# Matematyka dyskretna

Andrzej M. Borzyszkowski

Instytut Informatyki
Uniwersytet Gdański

sem. zimowy 2025/26

inf.ug.edu.pl/~amb/MaD





# Funkcje boolowskie





## Funkcje boolowskie

- dowolna funkcja $f : 2^n \to 2^m$
- dowolna algebra Boole'a jest izomorficzna z algebrą $2^\ell$, dla pewnego $\ell$, funkcje j.w. wystarczają do badania problemów funkcji boolowskich
- przykłady funkcji boolowskich (następny slajd):
  - funkcja sortująca $Sort : 2^4 \to 2^4$, $Sort(p, q, r, s) = (T_4^4(p, q, r, s), T_3^4(p, q, r, s), T_2^4(p, q, r, s), T_1^4(p, q, r, s))$ gdzie funkcje $T_j^i$ liczą, czy jest co najmniej $j$ jedynek w ciągu $i$ bitów
  - w wyniku $Sort$ otrzymujemy ciąg 4 bitów uporządkowanych od 0 do 1
  - funkcja $Sbox : 2^4 \to 2^3$, tzw. *substitution box*, używana w definicji szyfru DES (czyli funkcji $2^{56} \times 2^{64} \to 2^{64}$)





| $p$ | $q$ | $r$ | $s$ | $Sort(p, q, r, s)$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0000 |
| 0 | 0 | 0 | 1 | 0001 |
| 0 | 0 | 1 | 0 | 0001 |
| 0 | 0 | 1 | 1 | 0011 |
| 0 | 1 | 0 | 0 | 0001 |
| 0 | 1 | 0 | 1 | 0011 |
| 0 | 1 | 1 | 0 | 0011 |
| 0 | 1 | 1 | 1 | 0111 |
| 1 | 0 | 0 | 0 | 0001 |
| 1 | 0 | 0 | 1 | 0011 |
| 1 | 0 | 1 | 0 | 0011 |
| 1 | 0 | 1 | 1 | 0111 |
| 1 | 1 | 0 | 0 | 0011 |
| 1 | 1 | 0 | 1 | 0111 |
| 1 | 1 | 1 | 0 | 0111 |
| 1 | 1 | 1 | 1 | 1111 |

| $p$ | $q$ | $r$ | $s$ | $Sbox(p, q, r, s)$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 101 |
| 0 | 0 | 0 | 1 | 010 |
| 0 | 0 | 1 | 0 | 001 |
| 0 | 0 | 1 | 1 | 110 |
| 0 | 1 | 0 | 0 | 011 |
| 0 | 1 | 0 | 1 | 100 |
| 0 | 1 | 1 | 0 | 111 |
| 0 | 1 | 1 | 1 | 000 |
| 1 | 0 | 0 | 0 | 001 |
| 1 | 0 | 0 | 1 | 100 |
| 1 | 0 | 1 | 0 | 110 |
| 1 | 0 | 1 | 1 | 010 |
| 1 | 1 | 0 | 0 | 000 |
| 1 | 1 | 0 | 1 | 111 |
| 1 | 1 | 1 | 0 | 101 |
| 1 | 1 | 1 | 1 | 011 |

## Zapis tabelaryczny funkcji boolowskiej

- funkcja $f : 2^n \to 2^m$ wymaga podania $2^n$ wartości, każda $m$ bitów, razem $2^n \cdot m$
- zdefiniowanie funkcji $Sbox : 2^4 \to 2^3$ wymaga zapisania $16 \cdot 3$ bitów, czyli 6 bajtów
- w szyfrze DES występuje 16 funkcji $S : 2^8 \to 2^6$ wymagających łącznie $16 \cdot 2^8 \cdot 6$ bitów czyli 3 kB
- 

| funkcja | zapis | wielkość |
|---|---|---|
| $2^{16} \to 2^{16}$ | $2^{16} \cdot 16$ bitów | 128 kB |
| $2^{32} \to 2^{32}$ | $2^{32} \cdot 32$ bitów | 16 GB |
| $2^{64} \to 2^{64}$ | $2^{64} \cdot 64$ bitów | 128 mln TB |

- nie ma możliwości praktycznych zapisania funkcji $S : 2^{64} \to 2^{64}$
- szyfr DES wymaga podzielenia bloku 64 bitów na fragmenty, odrębnego przekształcenia fragmentów i odpowiedniego wymieszania bitów z podbloków





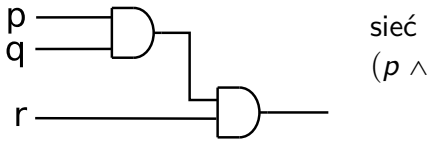
## Definicja funkcji boolowskiej

- funkcja boolowska może być zdefiniowana wzorem pozwalającym obliczyć jej wartość dla dowolnego argumentu
- zapis wzoru może być nieoczywisty, ale zależy raczej od długości liczby bitów na wejściu $n$, a nie od liczby możliwych wartości $2^n$
- definicja może nawet nie zależeć od długości $n$ wejścia, może opisywać wartości w zależności od niewielkiego fragmentu danych wejściowych
- funkcja sortująca zapisuje się za pomocą funkcji $T_4^4 = pqrs$, $T_3^4 = pqr + pqs + prs + qrs$, $T_2^4 = pq + pr + qr + ps + qs + rs$, $T_1^4 = p + q + r + s$
- przy tej liczbie argumentów zapis jest bardziej wymagający niż 8 bajtów
- ale realne są podobne wzory dla 64 bitów i więcej

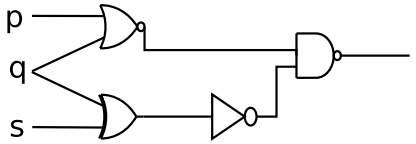




## Sieci boolowskie




sieć przedstawiająca koniunkcję $(p \wedge q) \wedge r$




$(p \text{ nor } q) \text{ nand } \neg(q \oplus s)$

- operator (bramka) NAND wystarcza do zdefiniowania wszystkich operatorów
- w teorii obwodów elektrycznych nawet $\neg(\neg p)$ ma znaczenie, wartość jest $p$ ale obliczenie tego działa jak opóźnienie (bufor)
- używane są bramki $\wedge$, $\vee$, $\oplus$, $\neg$ oraz ich negacje: nand, nor, nxor i bufor

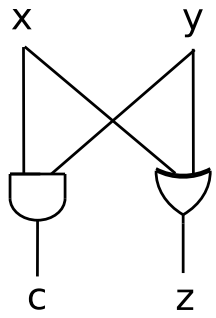




## Przykład: sumator

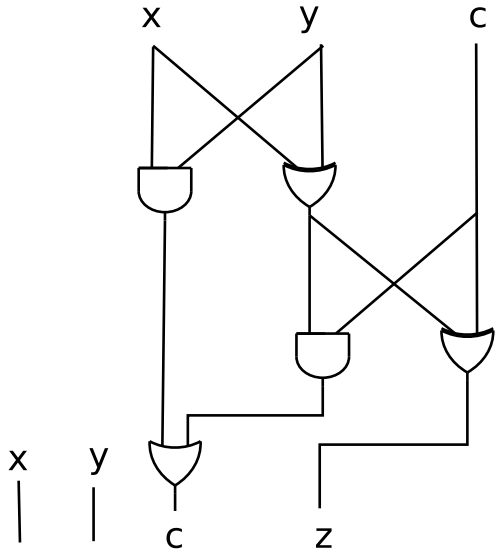
- zadanie: zaimplementować dodawanie liczb w zapisie binarnym
- $x = x_{31}x_{30}\ldots x_0$, $y = y_{31}y_{30}\ldots y_0$, $z = x + y \mod 2^{32} = z_{31}\ldots z_0$
- dodawanie jest modulo $2^{32}$, jeśli suma przekroczy tę wartość, bit przeniesienia jest ignorowany
- półsumator: $ha : 2^2 \to 2^2$, $ha(x_0, y_0) = (x_0 \oplus y_0, x_0 \wedge y_0)$ dodawaniem modulo 2 oraz bit przeniesienia
- sumator: ma *trzy* argumenty, dwa kolejne bity i bit przeniesienia
- $fa : 2^3 \to 2^2$, $fa(x, y, c) = (h_1(h_1(x, y), c), h_2(x, y) \oplus h_2(h_1(x, y), c))$ nie grożą nam dwa przeniesienia, bo jeśli $h_2(x, y) = 1$, to $x \wedge y$ więc $h_1(x, y) = 0$ i również $h_2(h_1(x, y), c) = 0$ $h_2(x, y) \oplus h_2(h_1(x, y), c) = h_2(x, y) \vee h_2(h_1(x, y), c)$

## Półsumator i sumator




w półsumatorze dodawane są dwa bity, jest wynik dodawania i przeniesienie

w sumatorze obsługiwany jest jeszcze bit z poprzedniego przeniesienia








## Operacje na wektorach

- zmienne boolowskie (flagi, semafory): w programie chcemy przechowywać wartość wielu zmiennych oznaczających zachodzenie warunków
- można te zmienne połączyć w wektor $x_n x_{n-1} \dots x_0$,
  - jeśli krok w programie zmienia wartość niektórych flag, to $x \mapsto x \oplus y$, gdzie działanie jest bitowe, a $y = y_n y_{n-1} \dots y_0$ ma jedynki dla flag, które zmieniamy
  - sprawdzenie dyzjunkcji wszystkich flag to sprawdzenie, czy $x \neq 0$
  - koniunkcję można zdefiniować poprzez prawa deMorgana, czy $\tilde{x} = 0$ gdzie $\tilde{x}$ jest bitową negacją
  - wybór niektórych flag, to $x \mapsto x \,\&\, y$, bitowa koniunkcja, a $y = y_n y_{n-1} \dots y_0$ ma jedynki dla wybranych flag (maska)





## Operacje na wektorach c.d.

- chcemy zaimplementować grę w szachy
- strategia itp. poza zakresem tego tematu
- reprezentacja stanu gry (nieoszczędna) to zestaw map $8 \times 8$ dla każdej z 32 figur zawierających po jednym bicie niezerowym
- zasady gry to m.in. dozwolone ruchy, dla każdego rodzaju figury i każdego aktualnego miejsca jest mapa możliwych ruchów ($7 \cdot 64 \cdot 8 = 3584$ bajtów)
- mapa możliwych ruchów jako maska służy do oceny możliwych ruchów w danym stanie





## Szyfr jednorazowy (*one-time pad*)

- szyfrowanie $E(k, m) = k \oplus m$, gdzie $k$ jest kluczem, $m$ jest wiadomością, każde jest ciągiem bitów tej samej długości $n$
- odszyfrowywanie $D(k, E(k, m)) = k \oplus E(k, m)$
- jedyny szyfr doskonale bezpieczny
- dw.: dany kryptogram $c$, pytanie, czy tekst $m$ może być zaszyfrowaną wiadomością?
  odpowiedź: tak, może być, mianowicie dla klucza $k = m \oplus c$
  każdy tekst $m$ być może jest tym zaszyfrowanym, czyli znajomość kryptogramu nic nam nie daje
- jedyną wiadonością przekazaną w takim szyfrze jest długość tekstu jawnego

## Szyfr jednorazowy c.d.

- klucz musi być długości tekstu
- jedyny pożytek, że mógł być wcześniej przekazany
- czy można ten sam klucz użyć wiele razy?
- jeśli przeciwnik zna $k \oplus m$ oraz $k \oplus m'$, to zna $m \oplus m'$
- tekst jawny nie jest losowym ciągiem bitów
  jeśli znanych jest kilka kryptogramów z użyciem tego samego klucza, to odtworzenie klucza jest prawie pewne
- szyfr bardzo kosztowny, ma sens głównie do szyfrowania kluczy sesyjnych





## Odciski (*fingerprints*)

- chcemy sprawdzić, czy tekst $m$ nie zmienił się w transmisji lub podczas przechowywania na dysku, $m \in 2^n$
- najprostsza wersja – bit parzystości $Par(m) = \bigoplus_{i=0}^{n-1} m_i$
- prawdopodobieństwo $P(Par(m) = Par(m') | m' \neq m) \approx p^2$
  $p$ jest prawdopodobieństwem przekłamania jednego bitu $m$ w transmisji, $m'$ jest wersją odczytaną
  jeśli $p$ jest bardzo małe, $p^2$ jest zaniedbywalne
- jeśli modyfikacji $m$ dokonuje celowo przeciwnik, to łatwo może zmienić takie bity, żeby zachodziła równość
- jeśli klucz $k$ jest maską $k \neq 0$ i przeciwnik nie zna $k$, prawdopodobieństwo, że $Par(m \,\&\, k) = Par(m' \,\&\, k)$ jest $1/2$
- potwierdzając równość dla kilkunastu/kilkudziesięciu masek możemy nabrać przekonania, że nie doszło do modyfikacji wiadomości





## Podział sekretu (*secret sharing*)

- dany sekret $k \in 2^n$, jest to tajny klucz kryptograficzny
- chcemy podać kilku osobom udziały w kluczu, t.j. ciągi bitów $k_i$, $i = 0, \ldots, \ell - 1$, tak by
  - znajomość wszystkich ciągów $k_i$, $i = 0, \ldots, \ell - 1$, gwarantowała odtworzenie $k$
  - brak choćby jednego udziału nie pozwalał na uzyskanie jakiejkolwiek wiedzy o $k$
- ujawnienie $n$ osobom po jednym bicie z $k$ jest złym pomysłem
  przeciwnik z zewnątrz musi przeszukać $2^n$ kluczy,
  jeden uczestnik podziału ma już $2^{n-1}$ kluczy do zbadania,
  ale $n - 1$ uczestników musi zbadać tylko dwie możliwości





## Podział sekretu c.d.

- losujemy $\ell - 1$ ciągów $k_1, k_2, \ldots, k_{\ell-1}$ bitów (rozkład prawdopodobieństwa jest jednostajny, losowania są niezależne)
- definiujemy $k_0 = \bigoplus_{i=1}^{\ell-1} k_i \oplus k$
- wówczas $\bigoplus_{i=0}^{\ell-1} k_i = k_0 \oplus \bigoplus_{i=1}^{\ell-1} k_i = k$ – można odtworzyć klucz $k$
- dowolny zbiór udziałów spośród $k_1, \ldots, k_{\ell-1}$ nie ma nic wspólnego z $k$
- ale dowolny podzbiór udziałów włącznie z $k_0$ daje w sumie $k \oplus k'$, gdzie $k'$ jest sumą brakującego podzbioru $k_1, \ldots, k_{\ell-1}$
- znajomość $k \oplus k'$ nie daje najmniejszej wiedzy o $k$, zgadywanie $k$ jest równoważne zgadywaniu $k'$, a ten wektor jest losowy

## Gra „Nim” (wikipedia.org/wiki/Nim)

- jest kilka kupek kamieni, np. trzy, po kilka kamieni w każdej kupce
  gracze naprzemian zabierają jakieś kamienie, zawsze z jednej kupki
  wygrywa gracz, który weźmie ostatni kamień
- stan gry = liczby kamieni w każdej kupce, np. $\langle x, y, z \rangle$
- tw. jeśli $x \oplus y \oplus z \neq 0$ to gracz mający ruch może wygrać
- dw. niech $r = x \oplus y \oplus z$
  - pewien bit jest najstarszym niezerowym bitem
  - co najmniej jedna z liczb $x, y, z$ np. $x$ musi mieć w tym samym miejscu bit niezerowy
  - $x \oplus r < x$ można pobrać $x \oplus r$ kamieni z tej kupki
  - nowy stan gry będzie równy $\langle x \oplus r, y, z \rangle$ i $(x \oplus r) \oplus y \oplus z = 0$
  - albo oznacza to puste kupki i gracz właśnie wygrał,
    albo po dowolnym ruchu drugiego gracza $\oplus$ wszystkich liczb
    znowu będzie niezerowy i pierwszy gracz kontynuuje grę