepsilon$

gdzie S to symbol startowy.

□

Ponadto, korzystając z lematu o pompowaniu dla języków regularnych (sekcja 7.2.1) możemy pokazać następujące twierdzenie:

Twierdzenie 7.1.7. Język

$$L = \{a^n b^n : n \in \mathbb{N}\}$$

nie jest językiem regularnym.

[1] Zaniepokojonych problemem stopu dla LBA informujemy, że problem stopu dla LBA jest rozstrzygalny; rozstrzygalność jego wynika z prostej rzeczy: mianowicie konfiguracji LBA jest skończenie wiele.

Dowód. Będziemy usiłowali pokazać, że L *nie* spełnia lematu o pompowaniu. To oznacza, że będziemy musieli pokazać, że zachodzi dla niego zaprzeczenie własności opisanej w tym lemacie. Innymi słowy, chcemy pokazać że:

$$\forall_{n>0}\exists_{w\in L}\forall_{xyz=w:|xy|\leq n\wedge|y|\geq 1}\exists_{i\in N} xy^iz \notin L$$

Prościej o tym myśleć jak o pewnej grze: nasz przeciwnik daje nam stałą n; my odpowiadmy mu słowem; ten nam je dzieli na 3 części które spełniają warunki lematu, a my możemy „wypompować" część y. Wygrywamy jeśli słowo nie należy do L. Jeśli mamy tutaj strategię wygrywającą, to znaczy że język nie jest regularny.

W przypadku naszego języka: nasz oponent daje nam jakieś n. Pora się odegrać, odegrać słowem postaci $a^{2n}b^{2n}$. Słowo to niewątpliwie należy do L, więc mogliśmy to zrobić. Przeciwnik musi je jakoś podzielić, tak by $w = xyz$, $|xy| \leq n$ i $y \geq 1$. Widzimy tutaj bardzo śmieszną rzecz: niezależnie od podziału, $xy = a^k$ dla jakiegoś k. Wobec tego również $y = a^l$ dla niezerowego l, które jest mniejsze niż n. W takim razie „depompujemy" y, ustawiając i na 0 i otrzymując słowo postaci $a^{2n-l}b^{2n}$, które z pewnością nie należy do języka. Hehe. □

7.1.3.3 Przykład języka kontekstowego

Twierdzenie 7.1.8. Język $\{a^nb^nc^n : n \in \mathbb{N}\}$ jest kontekstowy.

Dowód. Dowód będziemy realizować poprzez gramatykę kontekstową, ale na marginesie: można też pokazać LBA które rozpoznaje ten język. Jeśli wierzymy że LBA może mieć wiele taśm (co ma sens, ale trudno to pokazać *nieskazitelnie formalnie*) to po prostu możemy mieć 3 liczniki na taśmach które, ya know, policzą ile jest literek. I zweryfikują że są w takiej „formacji" jak powinny być. Jeśli wolimy nie dotykać tematu wielotaśmowych LBA z obaw o formalizm, możemy zrobić LBA które w każdym przejściu usuwa z taśmy literkę a, potem literkę b i c i pilnuje ich odpowiedniej „formacji" stanami.

Na początek zauważamy, że w gramatykach kontekstowych efektywnie legalne są przejścia typu $AB \to BA$, gdzie A, B są nieterminalami. Wynika to z faktu, że możemy wprowadzić dodatkowe nieterminale (nazwijmy je Z_1, Z_2) i wprowadzić następujące produkcje:

1. $AB \to Z_1B$
2. $Z_1B \to Z_1Z_2$
3. $Z_1Z_2 \to AZ_2$
4. $AZ_2 \to AB$

Oczywiście: dla każdej pary nieterminali A, B którą chcemy „przemieniać" konieczna jest oddzielna para Z_1, Z_2 do implementacji takiego przejścia.

Tworzymy CSG o następujących produkcjach (nieterminale to S, A, B, C, D, E; symbol startowy to S), :

1. $S \to DABCE$
2. $E \to ABCE$
3. $E \to \varepsilon$
4. $DA \to Da$
5. $Da \to a$

6. $aA \to aa$
7. $aB \to ab$
8. $bB \to bb$
9. $bC \to bc$
10. $cC \to cc$
11. $BA \to AB$
12. $CB \to BC$
13. $CA \to AC$

Omówmy sobie semantykę symboli których będziemy chcieli używać:

- Symbol startowy jaki jest każdy widzi;
- D to *delimiter* – nieterminal który (o ile znajduje się w którymś momencie wywodu) zawsze stoi na lewym końcu formy zdaniowej;
- E to taki symbol startowy, tylko że nie wstawia delimitera; jego jedynym zadaniem jest spamowanie nieterminali A, B, C w ramach kolejnych przejść wywodu;
- A, B, C to nieterminale które w określonych warunkach mogą zmienić się w terminale a, b, c.

Idea dowodu jest taka: „spamujemy" na początek symbolami A, B, C, a dzięki przejściom które mogą je permutować będzie można je „posortować" tak, by wygenerowały słowo postaci $a^n b^n c^n$.

Dowodzić, że gramatyka G jest taka, że generuje język $L = \{a^n b^n c^n : n \in \mathbb{N}\}$ będziemy dowodzić poprzez pokazanie inkluzji w obie strony.

Lemat 7.1.1. $L \subseteq L(G)$

Dowód. Pokażemy, że jeśli słowo w jest takie, że $w \in L$, to $w \in L(G)$.

Wiemy, że $w = a^n b^n c^n$ dla jakiegoś $n \in \mathbb{N}$. Pokażemy wywód w G, który generuje w w sposób następujący:

$$S \to_G DABCE \to_G DABCABCE \to_G^* D\overbrace{ABC \dots ABC}^{n \text{ razy}} E$$

$$D\overbrace{ABC \dots ABC}^{n \text{ razy}} E \to_G D\overbrace{ABC \dots ABC}^{n \text{ razy}}$$

„Wypychamy" teraz wszystkie nieterminale C na sam koniec korzystając z produkcji $CA \to AC$ i $CB \to BC$.

Następnie „wypychamy" wszystkie nieterminale B by znajdywały się po terminalach postaci A korzystając z produkcji $BA \to AB$.

Na wypadek gdyby ktoś miał tutaj wątpliwości formalne: każde takie przejście usuwa inwersję, a jeśli gdzieś jest inwersja to znaczy że gdzieś mamy nieterminale obok siebie które mogą być „swapnięte" tym przejściem. Zrobiliśmy tak zwane sortowanie bąbelkowe. Wracając: mamy teraz coś postaci:

$$D\overbrace{A\dots A}^{n\text{ razy}}\overbrace{B\dots B}^{n\text{ razy}}\overbrace{C\dots C}^{n\text{ razy}}$$

No i teraz elegancko korzystamy z produkcji $DA \to Da$ oraz $Da \to a$:

$$D\overbrace{A\dots A}^{n\text{ razy}}\overbrace{B\dots B}^{n\text{ razy}}\overbrace{C\dots C}^{n\text{ razy}} \to_G Da\overbrace{A\dots A}^{n-1\text{ razy}}\overbrace{B\dots B}^{n\text{ razy}}\overbrace{C\dots C}^{n\text{ razy}} \to_G a\overbrace{A\dots A}^{n-1\text{ razy}}\overbrace{B\dots B}^{n\text{ razy}}\overbrace{C\dots C}^{n\text{ razy}}$$

Korzystając z produkcji $aA \to aa$ dostaniemy:

$$\overbrace{a\dots a}^{n\text{ razy}}\overbrace{B\dots B}^{n\text{ razy}}\overbrace{C\dots C}^{n\text{ razy}}$$

No i teraz lecimy z tematem korzystając z produkcji $aB \to ab$, $bB \to bb$, $bC \to bc$, $cC \to cc$:

$$\overbrace{a\dots a}^{n\text{ razy}}\overbrace{B\dots B}^{n\text{ razy}}\overbrace{C\dots C}^{n\text{ razy}} \to_G^* \overbrace{a\dots a}^{n\text{ razy}}\overbrace{b\dots b}^{n\text{ razy}}\overbrace{c\dots c}^{n\text{ razy}} = w$$

Ale super. □

Lemat 7.1.2. $L(G) \subseteq L$

Dowód. Pokażemy, że jeśli jakieś słowo w jest takie, że $w \in L(G)$ to $w \in L$.

Na początek całkiem banalna obserwacja: wszystkie słowa generowane przez G są takie, że mają tyle samo wystąpień terminali a, b i c, jako że każda produkcja która dodaje terminal a usuwa jeden nieterminal A (analogicznie dla B i C; terminale D, E i S nie „przechodzą" w żadną literę).

Tym samym jedyne co mogłoby się „popsuć" to to gdzie występują literki w tych słowach.

Załóżmy więc nie wprost, że istnieje wywód w G w wyniku którego otrzymamy literę a gdzieś na pozycji $\geq n$ (zakładając indeksację od 0). Słowo które zostało otrzymane tym wywodem będziemy oznaczać jako w; $|w| = 3n$.

Na tę literę musiał przejść jakiś nieterminal, a jest dokładnie 1 nieterminal który mógłby na nią „przejść" i jest to nieterminal A. Czyli mielibyśmy coś od takiego:

$$\overbrace{a, b, \text{ lub } c}^{k\text{ razy, }k\geq n}\ \overbrace{a}^{\text{felerne } a}\ \overbrace{a, b, \text{ lub } c}^{l\text{ razy, }l\geq n}$$

gdzie $l + k = 3n - 1$. Prześledźmy w jaki sposób musiało dojść do czegoś takiego. Wiemy, że jedyne dwie produkcje która dadzą nam terminal a to $aA \to aa$ i $DA \to Da$. Jako, że D jest zawsze skrajnie po lewej stronie wywodu, jedyna produkcja która mogła nam tutaj dać a musiała być postaci $aA \to aa$.

A zatem, w którymś momencie przeprowadzania wywodu doszło do czegoś takiego:

$$\alpha aA\beta \to_G \alpha a\ \overbrace{a}^{\text{felerne } a}\ \beta$$

Ten terminal a który „spowodował" konwersję też skądś musiał się „wziąć" więc podobnie możemy stracebackować w jaki sposób on powstał (i znowu otrzymując, że musiał mieć terminal a po swojej lewej stronie). Innymi słowy: felerne a musi mieć po swojej lewej w finalnym słowie jedynie inne a. To daje nam sprzeczność, bo wiemy że liter a jest dokładnie n, a ten terminal znajduje się na miejscu $\geq n$.

Czyli wiemy już, że słowa generowane przez G mają postać:

$$\overbrace{a \dots a}^{n \text{ razy}} \beta$$

gdzie $\beta \in \{b, c\}^*$. Analogiczny dowód przeprowadzamy dla literek b i mamy pokazane, że G generuje jedynie takie słowa, które nas interesują. Ufff. □

Inkluzja w obie strony pokazana, super.

Ponadto język ten nie jest bezkontekstowy – pokażemy to jako przykład zastosowania lematu o pompowaniu dla języków bezkontekstowych (sekcja 7.2.2.1). □

7.1.3.4 Przykład języka rekurencyjnego

Uwaga. Podobnie jak już było zaznaczane wcześniej – przynajmniej anglojęzyczna Wikipedia nie notuje języków rekurencyjnych w hierarchi Chomsky'ego. Dla porządku podamy jednak przykład języka rekurencyjnego, który nie jest kontekstowy.

Żeby zrozumieć co tu się będzie działo, należy zapoznać się z sekcją 7.8.2 gdzie wprowadzone zostanie pojęcie uniwersalnej maszyny Turinga.

Istnieją prostsze języki rekurencyjne, ale wiele z nich może zostać również „rozwalone" przez LBA (czyli tak naprawdę są kontekstowe). Dlatego przykład języka rekurencyjnego który pokażemy będzie nieco dziwny, bo robi argument przekątniowy dla LBA.

Twierdzenie 7.1.9. Język $L = \{x_i : x_i \notin L(\mathrm{LBA}_i)\}$ jest taki, że $L \in \mathrm{R} \setminus \mathrm{CSL}_\Sigma$.

Dowód. Na początek proszę zauważyć, że możemy coś takiego zrobić: istnieje sposób (skończonego) kodowania Maszyn Turinga, a LBA jest Maszyną Turinga tylko że z ograniczoną taśmą. Tym samym Maszyny Turinga (w szczególności LBA) można enumerować (bo jest ich przeliczalnie wiele), a instytucja i-tego LBA w jakimś porządku ma sens. Nie trzeba tłumaczyć, że instytucja i-tego słowa w danym alfabecie również ma sens.

Jak już wiemy, że definicja naszego języka jest legalna, możemy przejść do właściwego dowodu. Najpierw dowiedziemy, że L nie jest kontekstowy, a potem pokażemy że jest rekurencyjny.

Lemat 7.1.3. $L \notin \mathrm{CSL}_\Sigma$

Dowód. Załóżmy, że język L jest kontekstowy. W takim razie istnieje LBA, które go rozpoznaje, a zatem musi być k-te w kolejności ze wszystkich LBA w jakimś porządku (czyli jest to LBA_k dla jakiegoś $k \in \mathbb{N}$). Rozważmy w takim razie k-te słowo w naszym porządku na słowach:

$$w_k \in L(\mathrm{LBA}_k) \iff w_k \in L$$

Jako że $L = L(\mathrm{LBA}_k)$ z założenia nie wprost.

No ale jednocześnie z defnicji L:

$$w_k \in L \iff w_k \notin L(\mathrm{LBA}_k)$$

skąd:

$$w_k \notin L(\mathrm{LBA}_k) \iff w_k \in L \iff w_k \in L(\mathrm{LBA}_k)$$

co daje nam sprzeczność. □

□

Lemat 7.1.4. $L \in \mathrm{R}$

Dowód. Po pierwsze wypada zauważyć, że problem stopu dla LBA jest rozstrzygalny, bo LBA może mieć jedynie skończenie wiele konfiguracji (a w takim razie istnieje MT która „spamiętuje" która konfiguracja wystąpiła, a która nie). W takim razie istnieje Maszyna Turinga która gdy dostanie dane słowo może policzyć które jest ono w alfabecie, a następnie wygenerować k-te LBA i sprawdzić, czy się zawiesi na tym słowie (i jaki będzie rezultat jego działania). Tym samym język ten jest ewidentnie rekurencyjny. □

7.1.3.5 Przykład języka rekurencyjnie przeliczalnego

Twierdzenie 7.1.10. Język stopu (L_{HP}) jest rekurencyjnie przeliczalny i nie jest rekurencyjny.

Dowód. Patrz: sekcja 7.8.2. □

7.2 Omów lemat o pompowaniu dla języków bezkontekstowych. Podaj przykład języka bezkontekstowego. Podaj przykład języka, który nie jest bezkontekstowy

Przykład języka bezkontekstowego znajduje się w sekcji 7.1.3.2.

7.2.1 Lemat o pompowaniu dla języków regularnych

Co prawda w treści pytania tego nie ma, ale twierdzenie to i tak warto znać w kontekście lematu o pompowaniu dla języków bezkontekstowych.

Lemat 7.2.1 (O pompowaniu dla języków regularnych)**.** Jeśli L jest językiem regularnym to:

$$\exists_{n>0} : \forall_{w:|w|\geq n} \exists_{xyz=w} : |xy| \leq n \wedge |y| \geq 1 \wedge \forall_{i\in\mathbb{N}} : xy^i z \in L$$

Dowód. Skoro L jest językiem regularnym to istnieje DFA A, który rozpoznaje L.

Niech $n = |Q|$ i weźmy dowolne słowo w dla którego $m = |w| \geq n$.

Skoro słowo jest akceptowane to istnieje ścieżka $s = q_0, \ldots q_m \in F$

Mamy więc $m + 1 > n$ stanów, czyli jakiś stan musi się powtarzać. Z takich stanów wybieramy q_i, którego pierwsze powtórzenie jest najwcześniej. Niech q_j będzie drugim wystąpieniem stanu q_i

Mamy zatem $x = a_0 \ldots a_{i-1}$, $y = a_i \ldots a_{j-1}$, $z = a_j \ldots a_{m-1}$

Oczywiście $|xy| \leq n$ bo inaczej q_i nie powtarzałby się najwcześniej.

Czynimy teraz fajną obserwację, mianowicie skoro $q_i = q_j$ to słowo y może wystąpić dowolną (również zerową) liczbę razy w akceptowanym słowie.

Innymi słowy, dla dowolnego $i \in \mathbb{N}$ $xy^iz \in L$ □

Proszę pamiętać, że jest to warunek konieczny, ale **nie jest on wystarczający** by język był regularny. Istnieją języki które zdecydowanie nie są regularne, a spełniają warunki przedstawione w lemacie o pompowaniu.

7.2.2 Lemat o pompowaniu dla języków bezkontekstowych

Twierdzenie 7.2.1 (O pompowaniu dla języków bezkontekstowych)**.** Jeżeli język L nad Σ^* jest bezkontekstowy, to:

$\exists_{n_0 \in \mathbb{N}}$
$\forall_{w \in L} : |w| \geq n_0$
$\exists_{a,b,c,d,e \in \Sigma^*}\ w = abcde \wedge |bcd| \leq n_0 \wedge |bd| \geq 1$
$\forall_{i \in \mathbb{N}}\ a \cdot b^i \cdot c \cdot d^i \cdot e \in L$

Dowód. Generalnie chcemy, żeby w derywacji naszego słowa w, które pompujemy dwa razy na jednej ścieżce pojawił się ten sam nieterminal A. Mamy wtedy $A \rightarrow_G^* c$ oraz $A \rightarrow_G^* bcd$ Wtedy możemy wstawić „górną" produkcję (tę co doprowadza do bcd) w miejsce „dolnej" (tej co doprowadza do c) i w ten sposób pompujemy *bcd* do *bbcdd*.

Niech $n = |N|$ – liczba dostępnych nieterminali oraz niech m – długość najdłuższej produkcji.

Wtedy jeśli nasze słowo jest długości co najmniej $n_0 = m^{n+1} + 1$ to drzewo wywodu musi mieć wysokość co najmniej $n + 1$ a zatem jakiś nieterminal musi się powtórzyć na jakiejś ścieżce.

Aby zapewnić warunek $|bcd| \leq n_0$ z tezy bierzemy ten nieterminal, którego „niższe" wystąpienie jest najwyżej ze wszystkich – tj. znajduje się na głębokości co najwyżej $n + 1$.

Jeśli mamy pecha i dla takiego wyboru $|bd| = 0$ to znaczy że przeszliśmy ścieżką, która nic nie produkuje. Szukamy wtedy innego spełniającego warunek nieterminala – musi taki istnieć bo w końcu słowo które produkujemy jest długie.

□

7.2.2.1 Zastosowanie lematu o pompowaniu dla języków bezkontekstowych

Twierdzenie 7.2.2. Język $L = \{a^n b^n c^n : n \in \mathbb{N}\}$ nie jest bezkontekstowy.

Dowód. Stosujemy lemat o pompowaniu dla języków bezkontekstowych. Pokażemy, że dla L nie zachodzi lemat o pompowaniu dla języków bezkontekstowych (a więc nie może być kontekstowy). W tym celu musimy pokazać, że:

$\forall_{n_0 \in \mathbb{N}}$
$\exists_{w \in L} : |w| \geq n_0$
$\forall_{a,b,c,d,e \in \Sigma^*}\ w = abcde \wedge |bcd| \leq n_0 \wedge |bd| \geq 1$
$\exists_{i \in \mathbb{N}}\ a \cdot b^i \cdot c \cdot d^i \cdot e \notin L$

Dla określonego n_0 robimy słowo $w = a^{n_0} b^{n_0} c^{n_0}$. Ma ono długość $3n_0$ (obviously). Zauważam, że dla dowolnego podziału słowa w (które spełnia warunki lematu) będzie tak, że podciąg $|bcd|$ (z racji faktu że ma długość maksymalnie n_0 może zawierać dokładnie 1 lub dokładnie 2 litery

ze zbioru liter $\{a, b, c\}$. Tym samym kiedy zdepompujemy (ustawimy $i = 0$) zredukujemy liczbę jednej lub dwóch liter występujących w słowie, ale nie wszystkich trzech. Tym samym będzie na pewno istnieć jedna litera która występuje n_0 razy i (z racji zdepompowania) jakaś litera która występuje razy mniej. A wtedy takie słowo ewidetnie do tego języka nie należy (bo raczej prosto zauważyć, że warunkiem koniecznym ale niewystarczającym przynależenia do L jest to, by mieć tyle samo liter a, b, c).

□

7.3 W jaki sposób języki regularne są charakteryzowane przez automaty skończone? Nakreśl ideę dowodu

W tej sekcji pokażemy, że dla dowolnego wyrażenia regularnego istnieje równoważny[2] ε - NFA, dla dowolnego ε - NFA istnieje równoważne DFA, a dla każdego DFA istnieje równoważny regex – prowadząc nas do wniosku, że jeśli mamy DFA które generuje jakiś język, to ten jest regularny.

7.3.1 DFA $\to$ ε-NFA

Na początek pokażemy konwersję DFA na NFA, a potem pokażemy dowód konwersji DFA na ε-NFA. Ideowo są one takie same, ale dowód z konwersją na NFA jest o epsilon prostszy.

Twierdzenie 7.3.1. Niech $L \subseteq \Sigma^*$. Następujące warunki są równoważne:

1. Istnieje DFA A_D, dla którego $L(A_D) = L$
2. Istnieje NFA A_N, dla którego $L(A_N) = L$

Dowód. „ $\Longrightarrow$ ”

Każdy DFA jest NFA, bo każda funkcja jest relacją.

„ $\Longleftarrow$ ”

Niech A_N będzie NFA takim, że:

$$A_N = (Q, \Sigma, \delta, S, F)$$

Korzystamy z faktu, że $\tilde{\delta}$ jest elegancką funkcją i konstruujemy DFA A_D takie, że:

$$A_D = (\mathcal{P}(Q), \Sigma, \tilde{\delta}, S, \mathcal{F})$$

gdzie $\mathcal{F} = \{\beta \in \mathcal{P}(Q) \mid \beta \cap F \neq \varnothing\}$

Intuicyjnie, tworzymy DFA, które w swoich stanach trzyma informację, w jakich stanach NFA mogłoby się znaleźć po przejściu tego samego prefiksu słowa. W związku z powyższym definicja $\mathcal{F}$ jest całkiem intuicyjna, bo chcemy zaakceptować wtedy i tylko wtedy, gdy któreś obliczenie NFA znalazło się w stanie akceptującym.

Dla pełności potrzebujemy pokazać, że $\hat{\delta}_{A_N}(S, u) \cap F \neq \varnothing \iff \hat{\delta}_{A_D}(S, u) \in \mathcal{F}$, ale to mamy tak naprawdę z definicji, bo $\hat{\delta}_{A_D} = \hat{\delta}_{A_N}$

□

[2]Pod względem generowanego/rozpoznawanego języka

Twierdzenie 7.3.2. Niech $L \subseteq \Sigma^*$. Następujące warunki są równoważne:

1. Istnieje DFA A_D, dla którego $L(A_D) = L$
2. Istnieje ε-NFA A_N, którego $L(A_N) = L$

Dowód. „$\Longrightarrow$”

Każdy DFA jest ε-NFA, bo każda funkcja jest relacją.

„$\Longleftarrow$”

Niech A_N będzie ε-NFA takim, że:

$$A_N = (Q, \Sigma, \delta, S, F)$$

Ponownie jak w poprzednim przypadku, chcemy w stanach DFA „trzymać” informacje o tym, w jakich stanach może znaleźć się $\varepsilon - NFA$. Jedyne co się zmienia to to, że mamy do czynienia z epsilonami. Jak o tym pomyślimy, to w sumie jednak nie jest to wielki problem, bo wystarczy chodzić po epsilon-domknięciach z użyciem funkcji Δ.

Konstruujemy więc A_D, które jest DFA takim, że:

$$A_D = (\mathcal{P}^{A_N,\varepsilon}(Q), \Sigma, \Delta, S, \mathcal{F})$$

gdzie $\mathcal{F} = \left\{\beta \in \mathcal{P}^{A_N,\varepsilon}(Q) \mid \beta \cap F \neq \varnothing\right\}$

□

7.3.2 DFA $\rightarrow$ Wyrażenie regularne

Pokażemy teraz, że dla dowolnego DFA A możemy skonstruować wyrażenie regularne α takie, że $L(A) = L(\alpha)$.

Czynimy obserwację, że jeśli $w \in L(A)$ to istnieje ścieżka $s \rightarrow q_f \in F$ idąca stanami

$$q_1, q_2, \ldots, q_{k+1}$$

taka że $s = q_1$ i $q_{k+1} \in F$.

Definiujemy $\alpha^k_{i,j}$ jako takie wyrażenie regularne, które reprezentuje wszystkie ścieżki prowadzące od q_i do q_j, w której stany pośrednie należą do $\{q_1, \ldots, q_k\}$ (stany numerujemy *arbitralnie* od 1, bo tak będzie wygodniej). Widzimy, że dla każdych i, j:

$$\alpha^0_{i,j} = \{a \in \Sigma : \delta(q_i, a) = q_j\} \cup \beta_{i,j}$$

gdzie

$$\beta_{i,j} = \begin{cases} \varnothing & \text{gdy } i \neq j \\ \{\varepsilon\} & \end{cases}$$

Zauważmy teraz, że dla dowolnych $k, i, j \leq |Q|, k \geq 1$ jest tak, że:

$$\alpha^k_{ij} = \alpha^{k-1}_{ij} + \alpha^{k-1}_{ik}(\alpha^{k-1}_{kk})^*\alpha^{k-1}_{kj}$$

To może wydawać się być overwhelming, więc to rozbijmy:

1. Pierwszy składnik sumy α_{ij}^{k-1} mówi nam o sytuacji, kiedy przechodzimy ze stanu q_i do stanu q_j nie przechodząc przez q_k ani stan o „wyższym" numerze w porządku.
2. Drugi składnik sumy mówi o sytuacji, kiedy przechodzimy przez q_k (bo teraz możemy). W takiej sytuacji nasza wybitna podróż dzieli się na 3 segmenty, których wyrażenia regularne są relatywnie proste i które wypada skonkatenować, by mieć opis całej podróży:
   (a) Przejście z q_i do q_k (po raz pierwszy w czasie podróży) – opisywane wyrażeniem α_{ik}^{k-1}
   (b) Chodzenie dowolną liczbę razy (być może 0) z q_k do q_k , po stanach po których nam wolno chodzić – można to łatwo opisać wyrażeniem $(\alpha_{kk}^{k-1})^*$ (gwiazdka oddaje to, że możemy chodzić wiele razy).
   (c) Przejście z q_k do q_j (po raz ostatni czas w podróży, tzn. po tym wyjściu z q_k już tam nie wrócimy) – opisywane wyrażeniem α_{kj}^{k-1}.

Okazuje się w takim razie, że dostaliśmy coś bardzo fajnego: mianowicie, pokazaliśmy że jak mamy DFA możemy sobie konstruować takie wyrażenia regularne dla kolejnych k. Jest to absolutnie wspaniałe na mocy naszej wcześniejszej obserwacji, tzn. tego, że jeśli $w \in L(A)$ to istnieje taka ścieżka w naszym DFA. W takim razie regex który opisuje tę ścieżkę ma postać:

$$\sum_{q_j \in F} \alpha_{ij}^{k}$$

gdzie k to liczba stanów, a q_i to stan początkowy. Jest to poprawnie zdefiniowane wyrażenie regularne, bo jego składowe są poprawnie zdefiniowane. Formalnie dowód tego, że wszystkie wyrażenia regularne tej postaci są poprawnie zdefiniowane można przeprowadzić pewnie jakąś indukcją; pozostawiamy to jako ćwiczenie dla czytelnika.

7.3.3 Wyrażenie regularne → ε-NFA

Ostatnia rzecz, którą chcemy uzyskać to sposób konwersji wyrażenia regularnego do ε-NFA. To dałoby nam już wspaniały rezultat, bo mielibyśmy, że wyrażenia regularne i wymienione wyżej automaty są wzajemnie równoważne jeśli chodzi o ich „moc". W tym celu możemy wykonać indukcję po długości wyrażenia regularnego, intuicyjnie konstruując nowy automat na podstawie „wcześniejszych".

Na początek wypada powiedzieć co robimy dla „trywialnych" wyrażeń regularnych, by indukcja się spięła. Jeśli wyrażene jest postaci a, dla $a \in \Sigma$, tworzymy trywialnie automat który akceptuje jedynie słowo takiej postaci. Dla pustego wyrażenia tworzymy automat który nie akceptuje niczego, dla 1 tworzymy (równie trywialnie) automat który zaakceptuje jedynie ε.

Mając z głowy nudne przypadki bazowe, możemy przejść do ciekawszej rzeczy dowodu, czyli konwersji poszczególnych wyrażeń regularnych na automaty. Będziemy się kejsować po tym, na co jesteśmy w stanie „rozbić" regex[3]:

- Jeśli wyrażenie regularne jest postaci $\alpha_1 + \alpha_2$, to z założenia indukcyjnego mamy, że istnieją DFA A_1, A_2 takie, że $L(A_1) = L(\alpha_1)$ i $L(A_2) = L(\alpha_2)$. Możemy stworzyć automat A_3, który składa się z tych samych stanów i przejść co A_1 i A_2, ale ma jeszcze dodatkowy stan q; q jest stanem startowym nowego automatu i ma epsilon przejścia do stanów startowych A_1 i A_2. Zauważamy, że jeśli słowo należy do języka akceptowanego przez

[3]czyli robimy indukcję strukturalną czy coś tam

któryś z tych automatów (nazwijmy go A), to w automacie A_3 istnieje obliczenie, które ze stanu startowego przechodzi epsilon-przejściem do stanu startowego A, po czym to słowo zostanie zaakceptowane. Jeśli żaden z automatów nie akceptował tego słowa, to trywialnie widać że nowy automat również go nie zaakceptuje.

- Jeśli wyrażenie regularne jest postaci $\alpha_1 \cdot \alpha_2$, to, podobnie jak powyżej, musimy wymyślić jakiś sposób na zrobienie nowego automatu mając automaty A_1 i A_2 tak, by ten akceptował konkatenację tych języków. Ponownie więc tworzymy A_3, który ma wszystkie stany i przejścia ze wspomnianych automatów wraz z nowym stanem startowym q. Dodaję ε-przejście z q do stanu startowego z A_1 oraz ε-przejścia ze stanów akceptujących A_1 do stanu startowego A_2. Stanami akceptującymi nowego automatu są **tylko** stany akceptujące z A_2. Widać że to działa ładnie – jeśli jakieś słowo w jest postaci xy, gdzie $x \in L(A_1)$ i $y \in L(A_2)$, to istnieje obliczenie które „przejdzie" x do stanu akceptującego, następnie „przeskoczy" po epsilonie do A_2, który z kolei zaakceptuje y. Jeśli zaś słowo nie należy do konkatenacji takich języków, to znaczy że taki podział nie istnieje, a więc albo w ogóle nie „przejdziemy" do A_2, albo po przejściu do A_2 nie zostaniemy zaakceptowani.
- Jeśli wyrażenie regularne jest postaci $(\alpha)*$, to mamy jakiś automat A, dla którego $L(A) = L(\alpha)$. Chcemy teraz zrobić jakąś „owijkę" na niego tak, by można było zaakceptować dowolne słowo będące „zlepkiem" słów z tego języka (bo to gwiazdka kleenego). Konstrukcja jest całkiem oczywista: robimy automat B, który, standardowo, ma wszystkie stany i przejścia z automatu A. Dodajemy oddzielny stan startowy q; dodajemy z niego ε-przejście do stanu startowego A. Dodajemy jeszcze ε-przejścia ze stanów akceptujących A z powrotem do stanu q. Stanem akceptującym również jest q. Zauważmy, że jeśli jakieś słowo należy do języka generowanego przez $(\alpha)^*$, to istnieje jego podział na słowa które należą do $L(\alpha)$, a więc istnieje obliczenie które będzie sobie akceptować po kolei te „podsłowa", a na samym kończu się zakończy. Jednocześnie, cokolwiek co jest zaakceptowane przez B musi być takim „zlepkiem", bo by „wrócić" do q trzeba było przejść sekwencją przejść po literach stanowiących słowo matchujące wyrażenie regularne.

7.4 Omów zamkniętość klasy języków regularnych na operacje na językach

7.4.1 Suma

Twierdzenie 7.4.1. Języki regularne są zamknięte na sumowanie.

Dowód. Mając wyrażenia regularne α_1 i α_2 możemy zrobić wyrażenie $\beta = \alpha_1 + \alpha_2$. □

7.4.2 Przecięcie

Twierdzenie 7.4.2. Języki regularne są zamknięte na przecięcie.

Dowód. Mając DFA $A_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ i DFA $A_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$ możemy skonstruować DFA B, które będzie rozpoznawać język $L(A_1) \cap L(A_2)$, takie że:

$$B = (Q', \Sigma, \delta', q', F')$$

gdzie:

- $Q' = Q_1 \times Q_2$
- $\delta'(q_1, q_2) = (\delta_1(q_1), \delta_2(q_2))$
- $s' = (s_1, s_2)$
- $F' = \{(q_1, q_2) : q_1 \in F_1 \wedge q_2 \in F_2\}$

Intuicyjnie: DFA B symuluje chodzenie po A_1 i A_2 akceptując wtedy i tylko wtedy, gdy w obu DFA dane słowo znalazłoby się w stanie akceptującym. Formalniej:

$$w \in L(A_1) \cap L(A_2) \iff w \in L(A_1) \wedge w \in L(A_2) \iff$$
$$\tilde{\delta}_1(s_1, w) \in F_1 \wedge \tilde{\delta}_2(s_2, w) \in F_2 \iff \delta'((s_1, s_2), w) \in F' \iff w \in L(B)$$

□

7.4.3 Dopełnienie

Twierdzenie 7.4.3. Wyrażenia regularne są zamknięte na dopełnienie.

Dowód. Mając DFA A możemy „odwrócić" wszystkie stany akceptujące, tzn. stworzyć DFA B takie, że $F_B = \{q \in Q_A : q \notin F_A\}$. Wtedy:

$$w \notin L(A) \iff \tilde{\delta}(s_a, w) \notin F_A \iff \delta(s_a, w) \in F_B \iff w \in L(B)$$

□

7.4.4 Konkatenacja

Twierdzenie 7.4.4. Wyrażenia regularne są zamknięte na konkatenację.

Dowód. Mając wyrażenia regularne α_1 i α_2 możemy skonstruować wyrażenie regularne $\alpha_1\alpha_2$. □

7.4.5 Gwiazdka Kleenego

Twierdzenie 7.4.5. Wyrażenia regularne są zamknięte na gwiazdkę Kleenego.

Dowód. Mając wyrażenie regularne α możemy stworzyć wyrażenie α^*. □

7.5 Omów zamkniętość klasy języków bezkontekstowych na operacje na językach

7.5.1 Suma

Twierdzenie 7.5.1. Języki bezkontekstowe są zamknięte na sumę.

Dowód. Niech $L_1, L_2 \in CFL$. Mamy zatem gramatyki bezkontekstowe G_1, G_2

Dla języka $L = L_1 \cup L_2$ konstruujemy gramatykę $G = G_1 \cup G_2$ (zakładamy, że zbiory nieterminali tych gramatyk są rozłączne). Dodajemy do gramatyki produkcję:

$$S \to S_1 \mid S_2$$

□

7.5.2 Konkatenacja

Twierdzenie 7.5.2. Języki bezkontekstowe są zamknięte na konkatenację.

Dowód. Czynimy tak jak w przypadku sumy ale naszą produkcją którą dodajemy jest

$$S \to S_1 S_2$$

□

7.5.3 Gwiazdka Kleenego

Twierdzenie 7.5.3. Języki bezkontekstowe są zamknięte na gwiazdkę Kleenego.

Dowód. Jak wyżej, ale naszą produkcją jest

$$S \to S_1 S \mid \varepsilon$$

□

7.5.4 Przecięcie

Twierdzenie 7.5.4. Języki bezkontekstowe **nie** są zamknięte na przecięcie.

Dowód. Kontrprzykładem jest $L_1 = \left\{a^n b^n c^k : n, k \in \mathbb{N}\right\}$ $L_2 = \left\{a^k b^n c^n : n, k \in \mathbb{N}\right\}$. Oczywiście $L_1 \cap L_2 = \left\{a^n b^n c^n : n \in \mathbb{N}\right\}$ co wiemy że nie jest bezkontekstowe. □

7.5.5 Dopełnienie

Twierdzenie 7.5.5. Języki bezkontekstowe **nie** są zamknięte na dopełnienie.

Dowód. Mamy z praw de Morgana

$$(L_1 \cap L_2) = \overline{(\overline{L_1} \cup \overline{L_2})}$$

Jeśli zatem mielibyśmy zamknięcie na dopełnienie to byśmy mieli zamknięcie na przecięcie co już pokazaliśmy że nie jest prawdziwe. □

7.6 Omów twierdzenie Myhilla–Nerode'a, podając ideę dowodu i związek z minimalizacją automatów skończonych

7.6.1 A-równoważność

Definicja 7.6.1. Stany q_1, q_2 zadanego DFA są **A-równoważne** jeśli

$$\forall_{w\in\Sigma^*}\hat{\delta}(q_1,w)\in F \iff \hat{\delta}(q_2,w)\in F$$

i **A-rozróżnialne** jeśli

$$\exists_{w\in\Sigma^*}\hat{\delta}(q_1,w)\in F \iff \hat{\delta}(q_2,w)\notin F$$

Lemat 7.6.1. A-równoważnosć jest relacją równoważności.

Dowód. 1. Zwrotność:

$$\forall_{w\in\Sigma^*}\hat{\delta}(q_1,w)\in F \iff \hat{\delta}(q_1,w)\in F$$

raczej nie wymaga komentarza.

2. Symetryczność:

$$\forall_{w\in\Sigma^*}\hat{\delta}(q_1,w)\in F \iff \hat{\delta}(q_2,w)\in F$$
$$\equiv$$
$$\forall_{w\in\Sigma^*}\hat{\delta}(q_2,w)\in F \iff \hat{\delta}(q_1,w)\in F$$

co wynika z przemienności równoważności.

3. Przechodniość; zakładając że mamy stany q_1, q_2 oraz q_3 takie, że:

$$\forall_{w\in\Sigma^*}\hat{\delta}(q_1,w)\in F \iff \hat{\delta}(q_2,w)\in F$$

oraz

$$\forall_{w\in\Sigma^*}\hat{\delta}(q_2,w)\in F \iff \hat{\delta}(q_3,w)\in F$$

trywialnie otrzymujemy, że:

$$\forall_{w\in\Sigma^*}\hat{\delta}(q_1,w)\in F \iff \hat{\delta}(q_2,w)\in F \iff \hat{\delta}(q_3,w)\in F$$

czyli q_1 i q_3 również są A-równoważne.

□

7.6.2 Algorytm D

Zdefiniujemy sobie tak zwany *Algorytm D*. Służyć on będzie do minimalizacji deterministycznych automatów skończonych.

- Wejście: DFA A
- Wyjście: (trójkątna) macierz Boolowska M, w której $M_{i,j} = 1 \iff a_i, a_j$ są rozróżnialne

Na początku wypełniamy M zerami. Jeśli $a_i \in F, a_j \notin F$ to $M_{i,j} = 1$. Dopóki macierz M się zmienia, to dla każdych dwóch stanów q_1, q_2 oraz litery a jeśli $\hat{\delta}(q_1, a) = q_k, \hat{\delta}(q_2, a) = q_l$ oraz $M_{k,l} = 1$ to ustawiamy $M_{i,j} := 1$.

Jeśli po zakończeniu algorytmu jakieś 2 stany q_i, q_j są takie, że $M_{i,j} = 0$ to wtedy wiemy, że q_i oraz q_j nie są rozróżnialne, a więc są A-równoważne. Zauważmy, że w ten sposób znajdujemy wszystkie stany które są sobie równoważne (działa to trochę jak taki Floyd-Warshall). Teraz co robimy, to kolapsujemy wszystkie równoważne stany w jeden wierzchołek. Okazuje się, że tak powstałe DFA jest minimalne pod względem liczby stanów (w porównaniu do innych DFA rozpoznających ten sam język).

Twierdzenie 7.6.1. Automat otrzymany w wyniku pracy algorytmu D jest najmniejszy spośród DFA rozpoznających dany język.

Dowód. Dowód nie wprost: załóżmy, że w wyniku minimalizacji jakiegoś automatu otrzymaliśmy jakiś automat A, a istnieje DFA B takie, że $L(A) = L(B)$ i B ma mniej stanów. Zdefiniujmy sobie DFA C[4] takie, że jego stanem startowym jest s_B, a $Q_C = Q_A \cup Q_B$, funkcje przejść również „sumujemy". Zauważmy, że dla automatu C:

$$\forall_{q_A \in Q_A} \exists_{q_B \in Q_B} \ q_a \ i \ q_b \ są \ C\text{-}równoważne$$

Dowód tej obserwacji jest prosty: jako, że A i B akceptują te same języki, to na pewno s_A i s_B są C-równoważne. No ale w takim razie dla każdego $a \in \Sigma$ jest tak, że $\delta(s_A, a)$ i $\delta(s_B, a)$ są również C-równoważne (i tak dalej). Stosując tę konstrukcję dalej, otrzymamy że dla każego stanu z A istnieje równoważny stan w B.

Na podstawie tej informacji wykonujemy *potężną* obserwację: skoro, z założenia nie wprost, A miał więcej stanów w B; to w połączeniu z powyższą obserwacją oznacza, że istnieją stany $q_1, q_2 \in A$ takie, że oba z nich są równoważne z jakimś stanem $q_3 \in B$. Ponieważ jednak równoważnośc jest przechodnia, mamy że q_1 i q_2 są wzajemnie równoważne. To prowadzi do sprzeczności, bo wiemy, że algorytm D będzie na tyle miły, że skolapsuje wszystkie stany które są wzajemnie równoważne. To kończy dowód.

□

7.6.3 Twierdzenie Myhilla-Nerode'a

Definicja 7.6.2. Mówimy, że relacja jest L-kongruentna dla pewnego języka $L \in \mathcal{P}(\Sigma^*)$ wtedy i tylko wtedy, gdy:

1. $\forall_{v \in \Sigma^*} x \equiv y \implies xv \equiv yv$
2. $x \equiv y \implies (x \in L \iff y \in L)$

Twierdzenie 7.6.2 (Myhill–Nerode)**.** Język L jest regularny wtedy i tylko wtedy gdy istnieje relacja równoważności $\equiv \subseteq \Sigma^* \times \Sigma^*$ o skończenie wielu klasach równoważności, która jest L-kongruentna.

Dowód.

[4]Tworzymy je tylko i wyłącznie z przyczyn formalnych, bo A-równoważnośc jest zdefiniowana dla określonego automatu.

$\Longrightarrow$

Skoro L jest regularny to istnieje DFA A, na podstawie którego definiujemy

$$x \equiv y \iff \hat{\delta}(s,x) = \hat{\delta}(s,y)$$

gdzie s jest stanem startowym w automacie A.

Należy teraz dowieść, że:

1. Relacja ta jest L-kongruentna:
   (a) $x \equiv y \iff \hat{\delta}(s,x) = \hat{\delta}(s,y) = q$; jednocześnie (oczywiście) $\hat{\delta}(q,v) = \hat{\delta}(q,v) \implies xv \equiv yv$
   (b) $x \equiv y \iff \hat{\delta}(s,x) = \hat{\delta}(s,y) = q$, więc jeśli $q \in F$ to warunek spełniony, a jeśli $q \notin F$ to również.
2. Relacja ta ma skończenie wiele klas równoważności, bo funkcja $\hat{\delta}$ może „przemapować" dane słowo na maksymalnie $|Q|$ stanów, a wszystkie słowa które „przechodzą" na ten sam stan są ze sobą równoważne.

$\Longleftarrow$

Tworzymy automat, którego stanami są klasy równoważności słów wyznaczone przez relację $\equiv$. Możemy tak zrobić, bo wiemy że klas równoważności jest skończenie wiele. Jako stany finalne ustalamy te klasy równoważności, w których wszystkie elementy należą do języka (z racji że jest to L-kongruencja wszystkie elementy klasy mogą albo należeć, albo nie należeć do języka). Stanem startowym jest klasa, do której należy ε.

Teraz żeby to się spięło i w ogóle, należy jeszcze ustalić jak wyglądają funkcje przejścia. Robimy to w ten sposób, że dla każdej klasy próbujemy „dokleić" jedną literkę z alfabetu i dodajemy przejście do klasy, w której wylądują powstałe słowa. Warto tutaj zauważyć, że z racji że jest to L-kongruencja, jeśli dodam literkę $a \in \Sigma$ do dowolnych słów z mojej klasy, to wszystkie słowa powstałe jako rezultat muszą znaleźć się w jednej klasie (z racji warunku pierwszego w definicji L-kongruencji).

Procedura „generowania przejścia" jest wykonywana dla każdej klasy i każdej litery; zauważmy, że wówczas funkcja przejścia jest już poprawnie zdefiniowaną funkcją.

Teraz należy udowodnić, że powstały DFA akceptuje język, na którym zdefiniowana jest L-kongruencja.

Zauważamy, że $\hat{\delta}(s,w)$ jest klasą równoważności, do której należy słowo w. Dowód przebiega z pomocą indukcji po długości w: gdy $|w| = 0$, to $w = \varepsilon$ i z definicji q_s jako klasy zawierającej epsilon wszystko działa poprawnie.

W kroku indukcyjnym, jeśli mamy słowo $w' = wa$, to z założenia indukcyjnego wiemy, że $\hat{\delta}(w)$ jest klasą równoważności, do której należy w. Wówczas przejście po literze a z tamtej klasy prowadzi nas do klasy zawierającej słowo wa, z tego jak zdefiniowaliśmy funkcję δ.

Cóż możemy zrobić z tą fascynującą obserwacją? Zauważyć, że teraz pokazanie że $L(A) = L$ mamy teraz za darmo, bo jeśli $w \in L$ to klasa równoważności która zawiera w jest stanem akceptującym; w przeciwnym razie nie jest. To dowodzi poprawności konstrukcji.

□

7.7 Determinizm i niedeterminizm dla maszyn Turinga: omów oba modele i związek między nimi

7.7.1 Niedeterministyczna Maszyna Turinga

Definicja 7.7.1. Niedeterministyczną Maszynę Turinga definiujemy tak samo jak deterministyczną Maszynę Turinga[5], przy czym funkcja δ zostaje relacją; ciągnie to za sobą dosyć oczywiste konsekwencje w definiowaniu relacji $\vdash$.

Co ciekawe, definicja języka akceptowanego się nie zmienia; nadal zwyczajnie chcemy mieć stan akceptujący w domknięciu przechodnim.

7.7.2 Równoważność Deterministycznej Maszyny Turinga i Niedeterministycznej Maszyny Turinga

Twierdzenie 7.7.1. Następujące stwierdzenia są równoważne dla dowolnego języka $L \subset \Sigma^*$:

1. Istnieje deterministyczna Maszyna Turinga DM taka, że $L(DM) = L$.
2. Istnieje niedeterministyczna Maszyna Turinga NM taka, że $L(NM) = L$.

Dowód. Dowód faktu, że mając deterministyczną Maszynę Turinga możemy utworzyć niedeterministyczną Maszynę Turinga jest trywialny (każda funkcja jest relacją).

Skupmy się zatem na dowodzie w drugą stronę; mamy daną niedeterministyczną Maszynę Turinga N i chcemy stworzyć równoważną deterministyczną Maszynę Turinga M. Aby to uczynić, będziemy (intuicyjnie) chodzić BFS-em po nieskończonym grafie konfiguracji maszyny N.

Formalizacja tego jest bardzo ciężka, mimo faktu że idea jest prosta (zatem podamy ideę). W alfabecie chcemy mieć jakiś separator, który jedyne do czego będzie używany, to do rozpatrywania kolejnych konfiguracji „w kolejce".

Na początku „w kolejce" konfiguracji do analizy mamy jedynie konfigurację startową. Mając głowicę w jakimś położeniu, znając jej stan i to co jest „na prawo" od niej, dopisujemy do kolejki symbol separacji konfiguracji, a następnie całą naszą konfigurację po wykonanym którymś przejściu z maszyny N. Procedurę tę wykonujemy dla każdego możliwego przejścia z N.

Gdy „przerobimy" wszystkie możliwe przejścia z N (dla każdej sytuacji jest ich skończenie wiele, więc zakodowanie tego jak M ma się zachowywać by wygenerować wszystkie te konfiguracji jest osiągalne), przechodzimy do następnej konfiguracji z kolejki. No BFS, co tu jeszcze można powiedzieć?

Za każdym razem, gdy analizujemy nową konfigurację, sprawdzamy czy zapisany w niej stan jest akceptujący; jeśli tak, to wtedy akceptujemy słowo. Jeśli nie ma zdefiniowanych przejść dla określonej sytuacji, to nie robimy nic i przechodzimy do kolejnej konfiguracji w kolejce.

Again, formalizowanie tego wymagałoby wprowadzenia dodatkowych stanów które trzymałyby informacje o tym, co „widzi" maszyna N; ponadto wszystkie stany maszyny N muszą wejść do alfabetu taśmowego maszyny M by móc je również trzymać taśmie. To wszystko jest wykonywalne, ale przykre w implementacji. □

[5]Patrz: sekcja 7.1.1.4

7.8 Omów złożoność obliczeniową problemu stopu oraz jego dopełnienia

7.8.1 Uniwersalna maszyna Turinga

By mówić o problemie stopu, musimy najpierw wprowadzić definicję uniwersalnej Maszyny Turinga – to znaczy takiej, że jest w stanie „zasymulować" działanie dowolnej innej MT za pomocą jakiegoś kodowania.

Niech $M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$ będzie (deterministyczną) maszyną którą chcemy symulować. Pokażemy jak zakodować M nad alfabetem $\Sigma_U = \{0, 1\}$.

- Niech $Q = \{q_1, \ldots, q_n\}$

  Kodujemy q_i jako ciąg i jedynek.

- $\sqcup$ będziemy kodować jako pojedynczą jedynkę.

- Podobnie niech $\Sigma = \{a_1, \ldots, a_n\}$

  Kodujemy a_i jako ciąg $i + 1$ jedynek.

- Dla $\Gamma = \{z_1, \ldots z_n\}$

  Kodujemy z_i jako ciąg $i + |\Sigma + 1|$ jedynek – dla odróżnienia od Σ.

- δ jest funkcją, czyli zbiorem par $((q_1, z_1), (q_2, z_2, \pm 1)$

  Parę $((q_i, z_j), (q_k, z_l), 1)$ zakodujemy jako

  $$\underbrace{1\ldots1}_{i}0\underbrace{1\ldots1}_{j}0\underbrace{1\ldots1}_{k}0\underbrace{1\ldots1}_{l}01$$

  Natomiast parę $((q_i, z_j), (q_k, z_l), -1)$ zakodujemy jako

  $$\underbrace{1\ldots1}_{i}0\underbrace{1\ldots1}_{j}0\underbrace{1\ldots1}_{k}0\underbrace{1\ldots1}_{l}011$$

  Całą deltę kodujemy kodując wszystkie jej pary (w dowolnej kolejności) i oddzielając je dwoma zerami.

- $F = \{q_{i_1}, \ldots q_{i_n}\}$ kodujemy jako

  $$\underbrace{1\ldots1}_{i_1}0\ldots0\underbrace{1\ldots1}_{i_n}$$

Całą maszynę M zapisujemy wpisując δ, q_0, F oddzielając je ciągiem 000.

Słowo $w = a_{i_1} \ldots a_{i_n}$ kodujemy jako

$$\underbrace{1\ldots1}_{i_1}0\ldots0\underbrace{1\ldots1}_{i_n}$$

Teraz musimy jeszcze powiedzieć jak kodujemy konfigurację maszyny M a robimy to dość prosto – Konfigurację $X_1 \ldots X_k q_i X_{k+1} \ldots X_n$ kodujemy jako

$$0\underbrace{1\ldots1}_{\text{kod } X_1}0\underbrace{1\ldots1}_{\text{kod } X_k}00\underbrace{1\ldots1}_{\text{kod } q_i}00\ \underbrace{1\ldots1}_{\text{kod } X_{k+1}}\ 0\underbrace{1\ldots1}_{\text{kod } X_n}0$$

Możemy założyć, że kodowanie M jest zapisane na taśmie wejściowej po której nigdy nie piszemy, zaś konfigurację M będziemy trzymać na osobnej taśmie.

Krok symulacji przebiega teraz następująco:

1. Zlokalizuj głowicę M (szukamy 00)
2. Znajdź w kodowaniu δ wpis odpowiadający przejściu na $q_i X_k$
3. Wykonaj zadane przejście – możemy wpisać nową konfigurację na osobną, tymczasową taśmę i przepisać ją z powrotem na taśmę na której operujemy.

7.8.2 Problem stopu

Definicja 7.8.1. Definiujemy język L_{HALT} następująco:

$$L_{HALT} = \{(M, w) : M \text{ zatrzymuje się na } w\}$$

Oczywiście, pisząc M mamy na myśli pewne kodowanie jakiejś Maszyny Turinga z którym jesteśmy w stanie pracować.

Twierdzenie 7.8.1. $L_{HALT} \in \mathrm{RE} \setminus \mathrm{R}$

Dowód. Oczywiście $L_{HALT} \in \mathrm{RE}$ bo możemy po prostu zasymulować M na w i jeśli się zatrzymamy to odpowiedź jest „TAK” i wszystko super. Jeśli zaś odpowiedź to „NIE” to nie musimy się w ogóle zatrzymywać.

Aby pokazać że $L_{HALT} \notin \mathrm{R}$ zakładamy nie wprost, że istnieje M z własnością stopu, taka że $L(M) = L_{HALT}$.

Konstruujemy teraz M' która:

1. wczytuje wejście x
2. symuluje M na (x, x)
3. jeśli M odpowiedziała „TAK” to wpadamy w nieskończoną pętlę
4. w przeciwnym razie zatrzymujemy się w dowolnym stanie

Wrzucamy teraz do M wejście (M', M')

Mamy dwie sytuacje:

- M zaakceptowała (M', M')

  Skoro $(M', M') \in L(M) = L_{HALT}$ to oznacza, że wykonaliśmy krok (4) co ma miejsce jedynie gdy $(x, x) \notin L(M)$. Tak się składa że u nas $x = M'$ czyli $(M', M') \notin L(M)$.

- M odrzuciła (M', M')

  Tutaj sytuacja jest podobna – skoro $(M', M') \notin L(M) = L_{HALT}$ to wykonaliśmy krok (3) co ma miejsce jedynie gdy $(x, x) \in L(M)$, a ponieważ $x = M'$ to $(M', M') \in L(M)$.

Łącząc oba powyższe dostajemy $(M', M') \in L(M) \iff (M', M') \notin L(M)$ co jest oczywiście sprzeczne.

W takim razie nie istnieje M z własnością stopu rozpoznająca L_{HALT}, zatem $L_{HALT} \notin \mathrm{R}$. □

7.8.3 Dopełnienie problemu stopu

Twierdzenie 7.8.2. $\overline{L}_{HALT} \notin \text{RE}$

Powyższe twierdzenie wynika wprost z poniższego lematu:

Lemat 7.8.1. Jeśli $L \in RE \setminus R$, to $\overline{L} \notin RE$

Dowód. Załóżmy, że $L \in RE \setminus R$ i $\overline{L} \in RE$. Mamy więc maszynę M rozpoznającą L oraz N rozpoznającą $\overline{L}$

Konstruujemy DMT M' która:

1. Wczytuje wejście w
2. Aż do akceptacji:
   (a) wykonaj jeden krok symulacji M na w
   (b) wykonaj jeden krok symulacji N na w
3. Jeśli M zaakceptowało to wypisz „TAK”
4. Jeśli N zaakceptowało to wypisz „NIE”

Oczywiście $w \in L \vee w \in \overline{L}$ więc albo M zaakceptuje w albo zrobi to N – któraś w końcu musi. Stanie się to po skończonej liczbie kroków niezależnie od w zatem $L(M') = M$ oraz M' ma własność stopu.

Trochę przypał bo w takim razie $L \in (RE \setminus R) \cap R$ czyli nie istnieje.

□

7.9 Omów klasy złożoności: PTIME, NPTIME oraz coNPTIME. Podaj przykład problemu, który jest w PTIME oraz przykłady jezyków zupełnych dla NPTIME i coNPTIME. Nakreśl dowód twierdzenia Cooke’a

7.9.1 Podstawowe definicje i obserwacje

Definicja 7.9.1. Mówimy, że maszyna M (deterministyczna lub nie) **działa w czasie** $T : \mathbb{N} \to \mathbb{N}$ jeśli dla każdej konfiguracji startowej q_0w każde obliczenie jest akceptujące lub odrzucające i ma długość co najwyżej $T(|w|)$

Definicja 7.9.2. Mówimy, że funkcja f jest **redukcją wielomianową** (Karpa) jeśli istnieje wielomian p oraz Maszyna Turinga obliczająca f w czasie p.

Jeśli istnieje redukcja wielomianowa z języka L_1 do języka L_2 to zapisujemy to jako $L_1 \leq_p L_2$

Definicja 7.9.3. Dla klasy problemów $C \subseteq R$ problem L jest **C-trudny** jeśli $\forall_{L' \in C} L' \leq_p L$

Definicja 7.9.4. Dla klasy problemów $C \subseteq R$ problem L jest **C-zupełny** jeśli jest C-trudny i $L \in C$

Lemat 7.9.1. Jeśli L_1 jest C-trudny i $L_1 \leq_p L_2$ to L_2 też jest C-trudny.

Dowód. Skoro L_1 jest C-trudny to znaczy, że dla każdego $L' \in C$ mamy redukcję f z L' do L_1. Mamy też redukcję g z L_1 do L_2.

Składając $g \circ f$ dostajemy redukcję z L' do L_2 co dowodzi, że L_2 jest trudniejszy niż dowolny $L' \in C$. □

7.9.2 PTIME

Definicja 7.9.5. Język L jest w klasie P (polynomial) jeśli istnieje wielomian p oraz Deterministyczna Maszyna Turinga działająca w czasie p taka, że $L(M) = L$

Twierdzenie 7.9.1. Język pusty jest w P.

Dowód. To było mało ambitne twierdzenie. Maszyna rozpoznająca ten język od razu przechodzi do stanu odrzucającego, a tym samym działa w czasie stałym. □

Twierdzenie 7.9.2. DNF-SAT ∈ P.

Dowód. To może przez chwilę szokować i powodować ból mózgu, ale, uwaga, jeśli problem SAT jest w postaci DNF to możemy to rozwiązać w czasie wielomianowym.

By to zrozumieć, wypada zobaczyć przykład formuły w DNF:

$$\overbrace{(x_1 \wedge \neg x_2 \wedge \neg x_3)}^{\text{Term DNF}} \vee \overbrace{(x_1 \wedge x_4)}^{\text{Term DNF}} \vee \overbrace{(\neg x_5)}^{\text{Term DNF}}$$

Obserwujemy, że aby formuła w DNF była spełnialna, *któryś* z jej termów musi być spełnialny.

...a kiedy ciąg koniunkcji *nie* jest spełnialny?

Wtedy i tylko wtedy, gdy zawiera w sobie jakąś zmienną i jej zaprzeczenie! Uzasadnienie tego faktu jest proste:

1. Jeśli w ciągu koniunkcji występuje zmienna i jej zaprzeczenie to nie istnieje takie wartościowanie by ten ciąg koniunkcji był prawdziwy, siłą rzeczy;
2. jeśli w ciągu koniunkcji *nie* występuje zmienna wraz z jej zaprzeczeniem, to wszystkie zmienne które występują zanegowane wartościujemy na fałsz, a wszystkie które występują niezanegowane wartościujemy na prawdę. W ten sposób otrzymujemy spełnialny term.

Zauważamy, że ten warunek jest (z perspektywy algorytmicznej) trywialny do zbadania w czasie wielomianowym od rozmiaru wejścia. Badamy więc ten warunek dla każdej klauzuli po kolei – jeśli którakolwiek z nich jest spełnialna, to cała formuła jest spełnialna. Ale super.

□

Twierdzenie 7.9.3. 2-SAT jest w P.

Dowód. To jest bardziej ambitne twierdzenie. Algorytm znamy z ASDów, ale akurat w wymaganiach egzaminacyjnych do ASD go nie ma (a więc zostanie przedstawiony tutaj).

Fundamentalna obserwacja w tym algorytmie opiera się na fakcie, że:

$$p \vee q \equiv (\neg p \implies q) \wedge (\neg q \implies p)$$

Aby popchnąć rozwiązanie tego problemu dalej, użyjemy *teorii grafów*.

Tworzymy sobie graf *literałów* dla formuły, którą dostaliśmy na wejściu. Na przykład dla takiej formuły:

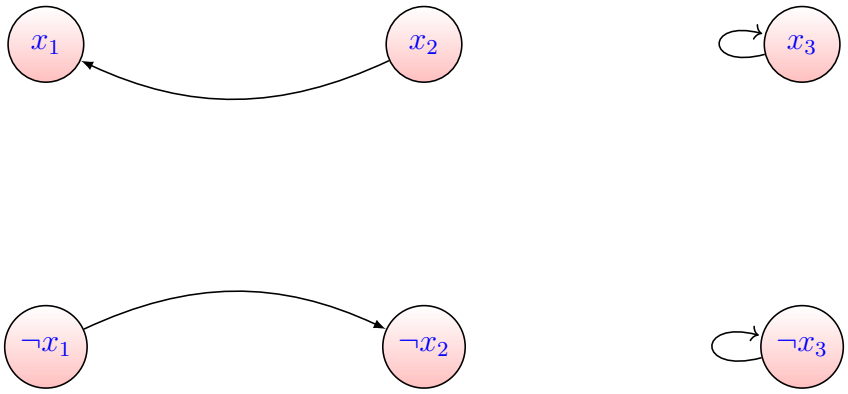
$$(x_1 \vee \neg x_2) \wedge (x_3 \vee \neg x_3)$$

graf ten będzie mieć 6 wierzchołków: $x_1, \neg x_1, x_2, \neg x_2, x_3, \neg x_3$.

Rozpiszmy sobie teraz tę formułę z przykładu korzystając z naszej wcześniejszej obserwacji:

$$(x_1 \vee \neg x_2) \wedge (x_3 \vee \neg x_3) \equiv (\neg x_1 \implies \neg x_2) \wedge (x_2 \implies x_1) \wedge (\neg x_3 \implies \neg x_3) \wedge (x_3 \implies x_3).$$

„Strzałki implikacji" zostają krawędziami w naszym grafie. Tak jest – konstruujemy graf implikacji literałów.



W tym grafie wyznaczamy silnie spójne składowe (jest to graf skierowany, rzecz jasna) i zastanówmy się jaką semantykę ma silnie spójna składowa w takim grafie.

Otóż – z definicji – w silnie spójnej składowej dla każdej pary wierzchołków do niej należących istnieje ścieżka. W naszym przypadku, oznacza to istnienie ciągu implikacji pomiędzy dowolną parą literałów (w jedną i drugą stronę). A to oznacza, że wszystkie literały w obrębie jednej silnie spójnej składowej są sobie **równoważne**. Innymi słowy – w obrębie danej spójnej składowej wszystkie literały muszą być zwartościowane tak samo jeśli formuła ma być prawdziwa.

Nazwijmy graf implikacji G, a formułę którą dostaliśmy na wejściu (i z której on powstał) φ. Wykonujemy fajną obserwację:

Lemat 7.9.2. φ jest spełnialna wtedy i tylko wtedy, gdy w żadnej silnie spójnej składowej G nie występuje zmienna wraz ze swoim zaprzeczeniem.

Dowód. Jeśli w obrębie danej silnie spójnej składowej występuje zmienna wraz ze swoim zaprzeczeniem, to w oczywisty sposób φ spełnialna nie będzie.

Zastanówmy się co się dzieje w sytuacji, gdy nie ma takiej silnie spójnej składowej w której występuje zmienna wraz ze swoim zaprzeczeniem. Otóż wtedy jest fajnie.

Wykonujemy standardowy trik i kompresujemy każdą silnie spójną składową do jednego wierzchołka; tak otrzymany graf silnie spójnych składowych jest DAGiem[6]. Skoro jest to DAG, możemy wykonać sortowanie topologiczne. Idziemy po tym grafie w odwrotnym porządku topologicznym i wartościujemy wierzchołki tego grafu (co ma taką interpretację, że wszystkie literały w obrębie danej silnie spójnej składowej są tak wartościowane). Jeśli dany wierzchołek nie ma przypisanego wartościowania to wartościujemy go na prawdę, a wierzchołek zawierający zanegowane zmienne od nas wartościujemy od razu na fałsz.

Trzeba niestety (siłą rzeczy) udowodnić, że ta procedura zawsze da nam prawidłowe wartościowanie.

Pierwsza obserwacja: dla dowolnej silnej spójnej składowej S, jeżeli „dostaje" ona wartościowane na prawdę, to istnieje **dokładnie** jedna silnie spójna składowa która zostanie zwartościowana na fałsz. Załóżmy nie wprost, że tak nie jest:

1. Jeśli zwartościowaliśmy na fałsz więcej niż jedną spójną, to oznaczałoby że mamy literały a, b takie, że $a \iff b$, ale $\neg a \not\iff \neg b$, co jest sprzeczne z tym jak konstruowaliśmy ten graf.
2. Analogiczny argument stosujemy by wykazać, że ta spójna którą obecnie chcemy zwartościować na fałsz nie była już tak wartościowana wcześniej podczas działania algorytmu.

Z tego mamy bijekcję między silnie spójnymi składowymi (na spójne które zawierają jakieś literały i spójne które zawierają dokładnie negacje tych literałów).

Teraz pokażemy, że algorytm przypisywania wartości działa poprawnie, to znaczy nie będzie takiej sytuacji gdzie po przypisaniu wartości „prawda" do jakiegoś wierzchołka będzie szła z niego krawędź do wierzchołka który już został zwartościowany na „fałsz".

Załóżmy nie wprost, że doszło do takiej sytuacji: mamy zatem literały x i y takie, że $x \implies y$, ale wartość y została ustawiona przez algorytm na fałsz, a x na prawdę. Zauważamy, że (ze wcześniejszej obserwacji) $\neg y$ zostało na pewno wcześniej zwartościowane na „prawdę", oraz $\neg y \implies \neg x$ z definicji naszego grafu. Ale przecież skoro teraz wartościujemy spójną składową z x, to jest ona później w porządku topologicznym niż spójna z $\neg y$ i wcześniej niż spójna z $\neg x$, a więc ścieżka z $\neg y$ do $\neg x$ istnieć nie może. Otrzymana sprzeczność pokazuje, że do takiej sytuacji dojść nie może. □

Innymi słowy, by stwierdzić czy 2-SAT jest spełnialny wystarczy sprawdzić czy istnieje jakaś „wewnętrznie sprzeczna" silnie spójna składowa w G. Aby explicite otrzymać wartościowanie, możemy użyć podanego algorytmu. Oczywiście, taki check (jak i ten algorytm) są wielomianowe, więc wszystko jest super.

□

7.9.3 NPTIME

Definicja 7.9.6. Język L jest w klasie NP (nondeterministic polynomial) jeśli istnieje wielomian p oraz Niedeterministyczna Maszyna Turinga działająca w czasie p.

Twierdzenie 7.9.4. SAT $\in$ NP

Dowód. Robimy NMT która „losuje" wartościowanie formuły, a następnie sprawdza, czy formuła jest prawdziwa dla takiego wartościowania. Jeśli tak to akceptuje, jeśli nie to odrzuca.

[6] Oczywiste.

Jeśli formuła jest spełnialna, to dla *któregoś* obliczenia uda się zaakceptować. □

7.9.4 Twierdzenie Cooka-Levina

Twierdzenie 7.9.5 (Cook-Levin)**.** Problem SAT jest NP-zupełny.

Dowód. Oczywiście SAT ∈ NP, co już udowodniliśmy wcześniej.

Pokażemy teraz, że SAT jest NP-trudny.

W tym celu musimy pokazać, że jeśli $L \in \text{NP}$ to istnieje redukcja z L do SAT. Skoro $L \in \text{NP}$ to istnieje niedeterministyczna Maszyna Turinga M, która go akceptuje.

Intuicyjnie – będziemy śledzić obliczenie prowadzone przez M na zadanym słowie wejściowym w i zapisywać je jako formułę logiczną.

M działa w czasie $p(|x|)$ zatem możemy wyobrazić sobie tablicę o wymiarach $p(|x|) \times p(|x|)$, w której wiersze reprezentują kolejne konfiguracje maszyny.

Możemy skonstruować formułę logiczną która będzie opisywała jakie są poprawne przejścia w konfiguracjach maszyny.

W tym celu będziemy potrzebowali jakoś kodować za pomocą formuł boolowskich, gdzie znajduje się jaka wartość na taśmie. Definiujemy sobie zatem zmienną x_{ija}. Będziemy usiłowali osiągnąć jej taką semantykę, żeby mówiła ona, że w i-tym wierszu (czyli i-tej w kolejności konfiguracji), j-tej kolumnie (czyli j-tym symbolu w konfiguracji) znajduje się symbol a.

Przy takiej interpretacji chcemy jeszcze dodać warunki na konfigurację startową i końcową w taki sposób, aby istnienie wartościowania formuły na 1 było równoważne istnieniu obliczenia akceptującego.

W tym celu będą potrzebne nam cztery rzeczy; każdą z nich wymusimy oddzielną formułą logiczną, a ich koniunkcja posłuży nam do przeprowadzenia udanej symulacji w całości.

Zanim jednak do tego przejdziemy, umawiamy się, że jak nasza NTM wejdzie w stan akceptujący, to ma zdefiniowane przejście które nic nie robi i idzie po prostu w prawo (i dalej jest akceptujące). Dlaczego tak robimy? Bo będzie mniej dziwnych przypadków później.

Nietrudno dla dowolnej NTM stworzyć równoważną która spełnia ten warunek, więc to założenie nie powinno nikogo boleć.

7.9.4.1 Wartości zmiennych są spójne

Zauważamy, że semantyka zmiennych x bardzo szybko spadłaby z rowerka, jeśli istniałyby 2 takie zmienne, że $x_{ija} = 1 \wedge x_{ijb} = 1$ (bo to by znaczyło że w jednym miejscu na taśmie są 2 symbole) lub gdyby $\forall_a x_{ija} = 0$ (bo to by znaczyło, że w jakimś miejscu na taśmie nie ma **nic** (blank również jest symbolem).

Aby zapobiec takim shenanigansom, definiujemy sobie formułę φ_1:

$$\varphi_1 = \bigwedge_{\substack{1 \le i \le p(n) \\ 1 \le j \le p(n)}} A_{ij} \wedge B_{ij}$$

Gdzie A_{ij} oraz B_{ij} odpowiadają, odpowiednio, formule wymuszającej by chociaż jeden symbol znalazł się na taśmie oraz formule wymuszającej, by nie było sytuacji gdzie w jednym miejscu jest ponad jeden symbol. Zdefiniujmy je zatem:

$$A_{ij} = \bigvee_{a \in \Gamma} x_{ija}$$

A_{ij} wymaga zatem, by *któryś* symbol był zwartościowany jako „prawda" (jeśli formuła ma mieć szanse się zwartościować na „prawdę").

$$B_{ij} = \bigwedge_{\substack{a \in \Gamma \\ b \in \Gamma \\ a \neq b}} \neg(x_{ija} \wedge x_{ijb})$$

B_{ij} z kolei zabrania, by jakiekolwiek 2 x_{ij*} zwartościowały się na „tak".

Zauważamy, że obie te formuły są wielomianowe, a jako że rozmiar naszej tablicy jest również wielomianowy względem wejścia wszystko jest absolutnie spoko.

7.9.4.2 Symulacja rozpoczyna się poprawnie

Aby w ogóle rozpocząć symulację NTM, musimy „zainicjalizować" jej konfigurację startową. Jest to akurat dosyć proste – wiemy, że na pierwszym miejscu znajduje się stan startowy q_s (umawiamy się, że wierzymy że NTM której taśma rozszerza się tylko w jedną stronę jest tak samo mocna jak taka, której taśma rozszerza się w obie strony; dowód dla czytelnika czy coś), dalej znajduje się słowo w, a potem blanki (tak by długość wiersza wynosiła $p(|w|)$.

Innymi słowy, robimy formułę wymuszającą początkowe wartościowanie:

$$\varphi_2 = a_{1,1,q_s} \wedge a_{1,2,w_1} \wedge a_{1,2,w_2} \wedge \cdots \wedge a_{1,k,\sqcup} \wedge a_{1,k+1,\sqcup} \wedge \cdots \wedge a_{1,p(|w|),\sqcup}$$

Długość formuły φ_2 jest oczywiście wielomianowa względem w.

7.9.4.3 Spełniamy jeśli NTM przeszła do stanu akceptującego

Biorąc pod uwagę naszą wcześniejszą umowę (że jeśli NTM zaakceptuje, to jedyne przejście które ma, to takie które zostawia je w stanie akceptującym i przesuwa w prawo), aby sprawdzić czy NTM akceptuje możemy po prostu sprawdzić czy którykolwiek ze znaków w ostatnim wierszu jest stanem akceptującym. Definiujemy zatem formułę:

$$\varphi_3 = \bigvee_{q_f \in F} \bigvee_{1 \leq j \leq p(|w|)} x_{p(|w|),j,q_f}$$

Jeśli gdziekolwiek znajdziemy jakiś stan akceptujący, jesteśmy w domu; jak nie, to znaczy że dane obliczenie nie akceptuje słowa (więc to wartościowanie jest do bani).

7.9.4.4 Przejścia są wykonywane poprawnie

To jest najciekawsza część dowodu – nareszcie nadszedł ten moment, że musimy zaimplementować jakoś przejścia.

Zauważamy, że przejście głowicy powoduje jedynie lokalną zmianę konfiguracji – z konfiguracji na konfigurację zmienią się maksymalnie 3 symbole (stan może ulec zmianie, znak na prawo od wcześniej lokacji głowicy może się zmienic, oraz głowica może przesunąć się w lewo, zmieniając swoją kolejnosć z literką po lewej).

Musimy zatem oddać każde możliwe przejście z σ – to jest to miejsce, gdzie wiele dowodów się poddaje i zaczyna machać, ale nie my.

Będzie nam za niedługo potrzebna formuła o semantyce „wymuś, by symbole na koordynatach (i_1, j_1) oraz (i_2, j_2) były sobie równe":

$$E(i_1, j_1, i_2, j_2) = \bigwedge_{a \in \Gamma} ((x_{i_1,j_1,a} \wedge x_{i_2,j_2,a}) \vee (\neg x_{i_1,j_1,a} \wedge \neg x_{i_2,j_2,a}))$$

Zdefiniujmy sobie formułę „wymuszającą przejście w prawo dla stanu q i znaku a, na koordynatach i, j", którą nazwiemy $T_r(q, a, i, j)$[7]:

$$\begin{aligned} &T_r(q, a, i, j) \\ &\quad = \bigvee_{(q', a', 1) \in \delta(q,a)} ((x_{i,j,q} \wedge x_{i,j+1,a}) \implies (E(i, j-1, i+1, j-1) \wedge x_{i+1,j,a'} \wedge x_{i+1,j+1,q'})) \end{aligned}$$

Zauważmy, że ta semantyka nawet ma sens: jeśli pod współrzędnymi i, j jest stan q, to jeśli mamy przesunąć się prawo, to chcemy aby zmienna na lewo była niezmieniona, na jego miejsce wskoczyła zmodyfikowana zmienna (która wcześniej była po jego prawej) a on sam zmienił swój stan na nowy.

Bardzo analogicznie zrobimy dla przesunięcia w lewo:

$$\begin{aligned} &T_l(q, a, i, j) \\ &\quad = \bigvee_{(q', a', -1) \in \delta(q,a)} ((x_{i,j,q} \wedge x_{i,j+1,a}) \implies (x_{i+1,j-1,q'} \wedge E(i, j-1, i+1, j) \wedge x_{i+1,j+1,a'})) \end{aligned}$$

Definiujemy teraz formułę odpowiedzialną za zebranie tego wszystkiego do kupy, by zapewnić by przejścia były poprawne:

$$C = \bigwedge_{\substack{q \in Q \\ a \in \Gamma \\ 1 \le i \le p(|w|) \\ 1 \le j \le p(|w|)}} T_r(q, a, i, j) \vee T_l(q, a, i, j)$$

C jest wielomianowa względem rozmiaru wejścia, bo rozmiar Q i Γ to stałe, więc dla każdego pola w tablicy (których jest wielomianowo wiele) dodajemy stale wiele formuł mających stałe długości.

Zauważmy jednak, że to nie koniec naszych przygód, a dopiero początek; musimy bowiem jeszcze wymyślić sprytny sposób, by w sytuacji gdzie wartości są poza tym lokalnym *miejscem niebezpiecznym* się bardzo elegancko przepisały.

[7]Osoby zaniepokojone użyciem implikacji uspokajam, że $p \implies q \equiv \neg p \vee q$.

Okazuje się, że możemy to nawet zakodować; będziemy potrzebować dodatkowego klocka, którym jest formuła logiczna sprawdzająca, czy coś **nie** jest stanem:

$$NQ(i,j) = \bigwedge_{q \in Q} \neg x_{ijq}$$

Jest to bardzo użyteczne, bo zauważam, że mogę „przepisać" dany znak z konfiguracji na konfigurację poniżej wtedy i tylko wtedy, gdy nie jest on stanem i nie ma stanu obok siebie (po lewej lub prawej). Jeśli jest stanem (lub ma go w pobliżu) to tym co ma znajdować się na konfiguracji poniżej zajęła się już formuła C.

Definiujemy więc formułę wymuszającą przepisanie symbolu z konfiguracji na konfigurację poniżej:

$$K(i,j) = (NQ(i,j-1) \wedge NQ(i,j) \wedge NQ(i,j+1)) \implies E(i,j,i+1,j)$$

Rozciągamy to na wszystkie możliwe współrzędne:

$$D = \bigwedge_{\substack{1 \le i \le p(|w|) \\ 1 \le j \le p(|w|)}} K(i,j)$$

D jest wielomianowa względem wejścia, bo $K(i,j)$ ma stałą długość, a z kolei tych formuł jest wielomianowo wiele (bo na każdą komórkę naszej gigatablicy przypada jedna).

I, koniec końców, otrzymujemy wyrażenie wymuszające poprawne przejścia:

$$\varphi_4 = C \wedge D$$

φ_4 również ma rozmiar wielomianowy, jako że jest koniunkcją dwóch formuł długości wielomianowej.

Proste, nie?

7.9.4.5 Pozbieranie do kupy

Teraz zbieramy to wszystko do kupy i otrzymujemy ostateczne gigawyrażenie:

$$\varphi = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$$

Jest ono spełnialne wtedy i tylko wtedy, gdy NTM akceptuje słowo w, a jego konstrukcja jest wielomianowa. Tym samym dowiedliśmy tw. Cooka-Levina. □

7.9.5 coNPTIME

Definicja 7.9.7. $L \in \text{coNP} \iff \overline{L} \in \text{NP}$.

Lemat 7.9.3. L jest C-trudny wtedy i tylko wtedy, gdy $\overline{L}$ jest coC-trudny.

Dowód. Dowodzimy równoważność w obie strony:

- $\Longrightarrow$

  Weźmy sobie $L' \in C$. Jako, że L jest trudne, to $L' <_p L$. Wiemy wobec tego, że istnieje taka funkcja f, obliczalna w czasie wielomianowym, że $x \in L' \iff f(x) \in L$.

  Równoważnie, $x \notin L' \iff f(x) \notin L$. No ale to z kolei oznacza, że $x \in \overline{L'} \iff f(x) \in \overline{L}$. Wobec tego, dla każdego problemu należącego do *coC* jest tak, że redukuje się on do $\overline{L}$, czyli $\overline{L}$ jest *coC*-trudny.

- $\Longleftarrow$ Centralnie piszemy to samo, ale w drugą stronę.

  Bierzemy sobie $L' \in coC$. Z racji faktu, że L jest trudne mamy, że $L' <_p \overline{L}$. To oznacza, że istnieje funkcja f, taka że $x \in L' \iff f(x) \in \overline{L}$.

  Równoważnie, $x \notin L' \iff f(x) \notin \overline{L}$. Z tego wiemy, że $x \in \overline{L'} \iff f(x) \in L$. Wobec tego, każdy problem z C redukuje się do L, czyli L jest C-trudne.

□

Pokażemy teraz problem coNP-zupełny.

Definicja 7.9.8. Problem TAUTOLOGY definiujemy jako problem w którym:

- wejście: formuła rachunku zdań φ
- wyjście: Czy φ jest tautologią tj. czy każde wartościowanie wartościuje φ na 1 ?

Twierdzenie 7.9.6. TAUTOLOGY jest coNP-zupełny.

Dowód. Pokażemy że $\overline{\text{TAUTOLOGY}}$ jest NP-zupełny.

Mamy zatem formułę φ i chcemy sprawdzić czy tautologią nie jest, czyli czy istnieje wartościowanie v takie, że $v(\varphi) = 0$.

Nie jest trudno zauważyć, że $v(\neg\varphi) = 0 \iff v(\varphi) = 1$.

Z tego mamy, że:

1. SAT $<_p \overline{\text{TAUTOLOGY}}$, bo dostając zapytanie czy formuła φ jest spełnialna możemy ją zanegować i sprawdzić czy po zanegowaniu *nie* jest tautologią (jeśli nie było świadków spełnialności, to formuła będzie tautologią; jeśli byli, to nie będzie). Z tego mamy, że $\overline{\text{TAUTOLOGY}}$ jest NP-trudny, bo zredukowaliśmy do niego inny problem NP-trudny.
2. $\overline{\text{TAUTOLOGY}} <_p$ SAT, bo dostając zapytanie czy formuła φ *nie* jest tautologią możemy ją zanegować i sprawdzić czy jest spełnialna. Rozumowanie analogiczne do tego z poprzedniego punktu. Z tego otrzymujemy, że $\overline{\text{TAUTOLOGY}}$ jest w NP, bo możemy przeprowadzić wielomianową redukcję tego problemu do problemu z klasy NP.

Skoro $\overline{\text{TAUTOLOGY}}$ jest w NP oraz jest NP-trudny, to wiemy że jest on NP-kompletny. A skoro tak, to TAUTOLOGY musi być coNP-kompletny, co należało dowieść. □