Булеви функции

"There are 10 types of people - those who understand binary and those who don't"

Юни 2025

Преди задачите е представена малко теоретична основа за различни теми в булевата алгебра и булевите функции, а също са разгледани основни похвати за решаването на някои типове задачи. Една част от материала в тази тема (отбелязан със *) излиза извън рамките на курса и е по-скоро допълнение.

Забележка. Възможно е настоящето на места да се отклонява от строгата формалност за булеви функции.

1 Булеви функции

Дефиниция. Дефинираме множеството от булевите функции на n аргумента като $\mathcal{F}_n := \{f \mid f : \{0,1\}^n \to \{0,1\}\}$ и множестовто от всички булеви функции $\mathcal{F} := \bigcup_{n\in\mathbb{N}} \mathcal{F}_n$.

Нотация. Макар съответствието с логическите съюзи да е голямо, има известни различия в записа/названията на някои функции:
- Вместо $x \wedge y$ понякога се ползва $x\&y$, както и xy;
- $x \oplus y$ тук е *сума по модул 2*, ползва и записът $x + y$
- $\overline{x}$ е съответно отрицанието на x;
- $x \downarrow y$ (NOR), *Стрелка на Пърс*;
- $x \mid y$ (NAND),*Черта на Шефър*;

Свойство 1. Свойствата на булевите функции на един и два аргумента са аналогични на тези на логическите операции. Да обърнем обаче внимание на: $x \oplus y = \overline{x}y \vee x\overline{y}$, $x|y = \overline{xy}$, $x \downarrow y = \overline{x \vee y}$.

Дефиниция (*симетрична функция*)**.** Функция на n променливи е симетрична тстк стойността ѝ е една и съща за всяка пермутация на променливите ѝ.

Свеждане в СДНФ и СКНФ

Дефиниция (*литерал*)**.** Литерал е синтактичен обект, име на променлива (потенциално с черта отгоре).

Дефиниция (*конюнктивна клауза*)**.** Непразна формула, съставена от конкатенация на литерали така, че всяко име на променлива се появява не повече от веднъж (с или без черта).

Дефиниция (*пълна конюнктивна клауза*)**.** Конюнктивна клауза, в която всяко име на променлива участва.

//TO DO

Полином на Жегалкин

По-надолу ще покажем, че множеството от булевите функции $\{\wedge, \oplus, \mathbf{1}\}$ (което също записваме и като $\{., +, \mathbf{1}\}$) е пълно, т.е. всяка булева функция може да бъде "представена" чрез тези три. Това представяне е именно полиномът на Жегалкин (наричан още *алгебрична нормална форма*). Ще дадем алгоритъм за получаване на такова представяне.
С цел по-добро разбиране на материята, може би тук е добре да се отбележи, че $(\mathbb{Z}_2, +, .) = (\mathbb{Z}_2, \oplus, \wedge)$ е комутативен пръстен с единица, откъдето полиномите $R[x]$ и изобщо $R[x_1, \cdots, x_n]$ са също комутативни пръстени с единица.

Алгоритъм:

1. **Намираме СДНФ за дадената функция**

2. **Замяна:** В СДНФ формулата заменяме:
   - всеки *отрицателен литерал* $\overline{x}$ с $x + \mathbf{1}$ (просто защото $\overline{x} = x \oplus \mathbf{1}$);
   - всеки символ $\vee$ за дизюнкция с такъв за *сума по модул 2*, $+$;

3. **Опростяване:** В получения израз разкриваме скобите по познатия от алгебрата начин. В контекста на булевата алгебра $x^k := \underbrace{x.x.\cdots x}_{k} = \underbrace{x \wedge x \wedge \cdots x}_{k} = x$, тоест всякакви степени са излишни и могат да бъдат премахнати от крайния вид на полинома. Знаем също, че $x + x = x \oplus x = \mathbf{0}$ - с други думи еднаквите изрази се "унищожават" (било то и по-сложни като например $xyz + xyz$) и също могат да бъдат премахнати.

Пример: Търсим полиномът на Жегалкин за булевата формула $x\overline{y} \vee zy$.
Ще използваме наготово, че СДНФ за дадената е $f(x, y, z) = \overline{x}yz \vee x\overline{y}\overline{z} \vee x\overline{y}z \vee xyz$.

Замяна: Преобразуваме $\overline{x}yz \vee x\overline{y}\overline{z} \vee x\overline{y}z \vee xyz \Rightarrow (x + 1)yz + x(y + 1)(z + 1) + x(y + 1)z + xyz$

Опростяване: $(x+1)yz+x(y+1)(z+1)+x(y+1)z+xyz = \cancel{xyz}+yz+[x(y+1)(z+1)+x(y+1)z]+\cancel{xyz} = yz + [x(y + 1)(\cancel{z} + 1 + \cancel{z})] = yz + x(y + 1)1 = yz + x(y + 1) = \boxed{\text{yz+xy+x}}$

Коректност:
Възниква въпросът защо това работи. За намирането на СДНФ, опростяването и замяната на $\overline{x}$ с $(x+1)$ е ясно, че не променят семантиката на формулата, т.е. те са един вид еквивалентни преобразувания. Интересно е обаче защо втората замяна (на $\vee$ с $+$) в СДНФ е коректна и дава еквивалентна формула. В контекста на дадения пример, чудим се защо $\overline{x}yz \vee x\overline{y}\overline{z} \vee x\overline{y}z \vee xyz = \overline{x}yz + x\overline{y}\overline{z} + x\overline{y}z + xyz$ (с напомнянето, че $+$ е $\oplus$). Това можем да в следната теорема:

1: Жегалкин

Нека f е булева формула в СДНФ, а ϕ е формула, получена от f при замяна на символа "$\vee$" с "$+$". Тогава f и ϕ имат еднаква семантика.

Доказателство. Ясно е, че при тази конструкция семантиката на конюнктивните клаузи на двете формули е еднаква (защото те са и синтактично еднакви). Това показва, че ако някоя от конюнктивните клаузи има ϕ има стойност 1, то същата клауза ще има стойност 1 във f, откъдето f ще приеме стойност 1. Това автоматично отхвърля случай, в който за дадена валюация f има стойност 0, а ϕ стойност 1. Остава въпросът дали е възможно обратното - да съществува остойностяване на променливите на функциите, за което f има стойност 1, а ϕ съответно 0. Да допуснем, че такова има. Понеже f има стойност 1, то поне една от конюнктивните клаузи има (семантика, съответстваща на) стойност 1. Но ϕ има стойност 0, което означава, че има четен брой конюнктивни клаузи със семантика 1, т.е. поне 2. Не е трудно да се види, че последното е невъзможно. Причината е, че в СДНФ клаузите са *пълни* и всеки две имат поне един различен литерал на една и съща променлива (едната клауза съдържа x, а другата - $\overline{x}$), тоест няма как едновременно да имат стойност 1. Противоречие. Значи за всяка валюация двете формули имат еднаква стойност, тоест тяхната семантика е еднаква, еквиваленти са. ∎

Забележка. Ако конюнктивните клаузи на f не бяха пълни, тази еквивалентност нямаше да е в сила!

Пълнота на множества булеви функции

Дефиниция. (*затваряне на множество от булеви функции*) Затваряне на множество от булеви функции $F \subseteq \mathcal{F}$ наричаме най-малкото множество, което: съдържа F, съдържа всички проекции $\pi_n^i(x_1, \cdots, x_n) = x_i$ и е затворено относно *суперпозиция*, т.е. композиция и *идентифициране* (ползване на една и съща променлива на различни места).
Казано неформално, $[F]$ е множеството от всички, функции, които могат да се построят от тези в F.

Дефиниция. (*пълно множество булеви функции*) Множество $F \subseteq \mathcal{F}$ е *пълно*, ако $[F] = \mathcal{F}$.

повече информация тук

2: Теорема на Boole

Множеството от трите булеви фунцкии негация, конюнкция, дизюнкция е пълно.

3: Условие за пълнота

Ако $G, F \subseteq \mathcal{F}$ (множеството от всички булеви функции). Тогава, ако F е пълно и $\forall f \in F : f \in [G]$, то и G е пълно.

В задачи понякога се налага да определим дали дадено множество от булеви функции G е пълно. Ако е, достатъчно е да покажем как чрез функциите от G могат да бъдат изразени функциите на някое пълно множество F (обикновено това е $\{\vee, \neg\}$ или $\{\wedge, \neg\}$). В този случай решението следва резултата от втората теорема (*условието за пълнота*).
Как обаче се доказва обратното - че множество G от булеви функции не е пълно? - Най-общият отговор на това е "*със структурна индукция* по построение на булевите формули", но това след малко. Тук ще покажем три подхода, които могат да се ползват за доказване на непълнота, а в *задача 9* ще ги видим и приложени на практика.
Обща схема за доказване: Преди да се спрем върху конкретните подходи, да дадем някаква интуиция. Общата идея за доказателството е проста - показваме, че всички функции, които можем да получим като композиция на тези от G, се държат еднакво, т.е. изпълняват някакво свойство (предикат P), $\forall g \in [G] : P(g)$. Същевременно показваме, че съществува булева функция, която не изпълнява това свойство ($\exists f \in \mathcal{F} : \neg P(g)$). Е, тогава е ясно, че $[G] \neq \mathcal{F}$, значи G не е пълно. *Обикновено предикатът P е принадлежност към някой клас функции - монотонни, линейни, запазващи 0/1, самодвойствени и др.*

1. Доказателство с остойностяване: Това е частен случай на горния подход. С него може да се докаже непълнота на функции, запазващи 1 (респективно 0), т.е. такива, за които $f(1, 1, \cdots, 1) = 1$. Идеята е, че ако при фиксиране на някои входни стойности (т.е. подаване на конкретни константи като аргументи - обикновено само 1-ци/само 0-ли) всички функции от даденото множество се държат еднакво (връщат една и съща стойност), то множеството не е пълно. Това е така, защото не всички функции във $\mathcal{F}$ дават еднаква стойност при конкретен вход. При този тип доказателство предикатът по-горе обикновено има горе-долу следният вид: $P(g)$:= при вход само 1-ци g връща 1.
Пример: нека $G = \{\wedge, \vee\}$, не е трудно да се види, че всички функции от $[G]$ при вход $\overrightarrow{x} = (1, \cdots, 1)$ имат стойност 1. Ясно е обаче, че *не* всички функции в $\mathcal{F}$ изпълняват това, например едноаргументната функция $neg : neg(1) = 0$ го нарушава, така че $[G] \neq \mathcal{F}$. ✓
Ако искаме да сме по-педантични, частта с "*не е трудно да се види, че всички ф-ии от* $[G]$..." трябва да се докаже по индукция (структурна).

2. Показване, че отрицанието neg не е част от множеството: Тук ще опитаме да докажем отсъствието на някоя функция от $[G]$ (в частност отрицанието) малко по-прецизно, по индукция.

За целта ще разгледаме конкретен пример: искаме да докажем, че $G = \{\wedge, \vee\}$ не е пълно. Функцията neg е едноаргументна, така че можем да считаме, че съществува булева формула/израз e, в която участва само една променлива x и чиято семантика е именно $neg(x)$. Ограничаваме се до азбуката $\Sigma = \{\vee, \wedge, x, (,)\}$. Ще правим структурна индукция по построение на булевите формули, така че е добре да имаме предвид тяхната индуктивната дефиниция. Нека E е множеството (езикът) от коректните булеви изрази, конструирани само с горните. Базата е $x \in E$. Ако $e_1, e_2 \in E$, то $(e_1 \vee e_2), (e_1 \wedge e_2) \in E$ (стъпка).
Тези формули/изрази обаче са само синтактични обекти, нашата цел обаче е да докажем, че тяхната семантика е различна от тази на $neg(x)$. Следва същинското доказателство по индукция:
База: Булевият израз x има семантика стойността на променливата x.
ИП: Нека $e_1, e_2 \in E$ са два булеви израза, получени чрез суперпозиция (композиция) от променливата x и $\vee, \wedge$ и имат семантика.
ИС: Тогава изразите $(e_1 \vee e_2), (e_1 \wedge e_2)$ имат семантика съответно $x \vee x = x, x \wedge x = x$. Това показва, че всички едноаргументни функции от $[G]$ имат семантика x, но не и $\overline{x}$, множеството не е пълно. ✓

3. Доказателство с монотонност:

Дефиниция. (*монотонна функция*) Булева функция $f : \{0,1\}^n \to \{0,1\}$ е монотонна, ако $\forall i \leq n \, \forall x_i, y_i \in \{0,1\}, x_i \leq y_i : f(x_1, \cdots, x_n) \leq f(y_1, \cdots, y_n)$, или по-просто $\forall x, y \in \{0,1\}^n, x \leq y : f(x) \leq f(y)$, като сравнението на булеви вектори е почленно.

Пример: Функциите $\wedge, \vee$ са монотонни, а $\neg, \to$ не са.

Важно за нас е следното свойство:

4: Лема

Композиция на монотонни функции е монотонна функция.

Доказателство. Нека $f : \{0,1\}^n \to \{0,1\}, g_i : \{0,1\}^m \to \{0,1\}$ за $i \leq n$ са монотонни функции. Разглеждаме композицията $h(x_1, \cdots x_m) = f(g_1(x_1, \cdots, x_m), \cdots, g_n(x_1, ..., x_m))$. Нека $x', x'' \in \{0,1\}^m, x' \leq x''$. От монотонността на $g_i : \forall i \leq n : g_i(x') \leq g_i(x'')$, откъдето пък $h(x') = f(g_1(x'), \cdots, g_n(x')) \leq f(g_1(x''), \cdots, g_n(x'')) = h(x'')$, като вземем предвид и монотонността на f. Следва, че $h(x)$ също е монотонна. ■

Забележка. Тук ползвахме общо приетата форма на композиция $h(x) := f(g_1(x), \cdots, g_n(x)), x \in \{0,1\}^m$. Визуално това може да има различия с композирането в лекционните записки, но имат еднаква изразителна мощ (функциите, които генерират са еднакви).

Самата идея за подхода с монотонност не е сложна. - Показваме, че функциите от $[G]$ са монотонни (например ползвайки свойството, че композицията запазва монотонност, като може това да се придружи с индукция). Не всички функции във $\mathcal{F}$ обаче са монотонни, както стана ясно от примера. Тогава G не е пълно. ✓

Още малко за пълнота: Горните подходи на пръв поглед разглеждат частни случаи и са "ad hoc". Донякъде да, но оказва се, че те са в основата на следващото *необходимо и достатъчно условие* за пълнота на множество функции. Преди това обаче една дефиниция.

Дефиниция (*Post function classes*)**.** Следните 5 множества наричаме *класове на Пост*:

- множеството от функции, запазващи 0: $T_0 := \{f \in \mathcal{F} | \, f(0, , \cdots, 0) = 0\}$;
- множеството от функции, запазващи 1: $T_1 := \{f \in \mathcal{F} | \, f(1, , \cdots, 1) = 1\}$;
- множеството от монотонните функции: $M := \{f \in \mathcal{F} | \, f$ е монотонна $\}$;
- множеството от самодвойствените функции: $S := \{f \in \mathcal{F} | \, f(\overline{x}) = \overline{f(x)}, x \in \{0,1\}^n\}$;
- множеството от линейните функции: $L := \{f \in \mathcal{F} | \, f$ е изразима само чрез $+$ (XOR) и константи$\}$, например $x + y + z$;

5: Post's Functional Completeness Theorem

Множество G *не* е пълно тстк е подмножество на поне един от петте класа T_0, T_1, M, S, L.

За упражнение върху (не)пълнота разгледайте *задача 9*.

Кодове на Грей*

В определени случаи би било удобно да можем да подредим двоични числа (или двоични низове) в редица така, че всеки две последователни от тях да се различават в точно един бит (броят битове, в които два низа/вектора се различават се нарича hamming distance). Още по-хубаво ще е, ако тази наредба има циклично свойство, т.е. първото и последното двоично число/низ също имат точно един бит разлика.
Забележете, че в обикновената наредба на естествените числа такава зависимост няма, например 5 ($=101_{(2)}$) и 6 ($=110_{(2)}$) са последователни числа, но техните представяния се различават в два бита.
Именно такава наредба представлява *кодът на Грей* (още наричан *binary reflected code*). Той намира приложение в генетични алгоритми, предотвратяване на грешки, преобразуване на аналогов към цифров сигнал и др.

- **Binary to Gray:** k-тото число от тази наредба образуваме, като разгледаме двоичното представяне на k (например $k = \underline{1}1101..._{(2)}$). Старшият бит на кода остава същият като този на k (в случая старшият бит е 1). Всеки следващ i-ти (за $i \geq 2$) бит на кода определяме като XOR-нем $(i-1)$-вия с i-тия бит на k. Казано по-просто, $code_k = k \oplus (k >> 1)$, където $>> 1$ е десен shift (с една позиция надясно).

- **Gray to Binary:** Налага се обаче и обратното, по даден код $b \in \{0,1\}^n$ на Грей да разберем на кое двоично число $a \in \{0,1\}^n$ съответства той. Определяме по следното рекурентно правило: $a_1 = b_1, a_i = b_i \oplus a_{i-1}$ за $i \geq 2$.

Пример: Нека $n = 4$, искаме да подредим (циклично) двоичните низове с дължина 4 по такъв начин, че всеки да съседни да имат разлика в точно един бит.
- Нулевото число в тази наредба намираме, като превърнем $0 = 0000_{(2)}$ в код на Грей по гореописания начин: $0 \oplus (0 >> 1) = 0$.
- Така например 12-тото число (започваме броенето от 0) в тази последователност ще е получим като $6 \oplus (6 >> 1)$, или

$$\begin{array}{r} 1100 \\ \oplus 110 \\ \hline 1010 \end{array}$$

В таблицата вдясно са дадени част от стойностите. Търсената последователност е: $0000 \rightsquigarrow 000\mathbf{1} \rightsquigarrow 00\mathbf{1}1 \rightsquigarrow 001\mathbf{0} \rightsquigarrow 0\mathbf{1}10 \rightsquigarrow 011\mathbf{1} \rightsquigarrow \cdots \rightsquigarrow 1001 \rightsquigarrow 100\mathbf{0} \circlearrowright$
Обратно, по $b = 1010$ можем да намерим съответстващото двоично число $a \in \{0,1\}^n$ като $a_1 = b_1 = 1, a_2 = b_2 \oplus a_1 = 1, a_3 = b_3 \oplus a_2 = 0, a_4 = b_4 \oplus a_3 = 0 \Rightarrow a = 1100$ ✓

Binary	Gray
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
...	...
1110	1001
1111	1000

Всъщност кодът на Грей задава биекция $g : \{0,1\}^n \to \{0,1\}^n, g(\overline{k}) = \overline{k \oplus (k >> 1)}$ със свойството, че $g(\overline{k})$ и $g(\overline{k+1})$ се различават в точно един бит (с $\overline{k}$ тук бележим двоичното представяне на числото $k, 0 \leq k < 2^n$).

Коректност: Доказателството, че така генерираният код има исканите свойства, е добра задача за упражнения, но тук няма да се спираме в детайл върху нея. Основно трябва да бъдат доказани две неща - че $g(k)$ е биекция и че $g(\overline{k}), g(\overline{k+1})$ за $k < 2^n - 1$ се различават в точно един бит (а също и $g(\overline{0}), g(\overline{2^n - 1})$). Втората част излиза относително лесно, като се има предвид, че двоичните представяния на $k, k+1$ имат общия вид $k = \alpha 0 \underbrace{1 \cdots 1}_{\geq 0}{}_{(2)}$ и $k+1 = \alpha 1 \underbrace{0 \cdots 0}_{\geq 0}{}_{(2)}$, като двоичният низ α е техен общ префикс. Оттук не е трудно да се види, че прилагането на g (тоест операцията binary to gray) върху тези двоични числа $\overline{k}, \overline{k+1}$ ще даде двоични числа, различаващи се в точно 1 бит, което и очакваме. ✓

Приложение: Интересно е, че кодът на Грей може да намери приложение при решаване на задачата за Ханойските кули. *виж повече*

Трансформация на Цейтин*

Karanaugh map*

Quine–McCluskey algorithm*

Схеми от функционални елементи

Задачи

Задача 1. Еквивалентни ли са следните двойки булеви изрази:

- *(!)* $\overline{x}$ и $x|x$
- $A \wedge B$ и $(A|B)|(A|B)$
- *(!)* $A(B+C)$ и $AB + AC$
- $A(A+B)$ и $A + AB$
- $\overline{x}(y \oplus z)$ и $\overline{x \vee ((y \to z)(z \to y))}$

Решение.

- $x|x = \overline{x.x} = \overline{x}$ $\blacksquare$
- $(A|B)|(A|B) = \overline{AB} \mid \overline{AB} = \overline{\overline{AB} \wedge \overline{AB}} = AB \vee AB = AB = A \wedge B$ $\blacksquare$
- $A(B+C) = A(B\overline{C} \vee C\overline{B}) = AB\overline{C} \vee AC\overline{B}$ и $AB + AC = AB.\overline{AC} \vee \overline{AB}.AC = AB(\overline{A} \vee C) \vee AC(\overline{A} \vee \overline{B}) = AB\overline{C} \vee AC\overline{B}$ $\blacksquare$
- $A(A+B) = A(\overline{A}B \vee A\overline{B}) = A\overline{A}B \vee AA\overline{B} = 0 \vee A\overline{B} = A\overline{B}$, също и $A + AB = \overline{A}AB \vee A\overline{AB} = A\overline{AB} = A(\overline{A} \vee \overline{B}) = A\overline{B}$ $\blacksquare$
- $\overline{x}(y \oplus z) = \overline{x}.\overline{y \leftrightarrow z} = \overline{x \vee (y \leftrightarrow z)} = \overline{x \vee ((y \to z) \wedge (z \to y))}$ $\blacksquare$

Задача 2. Колко са всички тотални булеви функции над n променливи?

Решение. Тоталните булеви функции са от вида $f : \{1,0\}^n \to \{1,0\}$. Домейнът е 2^n-елементен, а кодомейнът е двуелементен, значи търсеният брой е 2^{2^n} (на всяка стойност от домейна могат да съпоставени две стойности). ■

Задача 3. Намерете $|F|$, ако:

- $F := \{f \in \mathcal{F}^n | \ \forall \mathbf{x} \in \{0,1\}^n : f(\mathbf{x}) = \overline{f(\overline{\mathbf{x}})}\}$
- $F := \{f \in \mathcal{F}^n | \ \forall \mathbf{x} \in \{0,1\}^n : f(\mathbf{x}) = f(\overline{\mathbf{x}})\}$

Дефиниция (*симетрична функция*). Функция на n променливи е симетрична тстк стойността ѝ е една и съща за всяка пермутация на променливите ѝ.

Задача 4. Да се намери броят на симетричните булеви функции на n променливи.

Решение. Да започнем с пример за онагледяване, нека $f(x, y, x)$ е симетрична, тогава $f(0,0,1) = f(0,1,0) = f(1,0,0)$. Аналогично $f(0,1,1) = f(1,0,1) = f(1,1,0)$. Забелязваме, че необходимо и достатъчно условие, за да бъде дадена функция с n аргумента симетрична, е $\forall 0 \leq k \leq n$: за всеки вход с точно k единици, функцията да дава една и съща стойност. Имаме $n+1$ различни k-та, за всяко можем имаме свобода в определяне каква стойност да дава функцията при вход с точно k единици, "изборите" са независими, тоест общо 2^{n+1} ■

Задача 5. Да се намери броят на булеви функции на $n \geq 2$ променливи, които запазват стойността си при размяна на променливите x_1 и x_2.

Задача 6. Да се намери броят на булеви функции на n променливи, които приемат стойност 1 върху поне една двойка противоположни входни вектори.

Дефиниция. (*фиктивна променлива*) Променлива x_i е *фиктивна* за функция $f \in \mathcal{F}^n$, ако $\forall x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n \in \{0,1\} : f(x_1, \cdots, x_{i-1}, 0, x_{i+1}, \cdots, x_n) = f(x_1, \cdots, x_{i-1}, 1, x_{i+1}, \cdots, x_n)$.

Задача 7. Да се намери броят на булеви функции на n променливи, които нямат фиктивни променливи.

Задача 8. Дадени са булевите функции $f = 1010, g = 1100$, да се намери каноничното представяне на функцията $h(x_5, x_3, x_1) = f(x_3, g(x_1, x_5))$

Решение. Можем да преименуваме променливите $y_1 = x_5, y_2 = x_3, y_3 = x_1 \Rightarrow h(y_1, y_2, y_3) = f(y_2, g(y_3, y_1))$. Така например откриваме $h(0,0,1) = f(0, g(1,0)) = f(0,1) = 0$, останалите стойности се намират аналогично. ■

Задача 9. (*) Докажете или опровергайте, че следните множества от булеви функции са пълни:

- $\{\wedge, neg\}$
- $\{\wedge, \oplus, \mathbf{1}\}$
- $\{\downarrow\}$
- $\{|\}$
- $\{f, g\}$, ако $f(x, y) = 0110, g(x, y) = 1101$
- $\{\to, \mathbf{0}\}$

- $\{\to, \mathbf{1}\}$
- $\{\wedge, \oplus\}$
- $\{\wedge, \vee\}$
- $\{\wedge, \vee, \to\}$

Решение. Ще използваме *Теорема 1*:

- Пълно е. Следва от факта, че $F = \{\wedge, \vee, neg\}$ е пълно и всеки от трите му елемента може да бъде изразен посредством функциите в $G = \{\wedge, neg\}$, по-конкретно $x \vee y = \overline{\overline{x} \wedge \overline{y}}$. ■

- Отново е пълно, $x \oplus 1 = \overline{x}$, с което сведохме до горното множество, което е пълно. ■

- Пълно е: $x \downarrow x = \overline{x \vee x} = \overline{x}$, а също $x \vee y = \overline{x \downarrow y} = (x \downarrow y) \downarrow (x \downarrow y)$, с което изведохме функциите негация и дизюнкция, които са пълно множество. ■

- Пълно е: $x|x = \overline{x \wedge x} = \overline{x}$, а също $x \wedge y = \overline{x|y} = (x|y)|(x|y)$, с което изведохме функциите негация и конюнкция, които са пълно множество. ■

- Макар че можем да се справим и без това, тук наблюдение, че f е $\oplus$, а g е $\to$, би улеснило нещата. Ще докажем, че $\{\oplus, \to\}$ е пълно: $x \to x = 1 \Rightarrow \overline{x} = 1 \oplus x = (x \to x) \oplus x$, а $x \vee y = (x \to y) \to y$. ■

- Множеството е пълно: $x \to \mathbf{0} = \overline{x}$, $(x \to y) \to y = x \vee y$, като така сведохме до пълното множество $\{\neg, \vee\}$. ■

- Оказва се, че $\{\to, \mathbf{1}\}$ не е пълно. Лесно се вижда, че произволна булева функция, построена само с помощта на горните има стойност 1 при вход 1-ци. С други думи, функциите от $[\{\to, \mathbf{1}\}]$ *запазват 1*, тоест това не са всички функции. ■

- Множеството $\{\wedge, \oplus\}$ не е пълно. Причината е, че $[\{\wedge, \oplus\}] \subseteq T_0$ (класа на Пост с функции, запазващи 0 /0-preserving functions/), тоест при вход само 0-ли, функциите от затварянето връщат винаги 0. ■

- Множеството $\{\wedge, \vee\}$ също не е пълно. Тук можем да ползваме, че функциите $\wedge, \vee$ са монотонни, както са и всички техни производни, образувани при композиции. Следователно $[\{\wedge, \vee\}] \subseteq M$ съдържа само монотонни функции, но разбира се, не всички във $\mathcal{F}$ са такива. ■

- Ще докажем, че негацията не може да се представи само чрез горните. Подобно доказателство вече сме правили - става чрез индукция по построение на булевите изрази.
  База: Променливата x има стойност x.
  ИП: Разглеждаме булеви изрази e_1, e_2, построени само с една променлива x и трите операции $\wedge, \vee, \to$, които имат семантика стойността на x или $\mathbf{1}$.
  ИС: Разглеждаме произволен сложен булев израз, записан само чрез горните. Нека операцията с най-нисък приоритет е $op \in \{\wedge, \vee, \to\}$ (на върха на дървото на израз), тя е измежду трите операции по-горе, така че е двуместна. Тоест имаме нещо от сорта на e_1 op e_2. По ИП e_1 и e_2 имат семантика стойността на x или $\mathbf{1}$. Вижда се, че при прилагане на op стойността остава това остава в сила (на практика разглеждаме $x \vee x,\ x \vee 1,\ 1 \vee 1,\ x \wedge x,\ x \wedge 1,\ 1 \wedge 1,\ x \to x,\ x \to 1,\ 1 \to x,\ 1 \to 1$). Тоест не е възможно да построим формула за функцията $\overline{x}$. ■

Задача 10. Да се намерят СДНФ, СКНФ и полиномът на Жегалкин за следните булеви функции:

- $f = 11011101$;
- $f(x, y, z) = x \vee y \to z$
- $f(x, y, z) = (\overline{x} \vee z) \wedge (x \oplus y)$
- $f = x_1 \oplus (\overline{x}_2 \wedge x_1)$
- $f(g(x, y), g(y, x))$, ако $f(x, y) = 0110, g(x, y) = 1101$

Задача 11. Да се намери булева формула, ползваща само негация, конюнкция, дизюнцкия за следните:

- за $f = 0000$ (константа 0);
- за $f = 1111$ (константа 1);
- за $f = 00110101$;
- за $f = 10010110$;

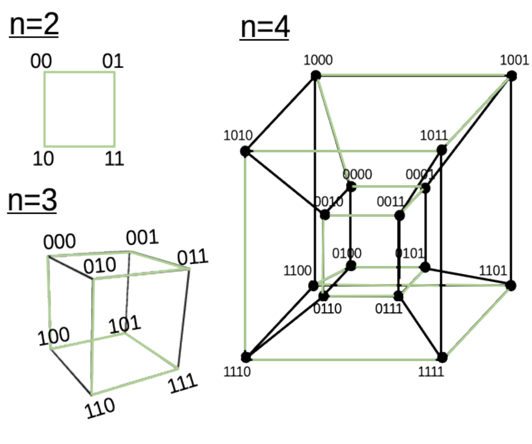
Задача 12. Верни ли са равенствата по-долу? Дайте кратка аргументация защо.

- $w\overline{x}yz \vee w\overline{xyz} \vee \overline{w}x\overline{y}z \vee \overline{w}xy\overline{z} \vee wxyz = w\overline{x}yz + w\overline{xyz} + \overline{w}x\overline{y}z + \overline{w}xy\overline{z} + wxyz$?
- $w\overline{x}y \vee w\overline{xyz} \vee \overline{w}x\overline{y}z \vee \overline{w}xy\overline{z} \vee wyz = w\overline{x}y + w\overline{xyz} + \overline{w}x\overline{y}z + \overline{w}xy\overline{z} + wyz$

Решение. Еквивалентни преобразувания и разглеждане на таблици на истинност не биха били удачни решения на тази задача, защото удължават решението чрезмерно. Освен това към примерите могат да бъдат добавени още променливи и клаузи, които да не променят идеята, но пък биха затормозили допълнително горните подходи.

- Да, равенството е вярно. Аргументацията е същата като тази, която ползвахме в доказателството на коректността на алгоритъма за образуване на полином Жегалкин от СДНФ. Важното тук е, че конюнктивните клаузи са пълни. ■
- Не всички конюнктивни клаузи са пълни. Това ни навежда на мисълта, че няма равенство. Наистина, при $w = y = z = 1, x = 0$ LHS има стойност 1, а RHS има стойност 0, така че двете формули не са еквивалентни. ■

Задача 13. Върховете на n-мерен хиперкуб са n-мерните двоични вектори (общо 2^n). Два върха са съседни (има ръб на куба, който ги свързва), ако съответните им двоични вектори се различават в точно един бит. Докажете, че графът, получен от върховете и ръбовете на хиперкуба, е Хамилтонов и предложете алгоритъм, който да намира Хамилтонов цикъл (на фигурата в зелено).



Решение. Всъщност n-битовият код на Грей реализира именно такъв Хамилтонов цикъл. Започваме от върха $\underbrace{0\cdots0}_{n}$ (който съответства на двоичното $0 = 0\cdots0_{(2)}$), продължаваме с връх $\underbrace{0\cdots01}_{n}$ (който съответства на двоичното $1 = 0\cdots01_{(1)}$), после с връх $\underbrace{0\cdots011}_{n}$ (който съответства на двоичното $2 = 0\cdots010_{(1)}$) и т.н., k-тия връх от цикъла е k-тото n-битово число от кода на Грей. ■