Булеви функции и формули.

Минко Марков

9 декември 2025 г.

Съдържание

1 Булеви вектори

Както знаем, елементите на $\{0,1\}^n$ са наредените n-орки от нули и единици. Тук ги наричаме "булевите вектори с дължина n" или накратко "n-векторите". За краткост и прегледност ще записваме n-векторите без скоби и запетаи-разделители; примерно, 010110, а не $(0,1,0,1,1,0)$.

Ползваме удобната конвенция имената на векторите да бъдат записвани с удебелени букви (в полиграфията се казва "получерен шрифт", на английски е *boldface*), например $\mathbf{b}$. Ако сме дефинирали някакво име на n-вектор, да кажем $\mathbf{b}$, то неговите елементи именуваме със същото име, само че тях изписваме с нормално дебели букви (*regular face*), ето така: $\mathbf{b} = b_1 b_2 \cdots b_n$, където $b_i \in \{0, 1\}$ за $1 \leqslant i \leqslant n$.

Нека $x \in \{0, 1\}$. *Обратната стойност на* x, която бележим с "$\overline{x}$", се дефинира така:

$$\overline{x} \stackrel{\text{деф}}{=} \begin{cases} 1, & \text{ако } x = 0 \\ 0, & \text{ако } x = 1 \end{cases}$$

Противоположни вектори са вектори с една и съща дължина, които се различават във всеки елемент; ако $\mathbf{a}$ и $\mathbf{b}$ са n-вектори, те са противоположни тстк

$$\forall i \in \{1, 2, \ldots, n\} : a_i = \overline{b_i}$$

Фактът, че $\mathbf{a}$ и $\mathbf{b}$ са противоположни, записваме накратко така: $\mathbf{a} = \overline{\mathbf{b}}$. Добре известно е, че $\overline{\overline{\mathbf{a}}} = \mathbf{a}$, така че $\mathbf{a} = \overline{\mathbf{b}} \leftrightarrow \overline{\mathbf{a}} = \overline{\overline{\mathbf{b}}} = \mathbf{b}$.

Нулев вектор е всеки вектор, който се състои само от нули $00 \cdots 00$. Ако дължината не е съществена за дискурса, може да казваме членувано *нулевия̀т вектор*. Нулевият вектор ще записваме с "$\mathbf{0}$". Напълно аналогично, въвеждаме *единичния вектор*, който записваме с "$\mathbf{1}$".

2 Дефиниция на "булева функция"

Определение 1. *Нека* $n \in \mathbb{N}$. Булева функция с n аргумента *е всяка функция от* $\{0, 1\}^n$ *в* $\{0, 1\}$. Булева функция *е всяка булева функция с* n *аргумента, за някое* $n \in \mathbb{N}$.

Неформално, булева функция с n аргумента е "раздаване" на нули или единици на n-векторите. Формално, булева функция с n аргумента е множество от наредени двойки

$$\{(x, y) \mid x \in \{0, 1\}^n, y \in \{0, 1\}\}$$

такова че всеки n-вектор се среща точно веднъж като първи елемент на наредена двойка. Последното влече, че мощността на това множество е 2^n. Примерно, конюнкцията е булева функция с 2 аргумента, а именно $\{(00, 0), (01, 0), (10, 0), (11, 1)\}$.

Множеството от всички булеви функции с n аргумента ще бележим с " $\mathcal{F}^n$ ". Съгласно изучаваното в раздела Комбинаторика, $|\mathcal{F}^n| = 2^{2^n}$. Примерно, булевите функции с 1 аргумент са 4 на брой, тези с 2 аргумента са 16 на брой, тези с 3 аргумента са 256 на брой, тези с 4 аргумента са 65 536 на брой, и така нататък.

Забележете, че n е естествено число, така че може да имаме 0 аргумента, като функциите с 0 аргумента са 2 на брой. Наистина, 0-кратното декартово произведение е $\{()\}$, следователно домейнът е едноелементен и има точно две булеви функции с 0 аргумента, които отъждествяваме с двете булеви константи 0 и 1.

Множеството $\mathcal{F}$ от всички булеви функции е изброимо безкрайно и се бележи с "$\mathcal{F}$". Очевидно

$$\mathcal{F} = \bigcup_{n \in \mathbb{N}} \mathcal{F}^n$$

3 Булевите функции като вектори

По дефиниция, всяка булева функция е множество от наредени двойки. Ето пример за булева функция с 3 аргумента:

$$f = \{((0,0,0),0),((0,0,1),1),((0,1,0),1),((0,1,1),0)$$
$$((1,0,0),1),((1,0,1),1),((1,1,0),0),((1,1,1),1)\}$$

Можем да опишем тази функция много по-прегледно с таблица:

			f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Но може да запишем функцията още по-компактно. Оттук насетне приемаме, че n-векторите винаги са подредени отгоре надолу лексикографски, точно както са подредени 3-векторите в тази таблица. Тогава е излишно да ги пишем – знаейки броя на аргументите, ние знаем на кой ред в таблицата, кой е n-векторът.

Ако не пишем векторите, таблицата става със само една колона. В нашия пример:

f
0
1
1
0
1
1
0
1

За да пестим място при писането, записваме функцията не като колона, а хоризонтално:

$$f = 01101101$$

Такова представяне на булева функция ще наричаме *канонично*. Каноничното представяне на булева функция с n аргумента е вектор от 2^n на брой булеви стойности.

Забележете, че дължината на вектора трябва да е точна степен на двойката, за да имаме право да кажем, че този вектор представлява булева функция. И така, ако е даден булев a вектор с дължина m, то

- ако m е точна степен на двойката, то a представлява булевата функция f (тя е една единствена) с $\log_2 m$ аргумента, дефинирана така: за всяко $i \in \{0,1,\ldots,m-1\}$, ако b е записът на i в двоична позиционна бройна система с дължина точно $\log_2 m$, с евентуално попълване с нули отпред, то $f(b) = a_i$;
- ако m не е точна степен на двойката, то a не представлява запис на булева функция.

Като пример, ако поначало беше даден векторът $\mathbf{a} = 01101101$, веднага виждаме, че дължината му е 8 и това е точна степен на двойката, така че той представлява булевата функция f с 3 аргумента, такава че $f(000) = 0$, $f(001) = 1$, $f(010) = 1$, и така нататък. Въпреки че, строго формално, функцията не е вектор, а множество от наредени двойки, имаме право да я отъждествим с вектора, защото той я определя напълно при споменатите имплицитни допускания.

Като друг пример, векторът 010110 не представлява булева функция, защото дължината му не е точна степен на двойката.

4 Аргументи и променливи

4.1 Променливите са имена на аргументите

Както знаем, "променлива" е величина, която може да приема като стойност кой да е елемент на някакво множество. Неформално, това е кутия, в която можем да сложим кой да е елемент на множеството. Когато говорим за булева функция f с n аргумента, можем да ползваме израза "$f(\mathbf{x})$", където $\mathbf{x}$ е променливата на функцията. Променливата $\mathbf{x}$ взема стойности от множеството $\{0,1\}^n$. Това множество има структура – то е Декартово произведение. Това ни дава право да ползваме израза "$f(x_1, x_2, \ldots, x_n)$", където x_1, x_2, $\ldots$, x_n са променливи, вземащи стойности от множеството $\{0,1\}$. Те са две по две различни променливи, въпреки че вземат стойности от едно и също множество. Иначе казано, в Декартовото произведение

$$\underbrace{\{0,1\} \times \{0,1\} \times \cdots \times \{0,1\}}_{n \text{ множители}}$$

асоциираме всеки множител с отделна променлива.

От казаното досега може би изглежда, че "променливи на булева функция" и "аргументи на булева функция" са синоними. Но това не е така. Променливите са имена на аргументите. Като пример, нека пак да разгледаме функцията на 3 аргумента

$$f = 01101101$$

Ако използваме записа "$f(x_1, x_2, x_3)$", ние казваме, че първият аргумент се казва x_1, вторият се казва x_2, а третият се казва x_3. Но можеше да напишем "$f(x_2, x_1, x_3)$". Сега наричаме първия аргумент x_2, втория, x_1, а третия, x_3. Какви имена ще ползваме, няма значение за функцията. Функцията си остава същата, каквито и имена да използваме за нейните променливи, стига тези имена да са две по две различни. В примера с f, можеше да е "$f(x, y, z)$".

Ако има съвпадение на имена на променливи, имаме частен случай, който ще разгледаме надолу.

4.2 Размяна на имена на аргументи може да промени функцията.

Ако **първо** сме дефинирали, че

$$f(x_1, x_2, x_3) = 01101101 \tag{1}$$

и **след това** в същия контекст ползваме записа "$f(x_2, x_1, x_3)$", вече говорим за друга функция! Да видим защо. Таблица 1 е таблицата на $f(x_1, x_2, x_3)$.

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Таблица 1: $f(x_1, x_2, x_3)$.

Да разгледаме израза "$f(x_2, x_1, x_3)$". В него имената "x_1" и "x_2" са разменени по отношение на (1). Таблица 2 е таблицата на $f(x_2, x_1, x_3)$. Тя се получава от Таблица 1 чрез размяна на първата и втората колона, а **не** чрез просто преименуване на колоните.

x_2	x_1	x_3	$f(x_2, x_1, x_3)$
0	0	0	0
0	0	1	1
1	0	0	1
1	0	1	1
0	1	0	1
0	1	1	0
1	1	0	0
1	1	1	1

Таблица 2: $f(x_2, x_1, x_3)$.

Ясно се вижда, че това е друга функция – каноничното ѝ описание е 01111001, докато началната функция беше 01101101. Забележете, че и в двете таблици, върху един и същи вектор функцията има една и съща стойност; примерно, $f(000) = 0$, $f(100) = 1$, и така нататък. Това е очевидно – стойностите на функцията върху векторите са нейната същностна характеристика. Разменяйки променливите x_1 и x_2 спрямо първоначално дефинираната $f(x_1, x_2, x_3)$ (1), ние променяме наредбата на векторите[†]. Променяйки наредбата на векторите, ние пермутираме елементите на колоната f в таблицата, при което може да получим друга функция, както в примера.

Прочее, свойството на булева функция да не се променя при пермутация на променливите си, се нарича *симетричност*. Вижте следната подсекция.

4.3 Симетричност на (булева) функция

За симетричност може да говорим не само при булевите функции. Функция на няколко променливи $f(x_1, x_2, \ldots, x_n)$, такава че променливите вземат стойности от едно и също множество, се нарича *симетрична*, ако стойността на функцията се запазва при всяка пермутация

[†]Наистина, в Таблица 2 векторите не са наредени лексикографски отгоре надолу.

на променливите. С други думи,

$$f(x_1, x_2, \ldots, x_n) = f(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)})$$

за всяка пермутация $\pi(1), \pi(2), \ldots, \pi(n)$ на вектора $1, 2, \ldots, n$. Например, ако $n = 3$, то:

$$f(x_1, x_2, x_3) = f(x_1, x_3, x_2) = f(x_2, x_1, x_3) = f(x_2, x_3, x_1) = f(x_3, x_1, x_2) = f(x_3, x_2, x_1)$$

Ако пък $n = 2$, то $f(x_1, x_2) = f(x_2, x_1)$, така че комутативността е частен случай на симетричността.

В частност, дефиниция е в сила за булевите функции, защото, ако $f(x_1, x_2, \ldots, x_n)$ е булева функция, нейните променливи наистина вземат стойности от едно и също множество, а именно $\{0, 1\}$.

4.4 Унификация на променливи.

За променливи, каквито разгледахме досега, казваме, че са *независими*, защото вземат стойностите си независимо една от друга. Да разгледаме възможността някои от тях да са *зависими* и по-точно стойността на едната да е точно равна на стойността на другата. Това се нарича *унификация* на променливи и се отбелязва чрез еднакви имена[†]. Нека пак ползваме като пример $f(x_1, x_2, x_3) = 01101101$. Да разгледаме функцията $f(x_1, x_1, x_3)$. Тя се получава от $f(x_1, x_2, x_3)$ чрез унификация на първите две променливи. Забележете, че $f(x_1, x_1, x_3)$ е функция на **две променливи** и с **три аргумента**. И така, при унификация на променливи, броят на променливите намалява, но броят на аргументите не намалява.

Да осмислим унификацията на променливи с помощта на табличното представяне. Таблица 3 е таблицата на $f(x_1, x_1, x_3)$. Тя се получава от Таблица 1 чрез премахване на всички редове (те са четири на брой), в които векторите имат различна първа и втора стойност.

x_1	x_1	x_3	$f(x_1, x_1, x_3)$
0	0	0	0
0	0	1	1
~~0~~	~~1~~	~~0~~	~~1~~
~~0~~	~~1~~	~~1~~	~~0~~
~~1~~	~~0~~	~~0~~	~~1~~
~~1~~	~~0~~	~~1~~	~~1~~
1	1	0	0
1	1	1	1

x_1	x_1	x_3	$f(x_1, x_1, x_3)$
0	0	0	0
0	0	1	1
1	1	0	0
1	1	1	1

Таблица 3: $f(x_1, x_1, x_3)$.

[†]Ако трябва да сме педантични, унификацията е на аргументи, понеже променливите са имената на аргументите.

Всяка унификация намалява дължината на функцията наполовина. В случая, $f(x_1, x_1, x_2) = 0101$ е с дължина 4. В екстремния вариант, унификация на всички променливи води до функция с дължина само 2, независимо от броя на аргументите; това е напълно очаквано, тъй като при унификация на всички променливи остава само една променлива, която има точно две възможни стойности.

Заслужава да се акцентира, че броят на аргументите не намалява при унификация на променливи, така че ако началната функция е с n аргумента, след унификация на променливи функцията пак е с домейн n-вектори, но вече не всички n-вектори, а само тези, които имат едни и същи стойности на дадени позиции. В примера, домейнът на $f(x_1, x_1, x_3)$ е $\{000, 001, 110, 111\}$.

Унификацията на променливи може да бъде описана формално чрез рестрикция на функцията до подмножество на домейна.

Определение 2. *Нека* $f(x_1, x_2, \ldots, x_n)$ *е булева функция. Нека* $1 \leqslant i < j \leqslant n$. *Нека*

$$X = \{a \in \{0,1\}^n \mid a_i = a_j\}$$

Булевата функция, която се получава от $f(x_1, x_2, \ldots, x_n)$ чрез унификация на x_i и x_j, *е* $f\big|_X$.

Записът "$f(x_1, x_2, \ldots, x_n)$", който често ползваме, предполага, че унификация на променливи няма, така че променливите са на брой колкото аргументите, а именно n.

4.5 Фиктивни и съществени променливи.

Неформално, променлива е фиктивна, ако нейното "присъствие" не се отразява на функционалните стойности.

Определение 3. *Нека* $f(x_1, x_2, \ldots, x_n)$ *е булева функция. Нека* $1 \leqslant i \leqslant n$. *Променливата* x_i *се нарича* фиктивна, *ако*

$$f(x_1, x_2, \ldots, x_{i-1}, 0, x_{i+1}, x_{i+2}, \ldots, x_n) = f(x_1, x_2, \ldots, x_{i-1}, 1, x_{i+1}, x_{i+2}, \ldots, x_n)$$

за всяка стойност на $(n-1)$*-вектора* $x_1 x_2 \cdots x_{i-1} x_{i+1} x_{i+2} \cdots x_n$. *Променлива, която не е фиктивна, се нарича* съществена.

Всяка функция, която има фиктивна променлива, има някаква повтаряемост: грубо казано, половината от функционалните стойности са същите като стойностите от другата половина. Таблица 4 показва три функции g, h и u на три променливи x_1, x_2 и x_3, като x_1 е фиктивна в g, x_2 е фиктивна в h и x_3 е фиктивна в u. Ясно се вижда, че

- x_1 е фиктивна тстк първата половина е равна на втората половина,
- x_2 е фиктивна тстк първата четвъртина е равна на втората четвъртина и третата четвъртина е равна на четвъртата четвъртина и
- x_3 е фиктивна тстк първата осмина (тоест, първата стойност) е равна на втората, третата е равна на четвъртата, петата е равна на шестата и седмата е равна на осмата.

Примерите от Таблица 4 показват частни случаи, в които точно една променлива е фиктивна. Може обаче няколко променливи да са фиктивни. В екстремния случай всички променливи

x_1	x_2	x_3	g
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

x_1	x_2	x_3	h
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

x_1	x_2	x_3	u
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Таблица 4: x_1 е фиктивна в g, x_2 е фиктивна в h и x_3 е фиктивна в u.

са фиктивни, като очевидно има точно две функции, в които всички променливи са фиктивни, а именно 0 и 1.

Лесно се показва, че броят на функциите от $\mathcal{F}^n$ без фиктивни променливи е $\sum_{k=0}^{n}(-1)^k\binom{n}{k}2^{2^{n-k}}$. Ще изведем този резултат в една от решените задачи чрез принципа на включването и изключването.

Една заключителна забележка. Наличието на фиктивни променливи е **съвсем различно нещо** от унификация на променливи. Унификацията на променливи намалява дължината на функцията наполовина. Наличието на фиктивни променливи означава, че има някаква повтаряемост във функцията, но нейната дължина не се променя от наличието на фиктивни променливи.

5 Булевите функции на един и два аргумента

Както знаем от Секция 2, $|\mathcal{F}^n| = 2^{2^n}$, така че $|\mathcal{F}^1| = 4$. Следва таблица с четирите булеви функции с един аргумент, написани в четири колони вдясно от двата 1-вектора, 0 и 1, които са един под друг вляво. Името на аргумента, тоест, променливата, е x.

x	0	$\mathtt{id}(x)$	$\mathtt{neg}(x)$	1
0	0	0	1	1
1	0	1	0	1

Ето разяснения за четирите функции.

- 0 се нарича *константа нула*. В случая е константа нула с един аргумент, но ние ще ползваме "0" за константа нула на произволен брой аргументи.
- $\mathtt{id}(x)$ е идентитет x. Ясно е защо се казва така: нейните стойности съвпадат с тези на x. На практика често тази функция се записва с "x". Въпреки че по този начин ползваме една и съща буква и за променливата, и за функцията, помним, че те са принципно различни неща.
- $\mathtt{neg}(x)$ е отрицание x. Ясно е защо се казва така: нейните стойности съвпадат с тези на $\overline{x}$. На практика често тази функция се записва с "$\overline{x}$".

- $\mathbf{1}$ се нарича *константа единица*. В случая е константа единица с един аргумент, но, също като при константа нула, ние ще ползваме "$\mathbf{1}$" за константа единица на произволен брой аргументи.

Следва таблица с някои, но не всички, булеви функции с два аргумента. Имената на аргументите, тоест, променливите, са x и y.

x	y	$\mathbf{0}$	$x \wedge y$	$\mathtt{id}(x)$	$\mathtt{id}(y)$	$x \oplus y$	$x \vee y$	$x \downarrow y$	$\mathtt{neg}(y)$	$\mathtt{neg}(x)$	$x \to y$	$x \mid y$	$\mathbf{1}$
0	0	0	0	0	0	0	0	1	1	1	1	1	1
0	1	0	0	0	1	1	1	0	0	1	1	1	1
1	0	0	0	1	0	1	1	0	1	0	0	1	1
1	1	0	1	1	1	0	1	0	0	0	1	0	1

Ето разяснения за тези функции.

- $x \wedge y$ е конюнкция. Допустимо е да се записва с амперсанд "$x\&y$" или просто с конкатенация на променливите "xy".

- $x \oplus y$ е сума по модул две. Това е аналога на изключващото или от съждителната логика. В теорията на булевите функции, сума по модул две има много по-голяма роля, отколкото изключващото или в съждителната логика. Понякога се записва с обикновен плюс "$x + y$", ако няма възможност за двусмислица.

- $\mathtt{neg}(x)$ е отрицание x. Ясно е защо се казва така: нейните стойности съвпадат с тези на $\overline{x}$. На практика често тази функция се записва с "$\overline{x}$".

- $x \vee y$ е дизюнкция. **Не ползвайте** записа "$x + y$" за дизюнкция! "$x + y$" е сума по модул две, а не дизюнкция.

- $x \downarrow y$ се нарича стрелка на Пърс (Peirce). Тази функция е широко известна като NOR, което идва от "negation of or". Недостатък на името "NOR" е, че то предполага композиция на функции; един вид, първо правим OR и после го негираме. Докато буквата "$\downarrow$" подчертава, че става дума за една функция.

- $x \to y$ е импликация. Заслужава да се спомене, че в съждителната логика импликацията има ключово значение, защото е свързана с правенето на изводи, докато в теорията на булевите функции тя няма никакво особено значение.

- $x \mid y$ се нарича черта на Шефер (Sheffer). Тази функция е широко известна като NAND, което идва от "negation of and". Недостатък на името "NAND" е, че то предполага композиция на функции; един вид, първо правим AND и после го негираме. Докато буквата "$\mid$" подчертава, че става дума за една функция.

Свойствата на булевите функции на два аргумента отговарят точно на свойствата на съответните логически съюзи от съждителната логика. Примерно, конюнкцията е комутативна и асоциативна и дизюнкцията също. Конюнкцията дистрибутира спрямо дизюнкцията и обратното. Негацията на негацията на x е същото като x, негацията на дизюнкцията е конюнкцията на негациите, и така нататък.

Заслужават да се споменат няколко ключово важни свойства на сумата по модул две, чиито аналози в съждителната логика не сме разглеждали.

1. $x \oplus y = y \oplus x$. Сумата по модул две е комутатитвна.

2. $x \oplus (y \oplus z) = (x \oplus y) \oplus z$. Сумата по модул две е асоциативна и имаме право да пишем $x \oplus y \oplus z$. Това се обобщава за произволен брой събираеми: записът $x_1 \oplus x_2 \oplus \cdots \oplus x_k$ е недвусмислен; допустимо е накратко да се пише $\bigoplus_{i=1}^{k} x_i$.

3. $x(y \oplus z) = xy \oplus xz$. Конюнкцията е дистрибутивна спрямо сумата по модул две, точно както в училищната математика умножението е дистрибутивно спрямо събирането.

4. $x \oplus (yz) \neq xy \oplus xz$. Сумата по модул две **не е** дистрибутивна спрямо конюнкцията, точно както в училищната математика събирането **не е** дистрибутивно спрямо умножението.

5. $x \oplus 0 = x$. Очевидно винаги е вярно, независимо дали x е 0 или 1.

6. $x \oplus 1 = \mathtt{neg}(x)$. Очевидно винаги е вярно, независимо дали x е 0 или 1.

7. $x \oplus x = 0$. Очевидно винаги е вярно, независимо дали x е 0 или 1.

8. $y \oplus x \oplus x = y$. Следва веднага от 7. и 5.

6 Композиция

Нека $f(x_1, x_2, \ldots, x_i, \ldots, x_n)$ и $g(y_1, y_2, \ldots, y_m)$ са булеви[†] функции. *Композицията на* g *на мястото на* x_i *във* f *е функцията* $f(x_1, x_2, \ldots, x_{i-1}, g(y_1, y_2, \ldots, y_m), x_{i+1}, x_{i+2}, \ldots, x_n)$. Това е функция, която е **различна** (в общия случай) и от f, и от g. Ако се интересуваме от изчисляването на тази функция, алгоритъм за нейното изчисляване може да се получи от алгоритми за изчисляването на f и на g: алгоритъмът за f вика алгоритъма за g.

Новата функция-композиция се дефинира така. Тя е функцията h с $n + m - 1$ аргумента, като за всеки $(n + m - 1)$-вектор $\mathbf{a}$, $h(\mathbf{a}) = f(\mathbf{b})$, където $\mathbf{b}$ е n-векторът, дефиниран така:

$$\begin{aligned}
b_1 &= a_1 \\
b_2 &= a_2 \\
&\ldots \\
b_{i-1} &= a_{i-1} \\
b_{i+1} &= a_{i+m} \\
b_{i+2} &= a_{i+m+1} \\
&\ldots \\
b_n &= a_{n+m-1}
\end{aligned}$$

а b_i има същата стойност като $g(a_i, a_{i+1}, \ldots, a_{i+m-1})$.

Колко са променливите на функцията-композиция? Очевидно множеството от променливите на функцията-композиция е $(\{x_1, \ldots, x_n\} \backslash \{x_i\}) \cup \{y_1, \ldots, y_m\}$. Ако е изпълнено $(\{x_1, \ldots, x_n\} \backslash \{x_i\}) \cap \{y_1, \ldots, y_m\} = \varnothing$[‡], то композицията е функция на $n + m - 1$ променливи. Например, ако са дадени $f(x_1, x_2, x_3, x_4, x_5)$ и $g(x_6, x_7, x_8)$, то композицията $f(x_1, x_2, x_3, g(x_6, x_7, x_8), x_5)$ е функция на $5 + 3 - 1 = 7$ променливи. Възможно е обаче $(\{x_1, \ldots, x_n\} \backslash \{x_i\}) \cap \{y_1, \ldots,$

[†]Не е необходимо тези f и g да са **булеви** функции, за да може да говорим за композиция. Композиция на функцията g на мястото на x_i във функцията f е мислима дори когато f и g са произволни функции при условие, че кодомейнът на g е същият като i-ия домейн на f. Казано на програмистки жаргон, при условие, че типът на изхода на g е същият като типа на i-ия вход на f.

[‡]С други думи, ако променливите на f без x_i, от една страна, и променливите на g, от друга страна, нямат общи елементи.

$y_m\} \neq \varnothing$. В такъв случай броят се получава чрез комбинаторния принцип на включването и изключването: от сумата от мощностите вадим мощността на сечението и вадим още една единица заради x_i. Например, ако са дадени $f(x_1, x_2, x_3, x_4, x_5)$ и $g(x_1, x_2, x_8)$, то композицията $f(x_1, x_2, x_3, g(x_1, x_2, x_8), x_5)$ е функция на $(5 + 3) - 2 - 1 = 5$ променливи.

Това може да се обобщи така. Ако g_1, g_2, ..., g_n са булеви функции съответно на m_1, ..., m_n променливи, а именно

$$g_1(y_{1,1}, \ldots, y_{1,m_1})$$
$$g_2(y_{2,1}, \ldots, y_{2,m_2})$$
$$\ldots$$
$$g_n(y_{n,1}, \ldots, y_{n,m_n})$$

то композицията на g_1 на мястото на x_1, на g_2 на мястото на x_2, ..., на g_n на мястото на x_n е булевата функция:

$$f(g_1(y_{1,1}, \ldots, y_{1,m_1}), g_2(y_{2,1}, \ldots, y_{2,m_2}), \ldots, g_n(y_{n,1}, \ldots, y_{n,m_n}))$$

7 Обобщена конюнкция и обобщена дизюнкция

Нека f_1 е булевата функция конюнкция. Нека x_1, x_2, x_3 и x_4 са булеви променливи. Всеки от следните записи:

$$f_1(x_1, x_2) \qquad f_1(x_1, x_3) \qquad f_1(x_1, x_4) \qquad f_1(x_2, x_3) \qquad f_1(x_2, x_4) \qquad f_1(x_3, x_4)$$

е допустим. От друга страна, следните записи:

$$f_1(x_1, x_2, x_3) \qquad f_1(x_2, x_3, x_4) \qquad f_1(x_1, x_2, x_3, x_4)$$

са, от формална гледна точка, **недопустими**, понеже конюнкцията е функция на точно **два** аргумента, а не на три или повече.

На практика обаче ние говорим за конюнкция на много променливи. Как става това? Ако трябва да сме напълно прецизни, конюнкцията на повече от две променливи не е функцията f_1, а друга функция, която се получава от f_1 чрез подходяща серия от композиции. Като пример да разгледаме следните пет функции:

$$f_1(f_1(f_1(x_1, x_2), x_3), x_4) \qquad f_1(x_1, f_1(x_2, f_1(x_3, x_4))) \tag{2}$$
$$f_1(f_1(x_1, x_2), f_1(x_3, x_4))$$
$$f_1(f_1(x_1, f_1(x_2, x_3)), x_4) \qquad f_1(x_1, f_1(f_1(x_2, x_3), x_4))$$

Всяка от тези пет функции е функция на четирите променливи x_1, x_2, x_3 и x_4. Лесно се вижда, че тези функции отговарят биективно на петте[†] начина за скобуване на редицата от променливите. Прочее, тези пет функции са равни—по-прецизно казано, (2) съдържа пет записа на една и съща функция—поради асоциативността на конюнкцията. Лесно се вижда освен това, че и всяка друга линейна наредба на променливите, да кажем $x_2x_4x_1x_3$ би довела до същото

[†]Странична забележка: в комбинаториката, n-тото число на Catalan, което записваме като C_n, е броят на всички начини дадена линейна наредба на $n + 1$ елемента, взети от някакво множество A, да бъде скобувана така, че дадена бинарна операция над A да бъде приложена над въпросната линейна наредба. В конкретния пример, множеството A е $\{0, 1\}$, броят на обектите е 4, бинарната операция е f_1, а линейната наредба е $x_1x_2x_3x_4$. Действително, $C_3 = 5$, което точно отговаря на броя на функциите в (2).

– функцията на четири променливи си остава същата; това пък е заради комутативността на конюнкцията.

И така, функцията от (2) е пример за *обобщена конюнкция*. Това ще рече, функция на две или повече променливи, която се получава от обикновената конюнкция чрез серия от композиции. Очевидно, обобщената конюнкция има стойност единица тстк всичките ѝ променливи са единици.

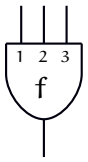
Напълно аналогично дефинираме и *обобщена дизюнкция*: функция на две или повече променливи, която се получава от обикновената дизюнкция чрез серия от композиции; тя има стойност единица тстк поне една от променливите ѝ е единица.

8 Други представяния на булевите функции

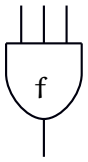
8.1 Представяне чрез схеми от функционални елементи

Да допуснем, че ни е дадено устройство, което има n входа и точно един изход, за някое $n \in \mathbb{N}$. На входовете се подават булеви стойности. Дадена ни е булева функция f с n аргумента. Казваме, че устройството *реализира функцията* f, ако за всяка комбинация-вектор $\mathbf{x}$ от булеви стойности на входовете, на изхода "излиза" булева стойност $f(\mathbf{x})$. Стандартно допускане е, че входовете на устройството са именувани; тоест всеки вход си има идентичност. Всяко такова устройство наричаме *функционален елемент*[†]. За целите на този курс ние разглеждаме само идеализирани функционални елементи, но такива функционални елементи може да бъдат реализирани физически, което е в основата на цифровата схемотехника.

Ще изобразяваме функционалните елементи примерно така:



Елементът се рисува с нещо като полукръг или полуовал, като входовете (в случая те са три на брой и са номерирани с 1, 2 и 3) са откъм правата част, а изходът, който винаги е точно един, е от заобления част. Името на функцията е написано върху елемента. Естествено, функцията трябва да е такава, че броят на аргументите ѝ да е точно равен на броя на входовете на елемента. Ако се разберем, че входовете са номерирани отляво надясно, можем спокойно да изпускаме изписването на номерата им; в такъв случай същият функционален елемент би бил нарисуван така:

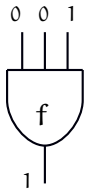


Нека отново f е следната функция:

$$f = 01101101$$

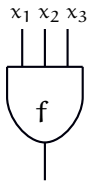
[†]На английски терминът е *gate*.

Тогава върху входния вектор 001, f има стойност 1. Това изобразяваме върху функционалния елемент ето така:



Очевидно допускаме, че има посока на "движение на информацията", която е винаги от входовете към изхода.

Схемите от функционални елементи са над някакво множество от булеви променливи. Удобно е да си представяме, че променливите "текат" по "жиците"[†], които влизат във входовете на функционалните елементи. Например, ако искаме да изобразим функционален елемент, съответстващ на $f(x_1, x_2, x_3)$, правим такава диаграма:

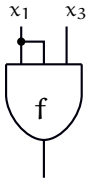


Забележете, че следното свързване на входовете на функционалния елемент към трите променливи е **съществено различно** от предишното

по същите причини, поради които $f(x_1, x_2, x_3)$ в общия случай е различна функция от $f(x_2, x_1, x_3)$ (припомнете си разликата между Таблица 1 и Таблица 2). И така, схемите от функционални елементи чудесно онагледяват какво става при пермутация на променливите в записа на функцията.

Чрез функционални елементи чудесно може да онагледим и унификация на променливи. Ако разгледаме същия функционален елемент, който ползвахме в примерите горе, то унификацията на x_1 с x_2 (припомнете си Таблица 3) изглежда така върху елемента:

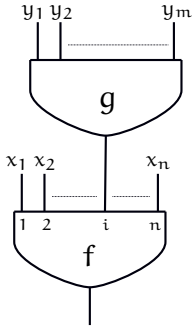
[†] Реалните функционални елементи са електронни устройства, по чиито жици текат токове и има напрежения, като напреженията се **интерпретират** като нули или единици. Идеализираните функционални елементи, които разглеждаме тук, са абстракции и при тях за токове и напрежения не става дума. Върху техните "жици" "текат" нули и единици. При реалните функционални елементи имат някакви закъснения—в природата нищо не се случва мигновено—така че при всяка промяна на входовете е необходимо някакво време, типично от порядъка на наносекунди или пикосекунди при модерните върхови цифрови електронни схеми. Идеализираните функционални елементи, които разглеждаме тук, нямат закъснения и при тях сигналите се разпространяват неограничено бързо.



Ясно се вижда, че след унификацията, променливите са само две. Черната точка е точка на разклонение на сигнала, при което идващата отгоре стойност x_1 се размножава и "влиза" в двата входа вляво. В терминологията на електротехниката, първите два входа на елемента са "на късо".

Използвайки каскадно свързани функционални елементи можем много елегантно да илюстрираме композицията на функция на мястото на променлива във функция. Както в Подсекция 6, нека $f(x_1, x_2, \ldots, x_i, \ldots, x_n)$ и $g(y_1, y_2, \ldots, y_m)$ са булеви функции. Съответните им функционални елементи си представяме така:

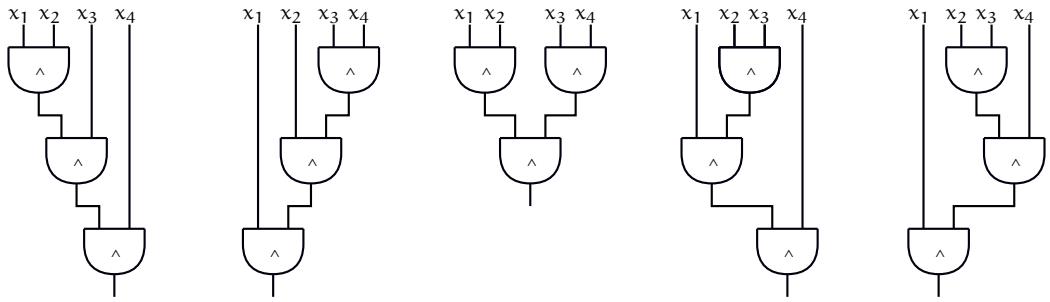
Тогава композицията на g на мястото на x_i във f се описва, в термините на функционалните елементи, като свързване на изхода на елемента на g към i-ия вход на елемента на f:



i-ият вход продължава да съществува, но вече не е именуван с "x_i", защото променлива x_i вече няма в смисъл, че не се ползва. Активните променливи, с други думи, тези, които се ползват, са $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n, y_1, \ldots, y_m$. Тази каскада от функционални елементи реализира функцията $f(x_1, x_2, \ldots, x_{i-1}, g(y_1, y_2, \ldots, y_m), x_{i+1}, x_{i+2}, \ldots, x_n)$.

Функциите обобщена конюнкция и обобщена дизюнкция, за които стана дума в Подсекция 7, може да се илюстрират чрез каскадни свързвания на функционални елементи тип конюнкция или дизюнкция. Нека ето този функционален елемент реализира булевата функция конюнкция:

Тогава петте композиции от (2) могат да бъдат представени така:



Както се каза в Подсекция 7, поради асоциативността на конюнкцията, петте композиции от (2) са всъщност пет начина да бъде написана една и съща функция. Следователно, петте различни свързвания на функционални елементи от последната фигура реализират една и съща функция, а именно обобщена конюнкция на четири променливи. Имаме право да кажем, че тези пет свързвания (схеми) са бихейвиорално (това означава поведенчески) неразличими.

8.2 Представяне чрез хиперкуб

Понякога е много удобно да мислим за булевите функции на n променливи в термините на *хиперкуб*. n-*мерен хиперкуб* е обобщение на редицата от геометрични обекти `точка`, `отсечка`, `квадрат`, `куб` и така нататък:

- точката е 0-мерен хиперкуб, който е атомарен в смисъл, че няма структура,
- отсечката е 1-мерен хиперкуб, състоящ се от 2 точки и едномерния обект, който ги свързва—можем да кажем, в някакъв смисъл, "който е ограден от тях",
- квадратът е 2-мерен хиперкуб, състоящ се от 4 точки, 4 отсечки и двумерния обект, ограден от тях,
- кубът е 3-мерен хиперкуб, състоящ се от 8 точки, 12 околни ръба, 6 квадрата и тримерния обект, ограден от тях,
- и така нататък.

От тази гледна точка, n-мерният хиперкуб е общият член на тази редица. Той е геометричен обект в n-мерното пространство. Може да мислим за хиперкуба като за обект, състоящ се от тези компоненти:

- 2^n точки, които са върховете му; това са 0-мерните компоненти;
- $n2^{n-1}$ отсечки, които са околните му ръбове; това са 1-мерните компоненти;
- $\binom{n}{2}2^{n-2}$ квадрата, които са околните му стени; това са 2-мерните компоненти;
- $\binom{n}{3}2^{n-3}$ куба; това са 3-мерните компоненти;
- и така нататък

- $\binom{n}{n-1}2^{n-(n-1)} = 2n$ на брой, $(n-1)$-мерни компоненти;
- една n-мерна компонента.

Лесно се вижда, че k-мерните компоненти на n-мерния хиперкуб са $\binom{n}{k}2^{n-k}$ на брой.

Може да пренебрегнем геометричния аспект на хиперкуба и да мислим за него като за чисто комбинаторен обект по следния начин:

- Върховете му са векторите от $\{0,1\}^n$. Това са 0-мерните компоненти на n-мерния хиперкуб.
- Два вектора са *съседни* тогава и само тогава, когато се различават в точно една позиция. Например, ако $n = 3$, векторите 001 и 011 се различават в точно една позиция (втората) и са съседни. Те задават един околен ръб на 3-мерния хиперкуб. И така, всеки околен ръб се идентифицира с двата върха, които му принадлежат, а те се различават в точно една позиция. С други думи, 1-мерните компоненти са всички множества от два вектора, които се различават в точно една позиция.
- Аналогично, всяка околна стена се идентифицира с четирите върха, които ѝ принадлежат. С други думи, 2-мерните компоненти са всички множества от четири вектора, такива че за всеки два вектора от тях има точно две позиции, в които те се различават.
- Аналогично, 3-мерните компоненти са всички множества от осем вектора, такива че за всеки два вектора от тях има точно три позиции, в които те се различават.
- И така нататък.
- n-мерната компонента е точно една: това е множеството от всички, 2^n на брой, n-вектори.

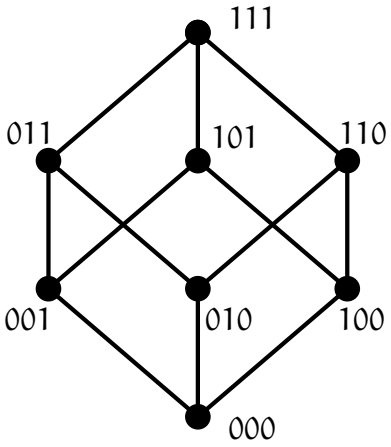
Тогава общият брой компоненти на n-мерния хиперкуб е:

$$\sum_{k=0}^{n}\binom{n}{k}2^{n-k} = \sum_{k=0}^{n}\binom{n}{n-k}2^{n-k} = \sum_{0\leqslant k\leqslant n}\binom{n}{n-k}2^{n-k} =$$
$$\sum_{0\geqslant -k\geqslant -n}\binom{n}{n-k}2^{n-k} = \sum_{n\geqslant n-k\geqslant 0}\binom{n}{n-k}2^{n-k} = \sum_{0\leqslant k\leqslant n}\binom{n}{k}2^{k} = \sum_{0\leqslant k\leqslant n}\binom{n}{k}2^{k}1^{n-k} = 3^n$$

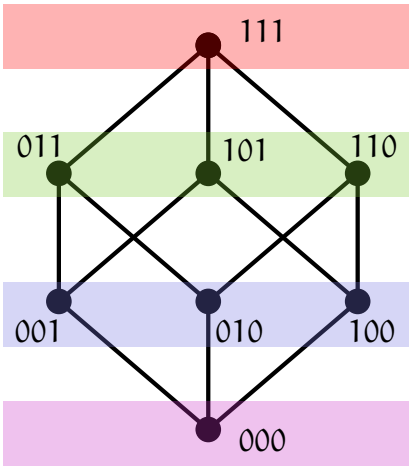
Като пример да разгледаме 3-мерния хиперкуб, записан **напълно подробно**:

$$\{$$
$$\{\{000\},\{001\},\{010\},\{011\},\{100\},\{101\},\{110\},\{111\}\},\quad //8 \text{ върха}$$
$$\{\{000,100\},\{000,010\},\{010,011\},\{001,011\},\{000,100\},\{001,101\},$$
$$\{011,111\},\{010,110\},\{100,101\},\{101,111\},\{111,110\},\{110,100\}\}\quad //12 \text{ ръба}$$
$$\{\{000,001,011,010\},\{000,100,101,001\},\{001,101,111,011\},$$
$$\{010,110,100,000\},\{010,110,111,011\},\{100,101,111,110\}\},\quad //6 \text{ стени}$$
$$\{\{000,001,010,011,100,101,110,111\}\}\quad //1 \text{ обем}$$
$$\}$$

Обикновено хиперкубът се рисува като граф: само върховете и страните. Това означава, че 0-мерните и 1-мерните компоненти се изобразяват, а останалите, не. Така е много по-прегледно. Но ние знаем, че хиперкубът като комбинаторен обект е съвкупност от обектите от всички размерности, от 0 до n [†]. Ето типично изображение на 3-мерния хиперкуб:

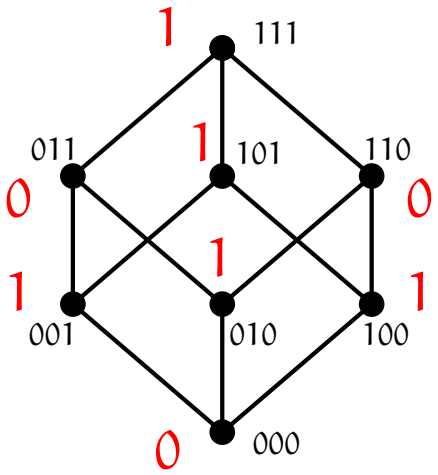


Върховете на n-мерния хиперкуб се разбиват на $n+1$ *слоя*. Върховете от един слой са точно тези вектори, които имат един и същи брой единици. Броят на единиците може да е $0, 1, \ldots, n$, затова и слоевете са $n+1$. Когато говорим за слой k, $0 \leqslant k \leqslant n$, имаме предвид слоя от векторите, всеки от които има точно k единици. Очевидно слой k има $\binom{n}{k}$ вектора в себе си. Ето същия хиперкуб, като четирите слоя са указани с различни цветове:



Всяка булева функция на n променливи може да бъде разглеждана като асоцииране на всеки връх на n-мерния хиперкуб (помним, че върховете му са точно n-векторите) с една булева стойност. Например, функцията $f = 01101101$ от миналата подсекция се изобразява върху хиперкуба така:

[†] Ако говорим за *граф-хиперкуб*, тогава имаме предвид обекта, който е съвкупност само от 0-мерните компоненти (върховете) и 1-мерните компоненти, които в този контекст наричаме "ребра". n-мерен граф-хиперкуб не е същото нещо като n-мерен хиперкуб: графът-хиперкуб е подмножество на хиперкуба. Хиперкубът съдържа и компонентите от размерности $\geqslant 2$.



Стойностите на функцията върху векторите са написани с червено.

От казаното дотук изглежда, че каноничното представяне на дадена булева функция и представянето чрез хиперкуб са едно и също нещо. Всъщност, разлика има и тя е само в наредбата на векторите. При каноничното представяне, векторите са наредени лексикографски[†], а при представянето с хиперкуб те са наредени от частичната наредба $\preccurlyeq \subseteq \{0,1\}^n \times \{0,1\}^n$, дефинирана по следния начин:

$$\forall a, b \in \{0,1\}^n : a \preccurlyeq b \leftrightarrow (\forall i \in \{1,2,\ldots,n\} : a_i \leqslant b_i) \tag{3}$$

Релацията $\preccurlyeq$ е *релацията на предшествие* върху множеството от n-векторите. Тя е частична наредба, която за $n > 1$ не е линейна, понеже има двойки вектори, които не са сравними (спрямо нея), например 011 и 100. Релацията на предшествие е полезна за осмислянето на различни понятия и решаването на много задачи от областта на булевите функции. На горния пример всъщност е показана диаграмата на Hasse на частичната наредба $\preccurlyeq$ върху 3-векторите.

Забележете разликата между релацията на предшествие, която не е линейна наредба, и лексикографската наредба, която е линейна. Лексикографската наредба е напълно условна в смисъл, че ние сме я предпочели измежду всички линейни наредби по субективни причини; тя няма никакво иманентно предимство пред останалите линейни наредби на n-векторите. Докато релацията на предшествие се основава на обективно важен начин да смятаме, че един вектор предшества друг. Забележете, че релацията на предшествие е практически същата като релацията $\subseteq_S$ от лекцията по релации! За да се убедите в това, интерпретирайте n-векторите като характеристичните вектори на n-елементно наредено множество.

Както казахме вече, съседни вектори са такива, които се различават в точно една позиция[‡]. Съседството на вектори може да осмислим и в термините на хиперкуба: два негови вектора са съседни тстк те са в съседни слоеве.

Съседство на вектори може да осмислим и чрез релацията $\preccurlyeq$ (виж (3)). А именно, ако a и b са n-вектори, то те са съседни тогава и само тогава, когато $a \prec b$ или $b \prec a$, където

[†]Лексикографската наредба е линейна.

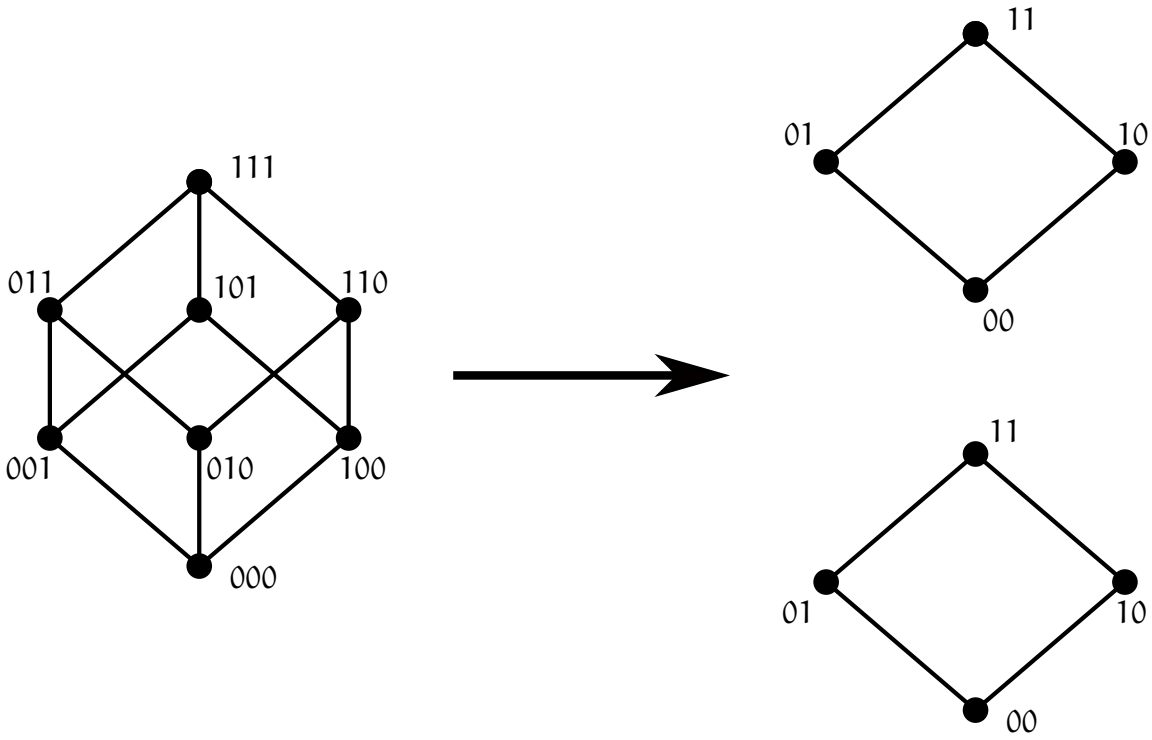
[‡]Освен това, за да говорим за съседство на вектори, трябва те да имат една и съща дължина. При вектори с различни дължини за съседство не може да става дума.

релацията $\prec$ е *релацията на непосредствено предшествие*. Тя се дефинира така:

$$\forall x, y \in \{0,1\}^n \ : \ x \prec y \overset{\text{def}}{\leftrightarrow} x \leqslant y \wedge x \neq y \wedge \neg\exists z\,(z \neq x \wedge z \neq y \wedge x \leqslant z \leqslant y) \tag{4}$$

Например, $0010 \prec 0011$ и $0010 \prec 1010$, но $0010 \not\prec 1111$, въпреки че $0010 \leqslant 1111$.

Срязване на n-мерния хиперкуб в i-тата дименсия е понятие, което първо ще онагледим с пример. Ето срязване на 3-мерния хиперкуб във втората дименсия:



За удобство, нека да мислим за хиперкуба като за граф-хиперкуб, тоест съвкупност от върхове и ребра. Срязването се състои в премахване на i-тата позиция на всички вектори-върхове, след което тяхната дължина става $n-1$, и премахването на точно тези ребра, които са от вида:

$$\{\alpha 0 \beta, \alpha 1 \beta\}$$

където α е булев вектор с дължина $i-1$, а β е булев вектор с дължина $n-i$. В примера със срязването на 3-мерния хиперкуб във втората дименсия, ребрата, които махаме, са точно

$$\{000, 010\}, \{001, 011\}, \{100, 110\}, \{101, 111\}$$

Ако гледаме на хиперкуба не като на граф, а като на "истински" хиперкуб с компоненти от всички възможни размерности, ясно е, че срязването води до изчезването на n-мерната компонента, както и до намаляването на броя на k-мерните компоненти от $\binom{n}{k}2^{n-k}$ на $\binom{n-1}{k}2^{n-k-1}$, тъй като резултатът от срязването е появата на два нови хиперкуба, всеки с размерност $n-1$.

От казаното досега може да не е ясно защо настояваме да се казва, че срязваме именно в i-тата размерност. Например, на последната фигура от един куб се получават два квадрата и по нищо не личи точно в коя от трите размерности е бил срязан куба. Отговорът на тази забележка е, че хиперкубът ни интересува в контекста на булевите функции, когато върховете му са "маркирани" с нули или единици – стойностите на булевата функция. При срязването асоциацията между върхове и стойности на функцията **се запазва**, така че в общия случай резултатът от срязването в различни размерности е **различен**.

8.3 Представяне чрез формули

Формула е чисто синтактично понятие. Средношколското разбиране за "формула" е "прост алгоритъм", например "формулата за лицето на кръг с радиус r е $S = \pi r^2$". Тук ние възприемаме съвсем друго разбиране за "формула". Формула[†] е всеки стринг, конструиран над дадена азбука съгласно дадени правила. Азбуката винаги е крайна, а множеството от формулите по правило е безкрайно.

Формулите на булевите функции може да се дефинират индуктивно по следния начин. Фиксираме изброимо безкрайно множество от булеви променливи $\{x_0, x_1, \ldots\}$. Азбуката е:

$$\Sigma = \{f, x, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, (,), ,\}$$

С червено са записани буквите от езика, който ще опишем, а именно езика от формулите на булевите функции, а с черно са буквите от метаезика, който използваме, за да опишем езика от формулите на булевите функции. Нека $\Sigma_d = \{0, 1, \ldots, 9\}$. Нека $\widetilde{\Sigma} \subset \Sigma_d^+$ е множеството от стрингове над Σ_d, които са валидни записи[‡] на числа в десетична позиционна бройна система. Нека ι е стандартната десетична позиционна бройна система, тоест биекцията

$$\iota : \widetilde{\Sigma} \to \mathbb{N}$$

която знаем от училище. Нека *enum* е произволно изброяване на всички булеви функции, тоест биекция $enum : \mathcal{F} \to \mathbb{N}$. За удобство вземаме най-простото и естествено изброяване на булевите функции:

- за всяко $n \in \mathbb{N}$ и всяко $k \in \mathbb{N}^+$, ако $f \in \mathcal{F}^n$ и $g \in \mathcal{F}^{n+k}$, то $enum(f) < enum(g)$,
- за всяко $n \in \mathbb{N}$, ако $f, g \in \mathcal{F}^n$ и $f \neq g$, то $enum(f) < enum(g)$ тогава и само тогава, когато каноничното представяне на f предхожда лексикографски каноничното представяне на g.

Нека изброените от *enum* булеви функции са f_0, f_1 и така нататък. Да разгледаме началото на изброяването. f_0 е константа нула с нула аргумента, тоест 0. f_1 е константа единица с нула аргумента, тоест 1. f_2 е константа нула с един аргумент, тоест 00. f_3, f_4 и f_5 са съответно 01, 10 и 11. f_6 е константа нула с два аргумента, тоест 0000. f_7 е 0001. И така нататък. f_{21} е константа единица с два аргумента, тоест 1111. f_{22} е константа нула с три аргумента, тоест 00000000. f_{23} е 00000001. И така нататък. f_{277} е константа единица с три аргумента, тоест 11111111. f_{278} е константа нула с четири аргумента, тоест 0000000000000000. И така нататък. f_{65813} е константа единица с четири аргумента, тоест 1111111111111111. f_{65814} е константа нула с пет аргумента. И така нататък.

f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	$\ldots$	f_{21}	f_{22}	f_{23}	$\ldots$	f_{227}	$\ldots$
0	1	0	0	1	1	0	0		1	0	0		1	
		0	1	0	1	0	0		1	0	0		1	
						0	0		1	0	0		1	
						0	1		1	0	0		1	
										0	0		1	
										0	0		1	
										0	0		1	
										0	1		1	

Тогава дефинираме "формула на булева функция" чрез следната индуктивна дефиниция. Тя формализира понятието "формула на булена функция" и въвежда дълбочина на формула, която бележим с ν.

[†]Етимологията на думата е следната: на латински "formula" е умалително от "forma". Не е грешка да казваме "форма" вместо "формула".

[‡]Например 0017 не е валиден запис заради двете водещи нули.

Определение 4 (Формули на булевите функции, дълбочина на формула)**.** *Всеки стринг* $\phi \in \Sigma^+$ *е* формула на булева функция *тогава и само тогава, когато в сила е точно едно от двете:*

- ***База.*** $\phi = x\alpha$*, където* $\alpha \in \widetilde{\Sigma}$*.* $\nu(\phi) \overset{деф}{=} 0$*.*
- ***Индуктивна стъпка.*** $\phi = f\alpha(\phi_1,\phi_2,\ldots,\phi_n)$ *където* ϕ_1*,* ϕ_2*, . . . ,* ϕ_n *са формули на булеви функции,* $\alpha \in \widetilde{\Sigma}$ *и* $enum^{-1}(\iota(\alpha))$ *има точно* n *аргумента*[†]*. Последното е важно! Забележете, че* $f999999(x1, x2)$ *е синтактично невалиден запис, защото булевата функция номер деветстотин деветдесет и девет хиляди деветстотин деветдесет и девет е с пет аргумента съгласно номерацията* $enum$*.*

$$\nu(\phi) \overset{деф}{=} \max\{\nu(\phi_1), \nu(\phi_2), \ldots, \nu(\phi_n)\} + 1$$

Нека Φ *е множеството от всички формули на булеви функции. Функцията* $\nu : \Phi \to \mathbb{N}$*, която току-що дефинирахме, се нарича* дълбочина*.* □

Неформално, ако си представим формулата, реализирана чрез каскада от функционални елементи, то дълбочината на формулата е максималният брой функционални елементи, през които преминава сигналът.

Дотук сме дефинирали формулите на булевите функции чисто синтактично. Не сме казали нищо за техния смисъл, или, иначе казано, за тяхната *семантика*. Семантиката можем да дефинираме, използвайки индуктивната дефиниция на синтаксиса, като аналогът на синтактичната операция "вмъкване на стринг на мястото на подстринг" (има се предвид вмъкването на формулите ϕ_i на местата на имената на променливите) е семантичната "композиция на функция на мястото на променлива във функция".

Определение 5 (Семантика на формулите на бул. функции)**.** *В контекста на Определение 4:*

- *Семантиката на всяка формула* $x\alpha$ *от базата е булевата променлива* $x_{\iota(\alpha)}$*.*
- *Нека* ϕ *е произволна формула от индуктивната стъпка. Да кажем,* $\phi = f\alpha(\phi_1,\phi_2,\ldots,\phi_n)$*, където* ϕ_1*,* ϕ_2*, . . . ,* ϕ_n *са формули. Тогава семантиката на* ϕ *е булевата функция* $f_{\iota(\alpha)}(g_1, g_2, \ldots, g_n)$*, където, за* $1 \leqslant i \leqslant n$*,*
  - *ако* ϕ_i *е от вида* $x\alpha$*, то* g_i *е булевата променлива* $x_{\iota(\alpha)}$*,*
  - *в противен случай,* g_i *е композицията на семантиката на* ϕ_i *на мястото на* i*-тата променлива на* $f_{\iota(\alpha)}$*.*

Лесно се вижда, че при изредените правила булевата функция, която е семантиката на някаква формула, е **една единствена**, но обратното не е вярно: за всяка функция има **безброй много** формули, на които тя е семантика. Ако булевата функция f е семантиката на формулата ϕ ще казваме, че ϕ *реализира* f. И така, ϕ реализира точно една функция f, но безброй много формули реализират функцията f.

Често срещана задача в теорията на булевите функции е тази: дадени са две формули и трябва да се определи дали те са еквивалентни. Тоест, дали съответната им булева функция е една и съща, или не. Това е частен случай на общата задача в Компютърните науки: дадени са два синтактични обекта (някакви стрингове, изградени по някакви правила) и трябва да

[†]Забележете, че $\iota(\alpha)$ е число, а $enum^{-1}(\iota(\alpha))$ е една от всички булеви функции, защото $enum^{-1} : \mathbb{N} \to \mathcal{F}$.

се определи дали семантиката им е една и съща, или не, като семантиката е някаква добре дефинирана функция.

Формулите, които ще разглеждаме на практика, по правило са само над ограничено множество булеви функции, което най-често е крайно. Нека $\mathsf{F} \subseteq \mathcal{F}$ е непразно; дали е крайно или не, няма значение. *Формулите над* F е това подмножество на Φ, чиито елементи–формули реализират функциите от F; с други думи, в индуктивната стъпка на Определение 4, стрингът от цифри α е такъв, че $\mathsf{f}_{\iota(\alpha)}$ е елемент на F.

Функциите, чиито формули ще използваме в този курс, са "стандартните" булеви функции на две променливи: конюнкцията, дизюнкцията, импликацията, сумата по модул 2, стрелката на Peirce и чертата на Sheffer, а така също и отрицанието, което е функция на една променлива. Има смисъл множеството от функциите, чиито формули ще се ползват, да бъде пълно множество. Това понятие ще разгледаме в следващата лекция.

9 Дизюнктивна Нормална Форма и Съвършена Дизюнктивна Нормална Форма

Определение 4 и Определение 5 дават мощен механизъм за изграждане на формули, чиито смисъл (семантика) са булеви функции. Този механизъм има съществен недостатък: формулите, изградени по него, са практически нечетими от хора. Заради това, в Подсекция 9 и Подсекция 10 ще въведем формули, които се изграждат по съвсем различни правила. Тези формули са лесно четими от хора и са универсални в смисъл, че всяка булева функция се реализира от някои от тези формули, както ще видим в следващата лекция. Що се отнася до семантиката обаче, тези формули не са над произволно множество от булеви функции, а само над функциите отрицание, конюнкция и дизюнкция.

Нека са фиксирани краен брой булеви променливи $x_1, \ldots, x_n$ за $n \geqslant 1$. *Литерал* ще наричаме всяко име на променлива или всяко име на променлива с черта отгоре. Литералите от първия вид се наричаме *положителни*, а от втория – *отрицателни*. Веднага подчертаваме, че литералите са **букви** и като такива са качествено различни от самите променливи, понеже буквите са понятия от синтактичното ниво, а променливите са от по-високото семантично ниво. Това, че използваме един и същи запис "x_1" и за името на променлива, което е буква, и за самата променлива, не води до объркване, защото опитният читател винаги може да разбере от контекста дали става дума за синтактичното ниво или за семантичното ниво. Примери за положителни литерали са x_1, x_4 и така нататък. Примери за отрицателни литерали са $\overline{x_1}$, $\overline{x_3}$ и така нататък. Отново: литералите са букви. Ерго, ако променливите са n на брой, то азбуката на формулите, които ще строим, съдържа $2n$ букви само заради променливите; а именно, n на брой положителни литерали и n на брой отрицателни литерали.

Конюнктивна клауза е всяка непразна формула, която се състои от конкатенация на литерали, такива че всяко име на променлива се появява най-много веднъж – било като положителен, било като отрицателен литерал. Ако променливите са $x_1, \ldots, x_6$, примери за конюнктивни клаузи са $x_1x_3x_4$, $x_1\overline{x_4}$, $x_1x_2x_3x_4x_5x_6$, $\overline{x_3}$ и $x_2\overline{x_5}\,\overline{x_6}$. Не са конюнктивни клаузи $x_1x_1x_2$ (има повторение на литерал) и $x_1\overline{x_2}x_2$ (срещат се положителният и отрицателният литерал на една и съща променлива). Не е задължително, но е силно препоръчително имената да се записват отляво надясно в нарастващ ред на индексите.

Пълна конюнктивна клауза е конюнктивна клауза, която съдържа точно n литерала. С други думи, това е непразна формула, която се състои от конкатенация на литерали, такива че всяко име на променлива се появява точно веднъж – било като положителен, било като отрицателен литерал. Ако променливите са $x_1, \ldots, x_6$, примери за пълни конюнктивни клаузи

са $x_1x_2x_3x_4x_5x_6$ и $x_1\overline{x_2}\,\overline{x_3}\,\overline{x_4}\,\overline{x_5}\,x_6$. Не е задължително, но е силно препоръчително имената да се записват отляво надясно в нарастващ ред на индексите.

Дизюнктивна нормална форма, съкратено ДНФ, е формула, която се състои от една или повече различни[†] конюнктивни клаузи, конкатенирани с буквата "$\vee$". В горния контекст, примери за ДНФ са $x_2\overline{x_3}x_4$ и $x_1x_4 \vee x_2\overline{x_5}\,\overline{x_6} \vee \overline{x_2}\,\overline{x_3}$. *Съвършена дизюнктивна нормална форма*, съкратено *СъвДНФ* е дизюнктивна нормална форма, в която участват само пълни конюнктивни клаузи. В горния контекст, пример за СъвДНФ е $x_1\overline{x_2}x_3x_4x_5x_6 \vee \overline{x_1}\,\overline{x_2}\,\overline{x_3}\,\overline{x_4}\,\overline{x_5}\,\overline{x_6}$.

Семантиката на литералите, конюнктивните клаузи и ДНФ е очевидната:

- Семантиката на всеки положителен литерал x_i е функцията идентитет-x_i. Семантиката на всеки отрицателен литерал $\overline{x_i}$ е функцията отрицание-на-x_i.
- Семантиката на всяка конюнктивна клауза $\lambda_1\lambda_2\cdots\lambda_k$, където λ_j са литерали за $1 \leqslant j \leqslant k$, е композицията $f_{con}(f_1, f_2, \ldots, f_k)$, където f_{con} е обобщената конюнкция на k променливи, а f_j е семантиката на λ_j, за $1 \leqslant j \leqslant k$.
- Семантиката на всяка ДНФ $\phi_1 \vee \phi_2 \vee \cdots \vee \phi_k$, където ϕ_j е конюнктивна клауза за $1 \leqslant j \leqslant k$, е $f_{dis}(f_1, f_2, \ldots, f_k)$, където f_{dis} е обобщената дизюнкция на k променливи, а f_j е семантиката на ϕ_j, за $1 \leqslant j \leqslant k$.

Пример за СъвДНФ. Като пример да разгледаме формулата (тя е СъвДНФ, ако $n = 6$)

$$\phi = x_1\overline{x_2}x_3x_4x_5x_6 \vee \overline{x_1}\,\overline{x_2}\,\overline{x_3}\,\overline{x_4}\,\overline{x_5}\,\overline{x_6}$$

Очевидно, нейната семантика е булевата функция—да я наречем h—на шестте променливи x_1, $\ldots$, x_6, която има стойност 1 върху векторите 000000 и 101111 и има стойност 0 върху всички останали, 62 на брой, вектори. Сега да си представим, че трябва да запишем h чрез формула, изградена съгласно индуктивното Определение 4. Естествено, има безброй начини да сторим това, но нека се опитаме да напишем формула, която е аналогична на ϕ. Лесно се вижда, че ни трябват формули за функциите отрицание, конюнкция и дизюнкция. Функцията h е равна на някаква композиция от тези функции[‡], а именно на дизюнкция от някакви конюнкции. Аналогът на това в синтактичния свят на формулите е: че формула за h може да бъде получена, като във формула за дизюнкция заместим стринговете-имена на променливи с някакви формули за конюнкции.

Да направим формула за h точно по Определение 4, като използваме червен цвят за буквите й. Функциите отрицание, конюнкция и дизюнкция имат номера съответно 4, 7 и 13 в изброяването *enum*, тоест, това са съответно f_4, f_7 и f_{13}. Ето пример за формула, съответна на h:

$$\begin{aligned}\psi = {} & \mathrm{f13}(\mathrm{f7}(\mathrm{x1}, \mathrm{f7}(\mathrm{f4}(\mathrm{x2}), \mathrm{f7}(\mathrm{x3}, \mathrm{f7}(\mathrm{x4}, \mathrm{f7}(\mathrm{x5}, \mathrm{x6}))))),\\ & \mathrm{f7}(\mathrm{f4}(\mathrm{x1}), \mathrm{f7}(\mathrm{f4}(\mathrm{x2}), \mathrm{f7}(\mathrm{f4}(\mathrm{x3}), \mathrm{f7}(\mathrm{f4}(\mathrm{x4}), \mathrm{f7}(\mathrm{f4}(\mathrm{x5}), \mathrm{f4}(\mathrm{x6})))))))\end{aligned}$$

Очевидно ϕ е несравнимо по-лека за четене от ψ, макар че са еквивалентни, имайки една и съща семантика. Може да възникне въпросът, защо изобщо ползваме тромавата конструкция

[†]Кога две конюнктивни клаузи са различни? Ако настояваме индексите на променливите да са в нарастващ ред отляво надясно, то две конюнктивни клаузи са различни тстк са различни като стрингове. Без това ограничение, можем да дефинираме "различни формули" и в частност различни конюнктивни клаузи чрез релация на еквивалентност.

[‡]Тази композиция не е единствена заради комутативността на дизюнкцията и конюнкцията.

на Определение 4, след като има начин да се записват еквивалентни формули, които са много по-лесни за четене. Отговорът е, че конструкцията на Определение 4 има чисто теоретично значение. Там искахме да дефинираме прецизно и кратко "формула" и "функция, съответна на формула", а не сме имали за цел получените формули да са кратки и ясни. Говорейки за ДНФ искаме друго – кратък и много ясен запис на формулите. За тази цел е много по-удачно да се въведат литерали и конюнктивни клаузи и чрез тях и буквите "$\vee$" да се дефинират ДНФ.

10 Конюнктивна Нормална Форма и Съвършена Конюнктивна Нормална Форма

Нека са фиксирани краен брой булеви променливи $x_1, \ldots, x_n$ за $n \geqslant 1$. *Дизюнктивна клауза* е всяка непразна формула, която се състои от литерали, конкатенирани с буквата "$\vee$", такива че всяко име на променлива се появява най-много веднъж – било като положителен, било като отрицателен литерал. Ако променливите са $x_1, \ldots, x_6$, примери за дизюнктивни клаузи са:

$$x_1 \vee x_3 \vee x_4,$$
$$x_1 \vee \overline{x_4},$$
$$x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6,$$
$$\overline{x_3},$$
$$x_2 \vee \overline{x_5} \vee \overline{x_6}$$

Не е задължително, но е силно препоръчително имената да се записват отляво надясно в нарастващ ред на индексите.

Пълна дизюнктивна клауза е дизюнктивна клауза, която съдържа точно n литерала. С други думи, това е непразна формула, която се състои от литерали, конкатенирани с буквата "$\vee$", такива че всяко име на променлива се появява точно веднъж – било като положителен, било като отрицателен литерал. Ако променливите са $x_1, \ldots, x_6$, примери за пълни дизюнктивни клаузи са:

$$x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6,$$
$$x_1 \vee \overline{x_2} \vee \overline{x_3} \vee \overline{x_4} \vee \overline{x_5} \vee x_6$$

Не е задължително, но е силно препоръчително имената да се записват отляво надясно в нарастващ ред на индексите.

Конюнкивна нормална форма, съкратено КНФ, е формула, която се състои от конкатенация на една или повече различни дизюнктивни клаузи, като, ако дизюнктивните клаузи са повече от една, всяка от тях е оградена от чифт скоби; ако дизюнктивната клауза е само една, скоби не се ползват. В горния контекст, примери за КНФ са:

$$x_2 \vee \overline{x_3} \vee x_4,$$
$$(x_1 \vee x_4 \vee x_2 \vee \overline{x_5})(\overline{x_6} \vee \overline{x_2} \vee \overline{x_3})$$

и така нататък.

Съвършена конюнктивна нормална форма, съкратено СъвКНФ е конюнктивна нормална форма, в която участват само пълни дизюнктивни клаузи. В горния контекст, пример за СъвКНФ е $(x_1 \vee \overline{x_2} \vee x_3 \vee x_4 \vee x_5 \vee x_6)(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3} \vee \overline{x_4} \vee \overline{x_5} \vee \overline{x_6})$.

Семантиката на КНФ е очевидната:

- Семантиката на всеки положителен литерал x_i е функцията идентитет-x_i. Семантиката на всеки отрицателен литерал $\overline{x_i}$ е функцията отрицание-на-x_i.
- Семантиката на всяка дизюнктивна клауза $\lambda_1 \vee \lambda_2 \vee \cdots \vee \lambda_k$, където λ_j са литерали за $1 \leqslant j \leqslant k$, е композицията $f_{\text{dis}}(f_1, f_2, \ldots, f_k)$, където f_{dis} е обобщената дизюнкция на k променливи, а f_j е семантиката на λ_j, за $1 \leqslant j \leqslant k$.
- Семантиката на всяка КНФ $(\phi_1)(\phi_2)\cdots(\phi_k)$, където ϕ_j е дизюнктивна клауза за $1 \leqslant j \leqslant k$, е $f_{\text{con}}(f_1, f_2, \ldots, f_k)$, където f_{con} е обобщената конюнкция на k променливи, а f_j е семантиката на ϕ_j, за $1 \leqslant j \leqslant k$.

11 Валюация на ДНФ или КНФ

При ДНФ и КНФ променливите са първични: първо е дадено множество от булеви променливи и от него, по някакви правила, правим формули и булевите функции се появяват като семантиките на тези формули. Това е радикално различен подход от подхода в Секция 2, в който функциите са първични, а променливите са някакви именувания на аргументите.

Нека е дадена ДНФ или КНФ ϕ над множество от булеви променливи $X = \{x_1, x_2, \ldots, x_n\}$. "Валюация" на тези булеви променливи практически съвпада с понятието "валюация" от лекцията по съждителна логика, с една незначителна разлика: в съждителната логика валюация беше раздаване на F или T на простите съждения, докато сега валюация е раздаване на 0 или 1 на булевите променливи. Формално, валюация на X е всяка функция $\mathbf{t} : X \to \{0, 1\}$. На английски често се ползва терминът *truth assignment*, откъдето и името "$\mathbf{t}$" на функцията.

Нека $\mathbf{t}$ е валюация на ϕ. Дефинираме *стойността на* ϕ *по отношение на* $\mathbf{t}$. Означаваме тази стойност с "$\mathsf{TV}(\phi, \mathbf{t})$", също както в съждителната логика. Но сега ще опишем изчисляването на $\mathsf{TV}(\phi, \mathbf{t})$ по начин, различен от онзи в лекцията по съждителна логика. ДНФ и КНФ имат дълбочина, не по-голяма от три, така че можем да опишем изчисляването на стойността без рекурсия[†], ползвайки понятието "рестрикция на функция".

- Първо да разгледаме литералите на ϕ. Нека α е литерал във ϕ и x_i е променливата на α. *Стойността на* α *под* $\mathbf{t}$ е тази:
  - ако α е положителен литерал, то стойността на α под $\mathbf{t}$ е $\mathbf{t}\big|_{x_i}$; тоест, тя съвпада със стойността, която $\mathbf{t}$ дава на променливата x_i;
  - ако α е отрицателен литерал, то стойността на α под $\mathbf{t}$ е $\overline{\mathbf{t}\big|_{x_i}}$, тоест, тя е обратната на стойността, която $\mathbf{t}$ дава на променливата x_i.
- Сега да разгледаме клаузите. Нека λ е клауза във ϕ. Стойността на λ под $\mathbf{t}$ е тази:
  - ако ϕ е ДНФ, то λ е конюнктивна клауза и стойността на λ под $\mathbf{t}$ се определя по правилата на конюнкцията: ако поне един литерал на λ има стойност 0 под $\mathbf{t}$, то λ има стойност 0 под $\mathbf{t}$, в противен случай λ има стойност 1 под $\mathbf{t}$;
  - ако ϕ е КНФ, то λ е дизюнктивна клауза и стойността на λ под $\mathbf{t}$ се определя по правилата на дизюнкцията: ако поне един литерал на λ има стойност 1 под $\mathbf{t}$, то λ има стойност 1 под $\mathbf{t}$, в противен случай λ има стойност 0 под $\mathbf{t}$.
- Сега да разгледаме цялата ϕ. Стойността на ϕ под $\mathbf{t}$ е тази:

[†]Разгледайте отново изчисляването на стойността на $\mathsf{TV}(R, \nu)$ в съждителната логика и ще се убедите, че то се основава на индуктивната дефиниция на "съставно съждение" и работи за произволна дълбочина.

- ако ϕ е ДНФ, то, ако поне една конюнктивна клауза има стойност 1 под $\mathbf{t}$, то ϕ има стойност 1 под $\mathbf{t}$, в противен случай ϕ има стойност 0 под $\mathbf{t}$;
- ако ϕ е КНФ, то, ако поне една дизюнктивна клауза има стойност 0 под $\mathbf{t}$, то ϕ има стойност 0 под $\mathbf{t}$, в противен случай ϕ има стойност 1 под $\mathbf{t}$.

Забележете, че булевата функция $f(x_1, \ldots, x_n)$, която е семантиката на ϕ, е множеството от всички наредени двойки $(\mathbf{t}, x)$, където $\mathbf{t}$ е валюация на ϕ, а x е стойността на ϕ под $\mathbf{t}$. В общия случай не можем да изчислим ефикасно f в канонично представяне (като вектор от нули и единици), защото размерът на това представяне е 2^n. Дори за скромни стойности на n като няколкостотин, 2^n е напълно отвъд възможностите на всеки физически компютър. Отново: това, което можем да изчислим ефикасно, е стойността на f върху една конкретна валюация на променливите.