Algebraické štruktúry III

- **Spínacie obvody**
- **Logické obvody**
- **Minimalizácia Boolových výrazov pomocou Quine a McCluskey metódy**

Spínacie obvody

Mnohé elektronické zariadenia, akými sú napr. počítače, telefónne ústredne, zariadenia na riadenie dopravy, obsahujú ako časť spínacie obvody.

Spínač môže byť chápaný ako taký spoj v obvode, ktorý ak je uzavretý, potom ním prechádza elektrický prúd, v opačnom prípade, ak je otvorený, elektrický prúd ním neprechádza. Spínač môžeme znázorniť takto:

A

Predpokladajme , že v spínacom obvode máme spínač A. Stav tohto spínača označíme premennou x, ak $x = 1$ ($x = 0$), potom spínač A je uzavretý (otvorený).

O trochu zložitejší prípad spínacieho obvodu obsahuje dva spínače A_1 a A_2

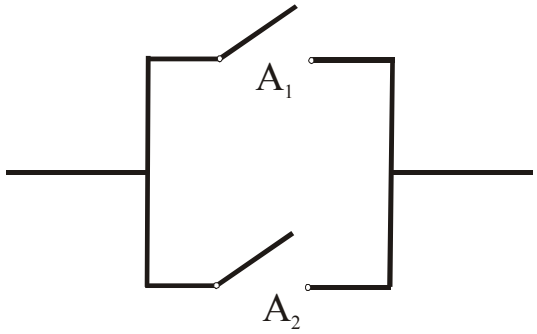
Hovoríme, že v tomto prípade sú spínače zapojené *sériovo*. Nech x_1 a x_2 sú premenné popisujúce stavy spínačov A_1 resp. A_2, tieto premenné ak sa rovnajú 1 (0) , potom daný spínač je uzavretý (otvorený). Nech $f(x_1, x_2)$ je funkcia, ktorej hodnota sa rovná 1 (0) pre tie hodnoty x_1 a x_2, ktoré umožňujú (znemožňujú) tok prúdu.

#	x_1	x_2	$f(x_1, x_2)$
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

spínacia funkcia

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

Ďalší druh spínacieho obvodu má paralélne zapojenie spínačov

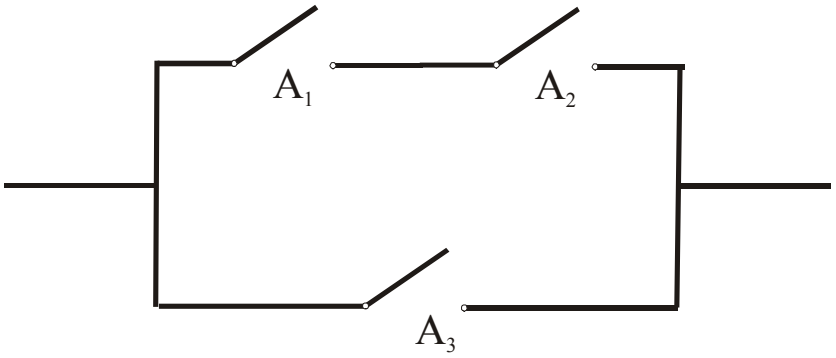


#	x_1	x_2	$g(x_1, x_2)$
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	1

Spínacia funkcia

$$g(x_1, x_2) = x_1 + x_2$$

Nasledujúci príklad spínacieho obvodu bude zložitejší spínací obvod, ktorý obsahuje tri spínače v sériovo-paralelnom zapojení



Spínacia funkcia má tvar

$$f\left(x_1, x_2, x_3\right) = f_1\left(x_1, x_2\right) + f_2\left(x_3\right) = x_1 x_2 + x_3$$

Príklad

Nech na začiatku a konci chodby sú umiestnené stenové vypínače S_1 a S_2, pomocou ktorých zapneme alebo vypneme svetlo nad schodišťom. Požadujeme, aby na každom konci sme svetlo mohli vypnúť alebo zapnúť nezávisle od polohy druhého vypínača. Alternatívna formulácia, ak sú oba spínače S_1 a S_2 vypnuté alebo zapnuté, potom zariadením nepreteká prúd, ale stačí, aby bol zapnutý práve jeden vypínač, potom zariadením preteká prúd, čo môžeme vyjadriť touto tabuľkou

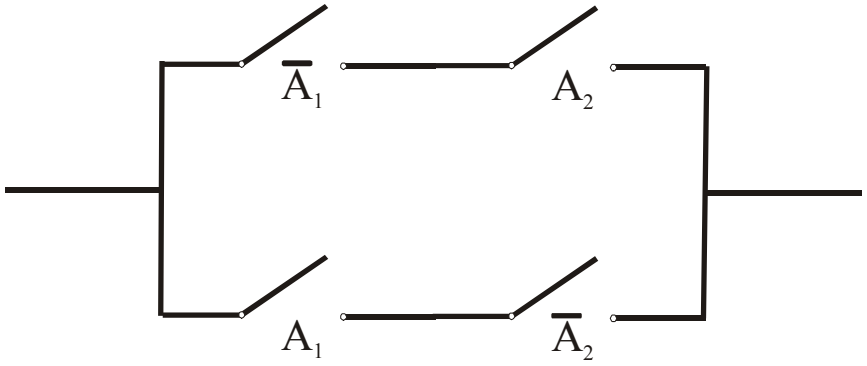
S_1	S_2	prúd
zapnuté	zapnuté	nie
zapnuté	vypnuté	áno
vypnuté	zapnuté	áno
vypnuté	vypnuté	nie

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

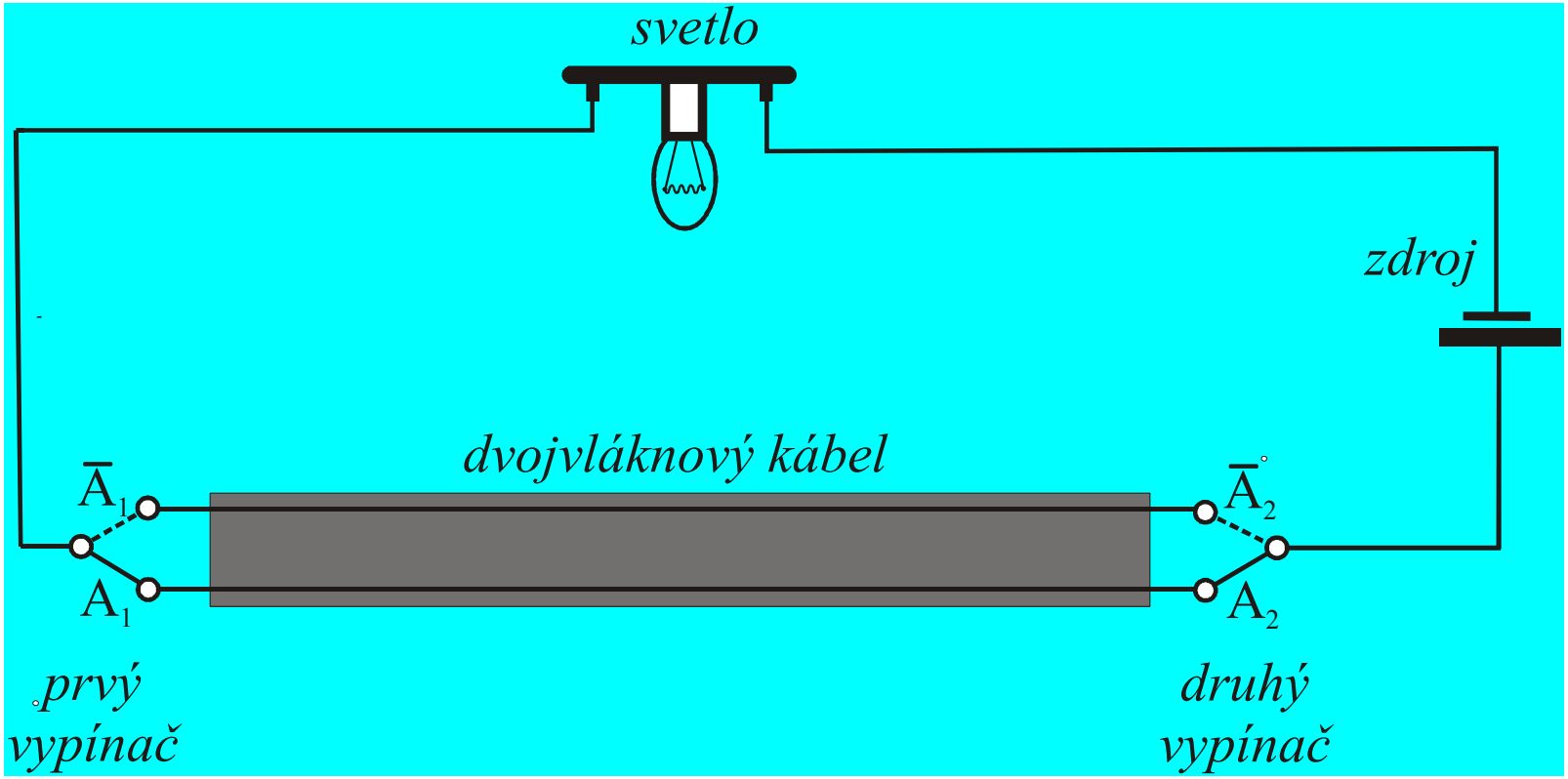
Tabuľka špecifikuje Boolovu funkciu v DNF

$$f\left(x_1, x_2\right) = \overline{x}_1 x_2 + x_1 \overline{x}_2$$

Spínací obvod so spínacou funkciou takto špecifikovanou má tvar



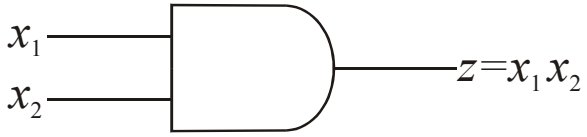
Realizácia tohto spínacieho obvodu, kde vytieňované oblasti tvoria stenové vypínače, ktoré sú spojené trojvláknovým káblom.



Logické obvody

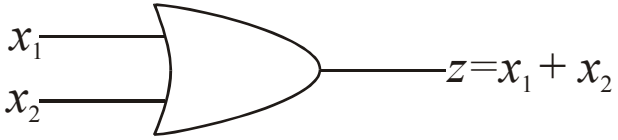
Logické brány

logická brána konjunkcie



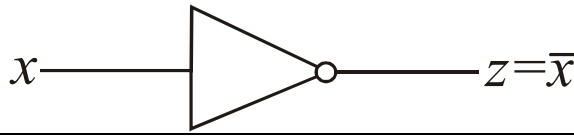
x_1	x_2	x_1x_2
0	0	0
0	1	0
1	0	0
1	1	1

logická brána disjunkcie



x_1	x_2	x_1+x_2
0	0	0
0	1	1
1	0	1
1	1	1

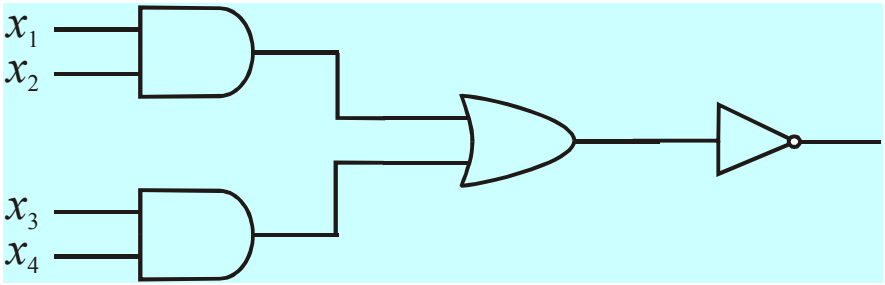
logická brána negácie



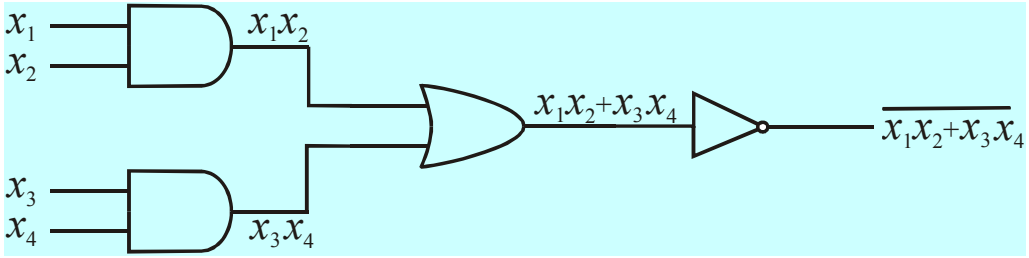
x_1	$\overline{x}_1$
0	1
1	0

Príklad

Zostrojte Boolovu funkciu pre logický obvod



Jednotlivé spoje tohto logického obvodu ohodnotíme takto



To znamená, že Boolova funkcia priradená tomuto obvodu má tvar

$$f\left(x_1, x_2, x_3, x_4\right) = \overline{x_1x_2 + x_3x_4}$$

Sumátor binárnych čísel

Prvá etapa spočíva v návrhu logického obvodu (nazývaného *polosumátor*) ktorý sčíta dve jednobitové čísla x a y

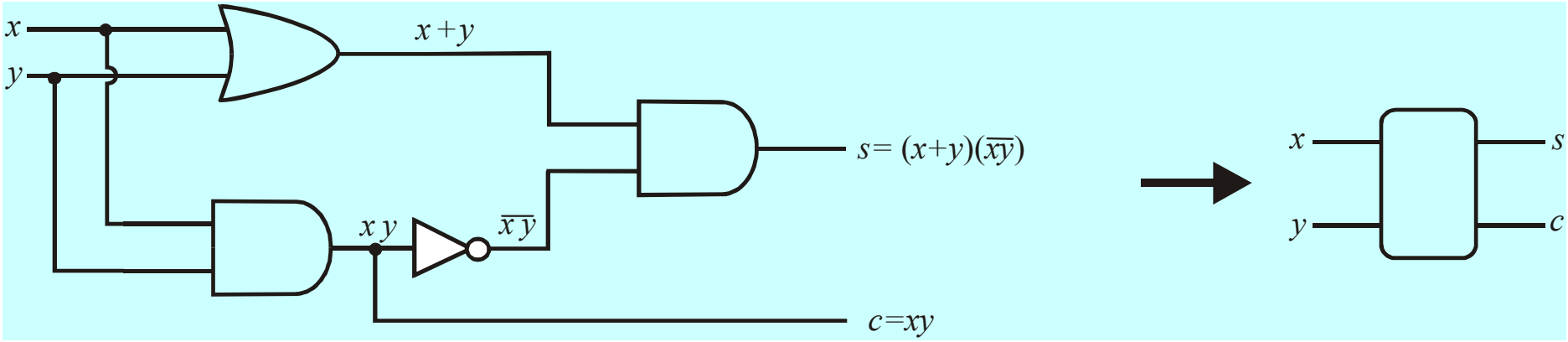
$$\begin{array}{r} x \\ y \\ \hline c\ \ s \end{array}$$

Uvedieme tabuľku všetkých prípadov tejto schémy, ktoré môžu nastať

vstup		výstup	
x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

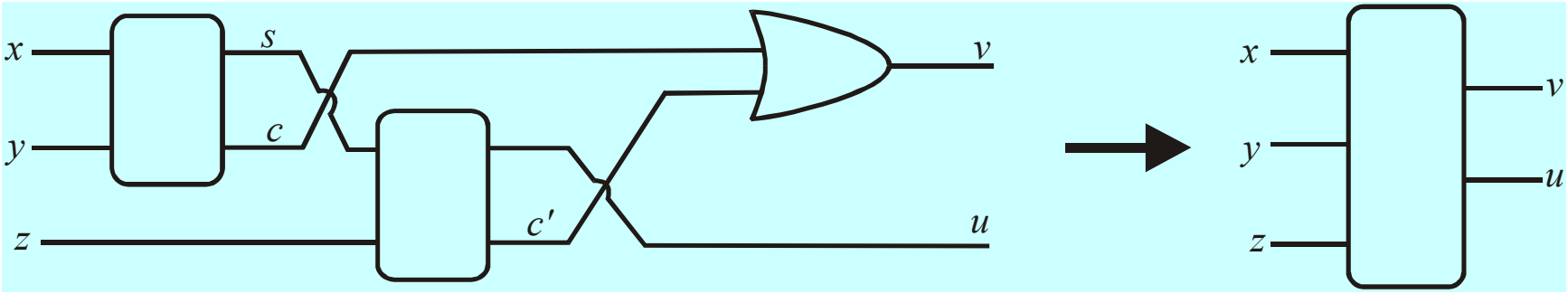
$$s = f(x,y) = \overline{x}y + x\overline{y} = (x+y)(\overline{x}+\overline{y}) = (x+y)\left(\overline{xy}\right)$$
$$c = g(x,y) = xy$$

kde pri konštrukcii alternatívnej pravej strany Boolovej funkcie f bol použitý distributívny zákon.



Druhá etapa konštrukcie sumátora spočíva v konštrukcii logického obvodu (nazývaného *dvojitý sumátor*) pre sčítanie troch jednobitových čísel

$$\begin{array}{r} x \\ y \\ \hline c\ s \\ z \\ \hline u\ v \end{array}$$



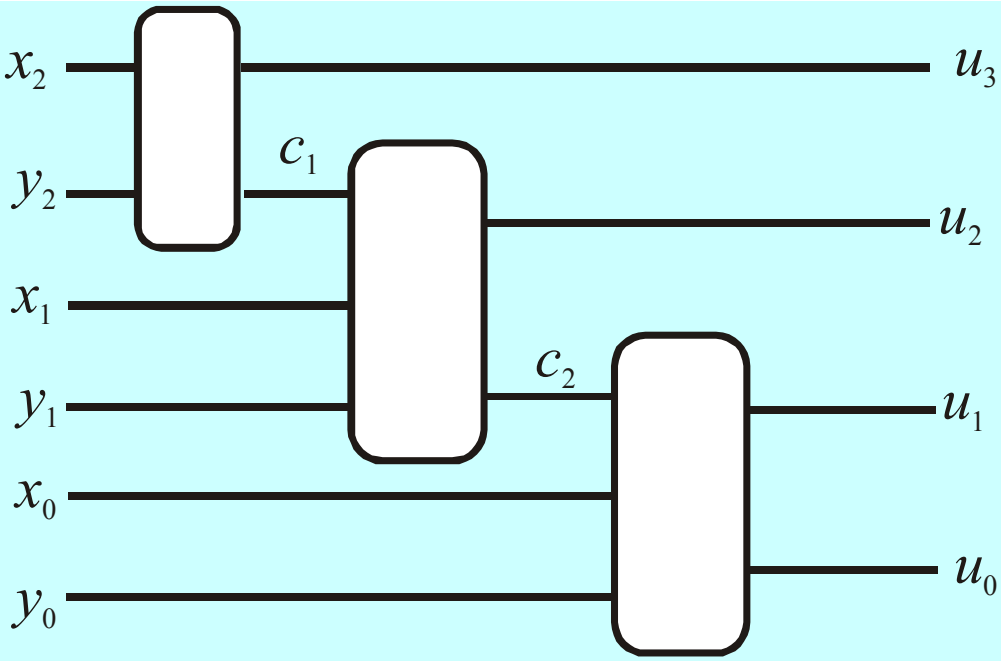
Tretia etapa využije dvojitý sumátor (ako blok) k realizácii sčítania dvoch bitový čísel ľubovolnej dĺžky *n*. Ako ilustračný príklad študujme sčítanie dvoch trojbitových čísel

$$\begin{array}{r} x_0\ x_1\ x_2 \\ y_0\ y_1\ y_2 \\ \hline u_0\ u_1\ u_2\ u_3 \end{array}$$

kde idúc postupne, z pravej do ľavej strany, uskutočňujeme sumáciu pomocou blokov polosumátora (*PS*) a dvojitéto sumátora (*DS*)

$$\begin{aligned} \left(u_3, c_1\right) &= PS\left(x_2, y_2\right) \\ \left(u_2, c_2\right) &= DS\left(x_1, y_1, c_1\right) \\ \left(u_0, u_1\right) &= DS\left(x_0, y_0, c_2\right) \end{aligned}$$

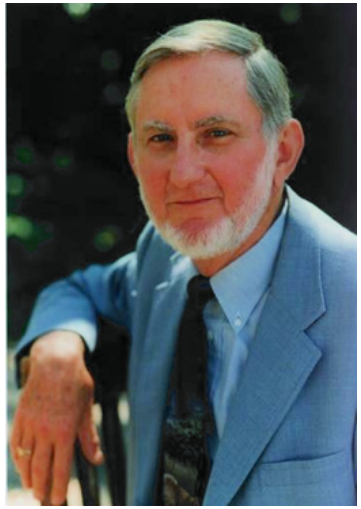
Túto postupnosť príkazov môžeme diagramaticky reprezentovať ako logický obvod s polosumátorom a dvoma dvojitými sumátormi



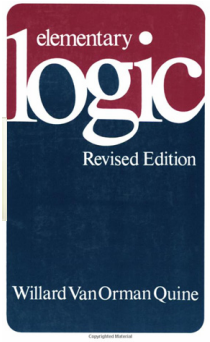
Pomocou blokov pre polosumátor a dvojitý sumátor môžeme zostrojiť logický obvod pre sumáciu dvoch digitálnych čísel ľubovolnej dĺžky.

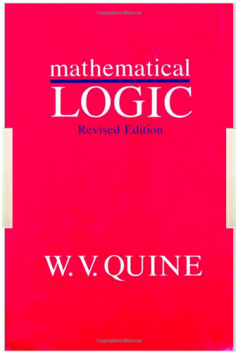
Minimalizácia Boolových výrazov pomocou Quine a McCluskey metódy

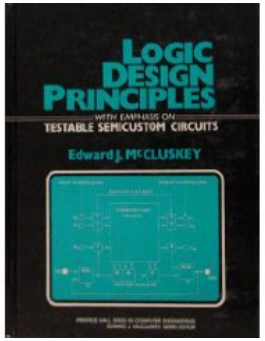
Willard Van Orman Quine (* 1908 – † 2000)



Edward J. McCluskey (*1929)







Ilustračný príklad

Quinovu a McCluskeyho metóda bude ilustrovaná konkrétnym prípadom optimalizácie jednoduchej Boolovej funkcie:

#	x	y	z	f
1	0	0	0	1
2	0	0	1	1
3	0	1	0	0
4	0	1	1	1
5	1	0	0	0
6	1	0	1	1
7	1	1	0	0
8	1	1	1	1

$$f\left(x,y,z\right)=\overline{x}\,\overline{y}\,\overline{z}+\overline{x}\,\overline{y}z+\overline{x}y\overline{z}+x\overline{y}\,\overline{z}+xyz$$

Každá klauzula môže byť reprezentovaná bitovým reťazcom $\boldsymbol{e} = (e_1, e_2, e_3) \in \{0,1\}^3$

$$l_{\boldsymbol{e}}(x, y, z) = x^{e_1} y^{e_2} z^{e_3} \rightarrow (e_1, e_2, e_3)$$

kde $\xi^e = \xi$, ak $e = 1$, $\xi^e = \overline{\xi}$, ak $e = 0$, pre $\xi = x, y, z$.

$$f(x, y, z) = (000) + (001) + (011) + (101) + (111)$$

Pre takto definovanú binárnu reprezentáciu môžeme použiť metriku ***Hammingovej vzdialenosti*** ku kvantifikácii podobnosti medzi binárnymi vektormi. Nech $\boldsymbol{e}_i = \left(e_1^{(i)}, e_2^{(i)}, ..., e_n^{(i)}\right)$ a $\boldsymbol{e}_j = \left(e_1^{(j)}, e_2^{(j)}, ..., e_n^{(j)}\right)$ sú dve binárne reprezentácie klauzúl, potom

$$d_H(\boldsymbol{e}_i, \boldsymbol{e}_j) = \sum_{k=1}^{n} \left| e_k^{(i)} - e_k^{(j)} \right|$$

Táto vzdialenosť pre binárne vektory nám špecifikuje počet polôh v ktorých sa binárne vektory vzájomne odlišujú.

Pre ilustračný príklad Boolova funkcia je reprezentovaná pomocou množiny 5 binárnych reťazcov dĺžky 3

$$U_f = \{(111),(101),(011),(001),(000)\}$$

Dve klauzuly z množiny U_f môžu byť vzájomne sčítané do jednej klauzuly vtedy a len vtedy ak sa líšia ich binárne reťazce práve v jednej polohe, čiže ak ich vzájomná Hammingova vzdialenosť sa rovná 1.

Proces sčítania klauzúl

Z množiny U_f vyberieme prvú a druhú klauzulu, ich binárne reprezentácie (111) a (101) sa líšia len hodnotou binárnej premennej v druhej polohe, $d_H(111,101)=1$. Tieto dve klauzuly sú sčítané takto

$$xyx + x\bar{y}z = x\underbrace{\left(y+\bar{y}\right)}_{1}z = xz$$

V binárnej reprezentácii tento proces zjednodušenia formálne vyjadríme takto

$$(1\underline{11})+(1\underline{01}) = sum\big((111),(101)\big) = (1\#1)$$

„*prázdny*“ symbol ′#′ reprezentuje prázdne miesto v binárnej reprezentácii novej klauzuly *xz*.

Nové klauzuly obsahujúce jeden symbol ′#′ tvoria množinu

$$U_f^{(1)} = \{(1\#1),(\#11),(\#01),(0\#1),(00\#)\}$$

V ďalšej etape vytvárame z množiny $U_f^{(1)}$ novú množinu $U_f^{(2)}$, ktorá obsahuje klauzuly s dvoma prázdnymi symbolmi ′#′ a ktoré boli vytvorené operáciou súčtu klauzúl z množiny $U_f^{(1)}$

$$U_f^{(2)} = \{(\#\#1)\}$$

Proces sčítania klauzúl obsahujúcich symboly ′#′ musí byť podrobnejšie špecifikovaný:

(a) Sčítať môžeme len také dve klauzuly, ktoré obsahujú rovnaký počet symbolov ′#′, pričom tieto symboly v oboch použitých klauzulách musia byť umiestnené v rovnakých polohách v oboch binárnych reprezentáciách.

(b) Klauzuly, ktoré vyhovujú podmienke (1) môžeme sčítať len vtedy, ak ich binárne komponenty sa líšia len v jednej polohe.

Zavedieme operátor $\mathcal{A}$, ktorý špecifikuje prechod množiny $U_f^{(k)}$ na množinu $U_f^{(k+1)}$ pomocou rekurentnej formuly

$$U_f^{(k+1)} = \mathcal{A}\left(U_f^{(k)}\right)$$

Rekurentná formula je inicializovaná množinou $U_f^{(0)} = U_f$. Musí existovať také kladné celé číslo *n*, že tento proces tvorby nových množín je ukončený, t. j. platí $U_f^{(n+1)} = \mathcal{A}\left(U_f^{(n)}\right) = \varnothing$.

- V 1. etape vytvoríme procesom sčítania dvoch klauzúl z množiny $U_f^{(0)} = U_f$ klauzuly s jedným prázdnym symbolom ´#´,
- v 2. etape vytvoríme z množiny $U_f^{(1)}$ klauzuly s dvoma symbolmi #.
- Tento rekurentný proces je ukončený vtedy, ak operátor $\mathcal{A}$ aplikovaný na množinu $U_f^{(n)}$ produkuje prázdnu množinu, t. j. $\mathcal{A}\left(U_f^{(n)}\right) = \varnothing$.

0. etapa			1. etapa			2. etapa		
1	(111)		1	(1,2)	(1#1)	1	(1,4)	(##1)
2	(101)		2	(1,3)	(#11)	2	(2,3)	(##1)
3	(011)		3	(2,4)	(#01)			
4	(001)		4	(3,4)	(0#1)			
5	(000)		5	(4,5)	(00#)			

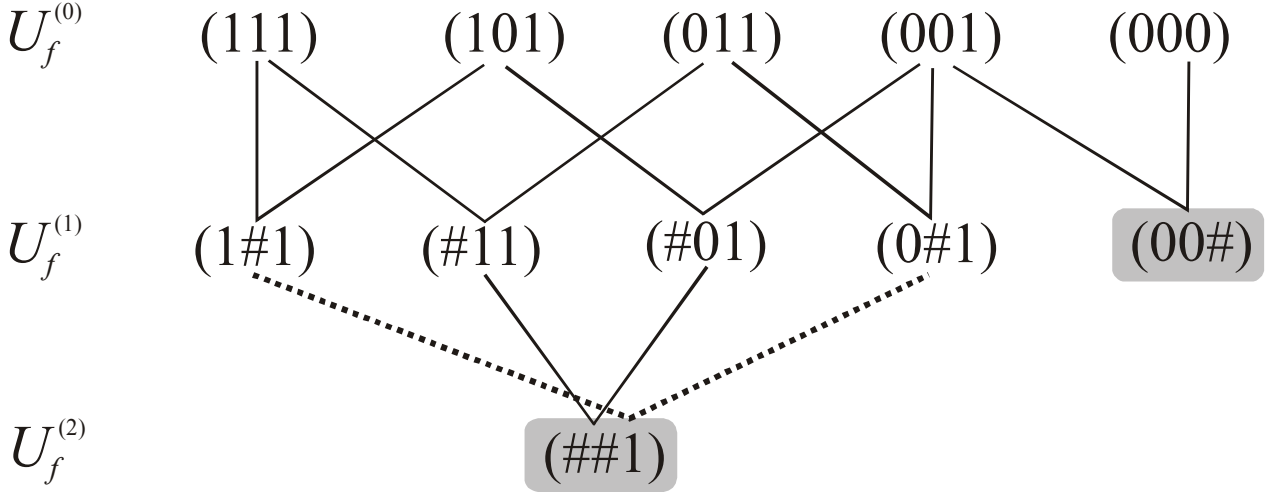
V stĺpcoch pre prvú a druhú etapu sú uvedené aj dvojice indexov klauzúl z predchádzajúceho stĺpca, ktoré boli použité v sumačnom procese.

Úloha: ako vybrať taký minimálny počet klauzúl zostrojených v prvej alebo v ďalších etapách, ktoré sú odvoditeľné zo všetkých pôvodných klauzúl z množiny $U_f^{(0)} = U_f$.

Množina klauzúl, ktorá vznikne zjednotením množín $U_f^{(0)}$, $U_f^{(1)}$, $U_f^{(3)}$,...

$$\tilde{U}_f = U_f^{(0)} \cup U_f^{(1)} \cup U_f^{(2)} \cup \ldots$$

je čiastočne usporiadaná a je znázornená Hasseho diagramom



Z tohto diagramu vyplýva, že má 5 maximálnych klauzúl (klauzuly patriace do množiny $U_f^{(0)}$) a dve minimálne klauzuly (##1) a (00#), ktoré sú na Hasseho diagrame vysvietené.

Úloha: Vybrať také minimálne klauzuly, ktoré pokrývajú pôvodné klauzuly z množiny $U_f^{(0)}$. Pre každú klauzulu $\boldsymbol{e} \in U_f^{(1)} \cup U_f^{(2)} \cup ...$, ktorá obsahuje aspoň jeden prázdny symbol, zostrojíme množinu

$$U(\boldsymbol{e}) = \left\{ \boldsymbol{e}' ; \left(\boldsymbol{e}' \in U_f \right) \wedge \left(\boldsymbol{e} \subseteq \boldsymbol{e}' \right) \right\} \subseteq U_f$$

ktorá obsahuje všetky pôvodné klauzuly (neobsahujúce prázdne symboly #), ktoré sú pokryté klauzulou ***e***

Nech množina minimálnych klauzúl je označená V, potom hľadáme takú jej podmnožinu $\hat{V} \subseteq V$, ktorej klauzuly plne pokrývajú množinu $U_f^{(0)}$

$$\bigcup_{e \in \hat{V}} U(e) = U_f$$

Podmnožina $\hat{V}$ je určená podmienkou minimálnosti počtu literálov, $[\hat{V}]$, ktoré obsahuje

$$\hat{V} = arg \min_{V' \subseteq V} [V']$$

- Riešenie tohto optimalizačného problému je pre malý počet premenných (cca do päť) obvykle zvládnuteľný ***ručne*** tak, že preberieme všetky možnosti, ktoré pokrývajú množinu U_f.
- Pre väčšie problémy môže byť použitá metóda ***spätného prehľadávania***, ktorá systematicky preskúma všetky možnosti.
- Žiaľ, tento prístup je nepoužiteľný pre niekoľko desiatok premenných, v dôsledku exponenciálneho rastu zložitosti. Potom nastupujú ***evolučné metódy***, ktoré poskytujú v reálnom čase kvalitné suboptimálne riešenie, ktoré je často rovné optimálnemu riešeniu.

V tomto konkrétnom prípade sa jedná o jednoduchý problém, musíme vybrať obe minimálne klauzuly (##1) a (00#), ktoré pokrývajú pôvodné klauzuly.

Potom môžeme písať Boolovu funkciu

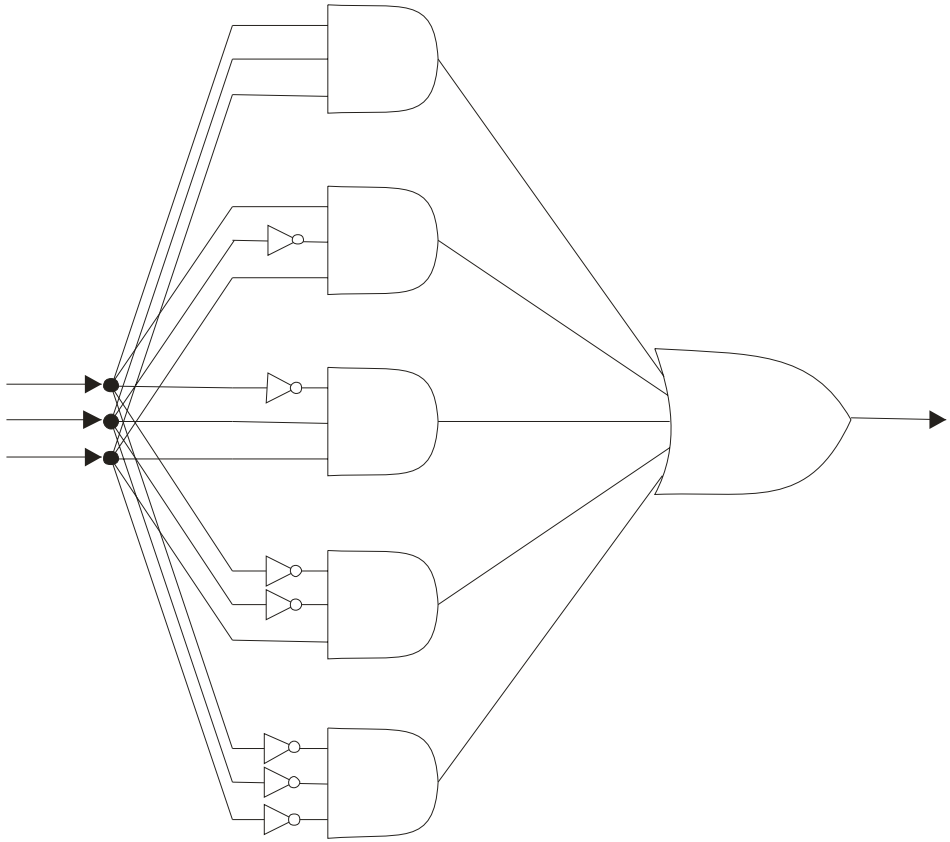
$$f\left(x,y,z\right)=xyz+x\overline{y}z+\overline{x}yz+\overline{x}\,\overline{y}z+\overline{x}\,\overline{y}\,\overline{z}$$

v ekvivalentnom tvare

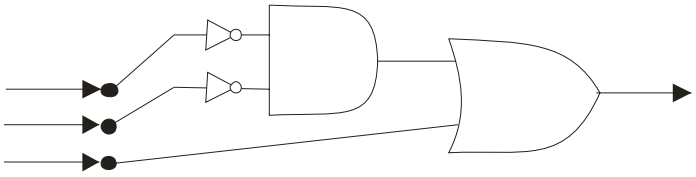
$$f\left(x,y,z\right)=z+\overline{x}\,\overline{y}$$

čo reprezentuje podstatné zjednodušenie (optimalizáciu) pôvodnej Boolovej funkcie (7.39), ktorej počet literálov z 15 klesol na 3.

$$f(x,y,z) = xyz + x\bar{y}z + \bar{x}yz + \bar{x}\bar{y}z + \bar{x}\bar{y}\bar{z}$$



$$f(x,y,z) = z + \bar{x}\,\bar{y}$$



Príklad

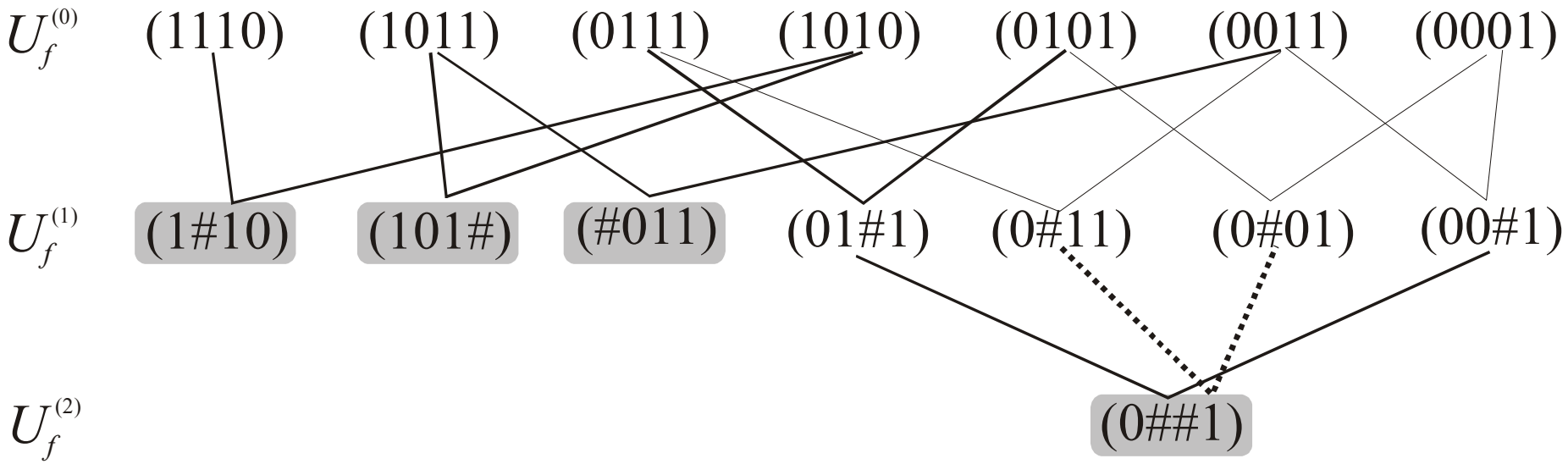
Uvažujme Boolovu funkciu

$$f\left(w,x,y,z\right) = wxy\bar{z} + w\bar{x}yz + \bar{w}xyz + w\bar{x}y\bar{z} + \bar{w}x\bar{y}z + \bar{w}\bar{x}yz + \bar{w}\,\bar{x}\,\bar{y}z$$

V nasledujúcej tabuľke je znázornený postup vytvárania všetkých možných sumácií medzi klauzulami (v binárnej reprezentácii) k tejto Boolovej funkcie

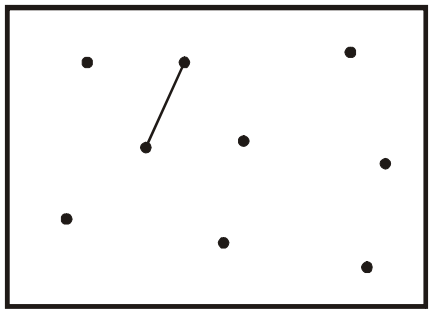
0. etapa			1. etapa				2. etapa		
1	(1110)		1	(1,4)	(1#10)		1	(4,7),(5,6)	(0##1)
2	(1011)		2	(2,4)	(101#)				
3	(0111)		3	(2,6)	(#011)				
4	(1010)		4	(3,5)	(01#1)				
5	(0101)		5	(3,6)	(0#11)				
6	(0011)		6	(5,7)	(0#01)				
7	(0001)		7	(6,7)	(00#1)				

Hasseho diagram

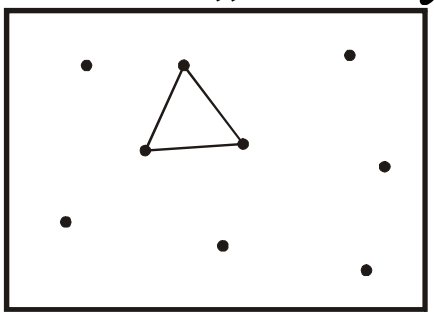


Teraz stojíme pred problémom ako vybrať taký minimálny počet klauzúl, ktoré nám budú pokrývať celú pôvodnú množinu klauzúl.

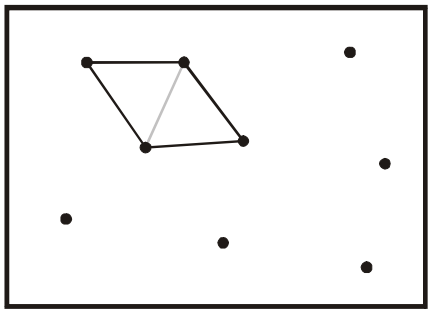
„Greedy“ metóda



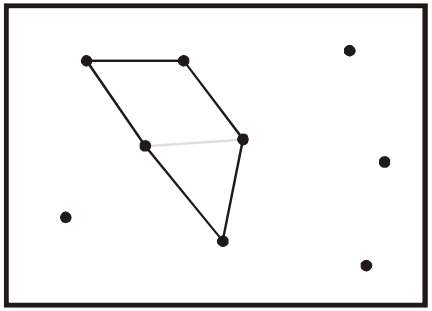
1. *krok*



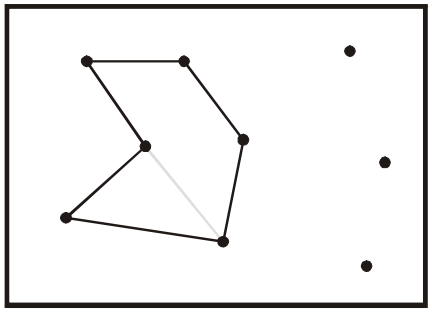
2. *krok*



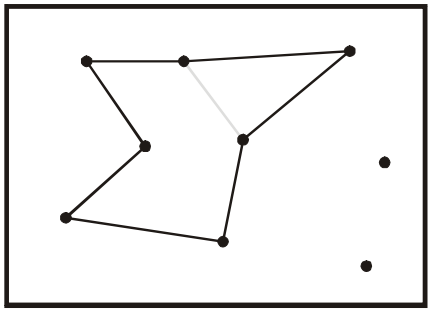
3. *krok*



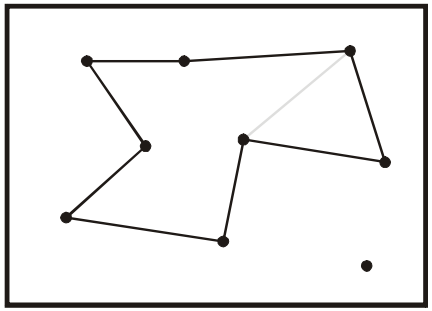
4. *krok*



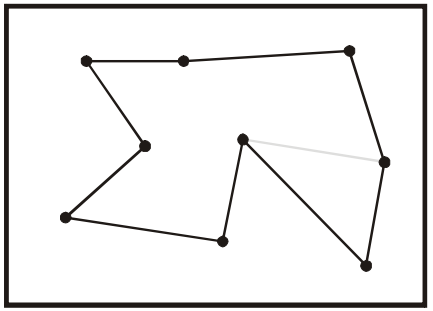
5. *krok*



6. *krok*



7. *krok*



8. *krok* (*konečné riešenie*)

Použitie jednoduchej “greedy” heuristiky pre konštrukciu cesty obchodného cestujúceho. Algoritmus je inicializovaný trojuholníkom, ktorý je zostrojený z najkratšej hrany doplnenej o najbližší vrchol. V ďalšom kroku nájdeme voľný vrchol x a hranu (y,z) tak, aby veličina $d(x,y)+d(x,z)-d(y,z)$ bola minimálna. Aktuálny cyklus rozšírime o hrany (x,y) a (x,z), pričom pôvodnú hranu (y,z) odstránime. Tento proces opakujeme tak dlho, až

každý vrchol je zapojený do vytvoreného cyklu. Naše simulačné výpočty naznačujú, že takto vytvorená uzavretá cesta obchodného cestujúceho poskytuje kvalitné suboptimálne riešenie (ktoré v mnohých prípadoch je totožné s optimálnym riešením).

V prvom kroku vyberieme takú minimálnu klauzulu, ktorá pokrýva najväčší počet maximálnych klauzúl. V prípade, že existuje niekoľko rovnocenných minimálnych klauzúl, ktoré pokrývajú rovnaký počet maximálnych klauzúl, tak vyberieme takú minimálnu klauzulu, ktorá má minimálny počet literálov. V tomto prvom kroku vyberieme minimálnu klauzulu (0##1), ktorá pokrýva štyri maximálne klauzuly. Táto možnosť je znázornená na nasledujúcej tabuľke:

minimálne klauzuly	maximálne (pôvodne) klauzuly						
	(1110)	(1011)	(0111)	(1010)	(0101)	(0011)	(0001)
(1#10)	1			1			
(101#)		1		1			
(#011)		1				1	
(0##1)			1		1	1	1

V druhom kroku máme dve alternatívne možnosti, a to výber minimálnej klauzuly (1#10) alebo výber minimálnej klauzuly (101#). V oboch prípadoch tieto mimimálne klauzuly pokrývajú dve maximálne klauzuly, pretože majú rovnaký počet literálov, tak sú rovnocenné. Predpokladajme, že vyberieme prvú možnosť, potom predchádzajúca tabuľka má tvar:

minimálne klauzuly	maximálne (pôvodne) klauzuly						
	(1110)	(1011)		(1010)			
(1#10)	1			1			
(101#)		1		1			
(#011)		1					
(0##1)							

Ak z tejto tabuľky odstránime vybranú klauzulu, potom sa tabuľka ďalej zjednoduší do tvaru:

minimálne klauzuly	maximálne (pôvodne) klauzuly						
		(1011)					
(1#10)							
(101#)		1					
(#011)		1					
(0##1)							

Na záver ***v tret'om kroku*** máme dve rovnocenné možnosti výberu minimálnych klauzúl (101#) resp. (#011), vybrali sme prvú možnosť (101#), tým sme dokázali, že sedem maximálnych klauzúl môže byť pokrytých pomocou troch minimálnych klauzúl (0##1), (1#10) a (101#), ktoré majú dohromady osem literálov. Ekvivalentná (minimálna) Boolova funkcia, ktorá je určená týmito klauzulami má tvar

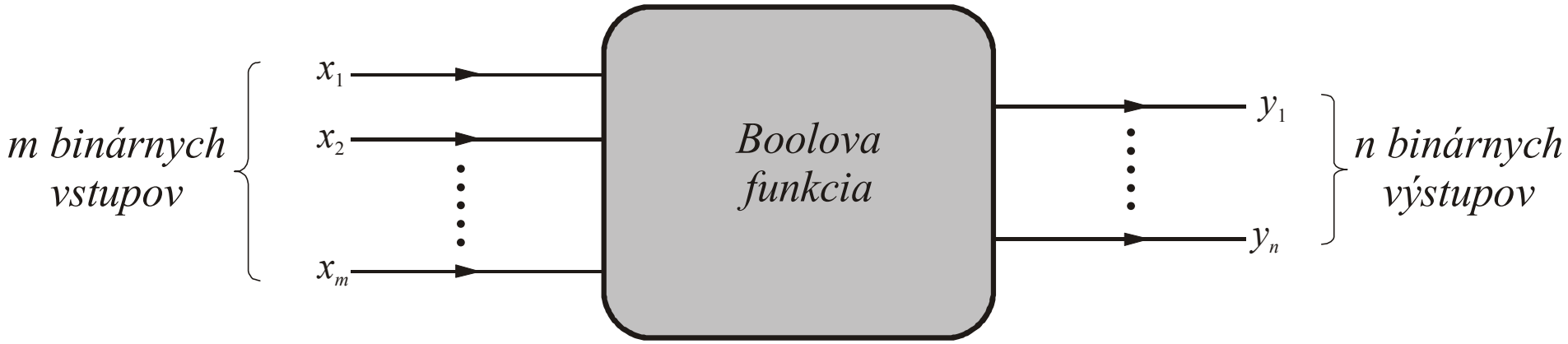
$$f_1(w,x,y,z) = \overline{w}z + wy\overline{z} + w\overline{x}y$$

Ak by sme vybrali druhú alternatívnu možnosť (#011), potom alternatívny tvar minimálnej Boolovej funkcie je

$$f_2(w,x,y,z) = \overline{w}z + wy\overline{z} + \overline{x}yz$$

Optimalizácia Boolovej funkcie s *viacerými* funkčnými hodnotami

$$f : \{0,1\}^m \rightarrow \{0,1\}^n, \text{ kde } n \geq 2$$



Poznámka: Optimalizačná metóda Q-M sa ľahko zovšeobecní aj pre tento prípad. Jednotlivé členy Boolovej funkcie budú označené horným indexom výstupu, ktorý špecifikuje.

Tabuľka špecifikujúca Boolovu funkciu

#	x_1	x_2	x_3	x_4	f_A	f_B
1	0	0	0	0	0	0
2	0	0	0	1	1	1
3	0	0	1	0	1	1
4	0	0	1	1	0	0
5	0	1	0	0	0	0
6	0	1	0	1	1	1
7	0	1	1	0	0	0
8	0	1	1	1	0	0

#	x_1	x_2	x_3	x_4	f_A	f_B
9	1	0	0	0	0	0
10	1	0	0	1	0	0
11	1	0	1	0	1	1
12	1	0	1	1	0	0
13	1	1	0	0	0	0
14	1	1	0	1	0	0
15	1	1	1	0	1	1
16	1	1	1	1	1	0

Vybrané riadky s jednotkovou funkčnou hodnotou

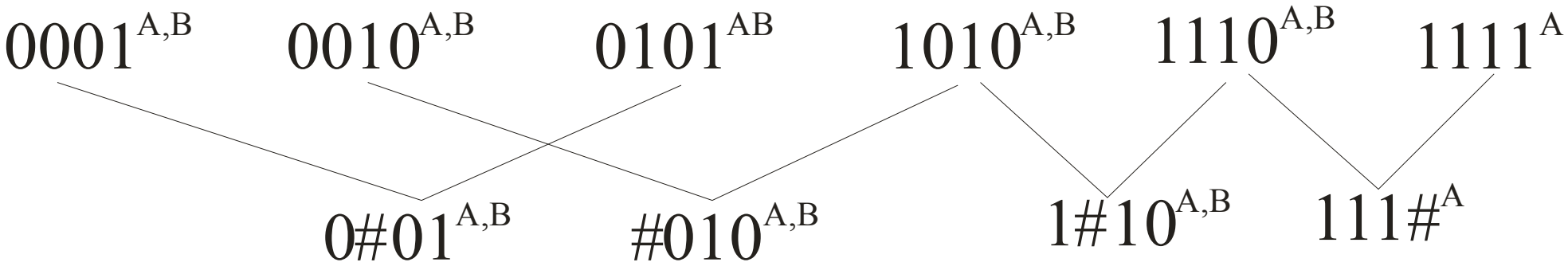
#	x_1	x_2	x_3	x_4	f_X
2	0	0	0	1	*A,B*
3	0	0	1	0	*A,B*
6	0	1	0	1	*A,B*
11	1	0	1	0	*A,B*
15	1	1	1	0	*A,B*
16	1	1	1	1	*A*

#	r_i - r_j	x_1	x_2	x_3	x_4	f_X
1	2 - 6	0	#	0	1	*A,B*
2	3 - 11	#	0	1	0	*A,B*
3	11 - 15	1	#	1	0	*A,B*
4	15 - 16	1	1	1	#	*A*

Poznámka: Budeme vyberať do procesu „súčtu“ len také dva riadky, ktoré

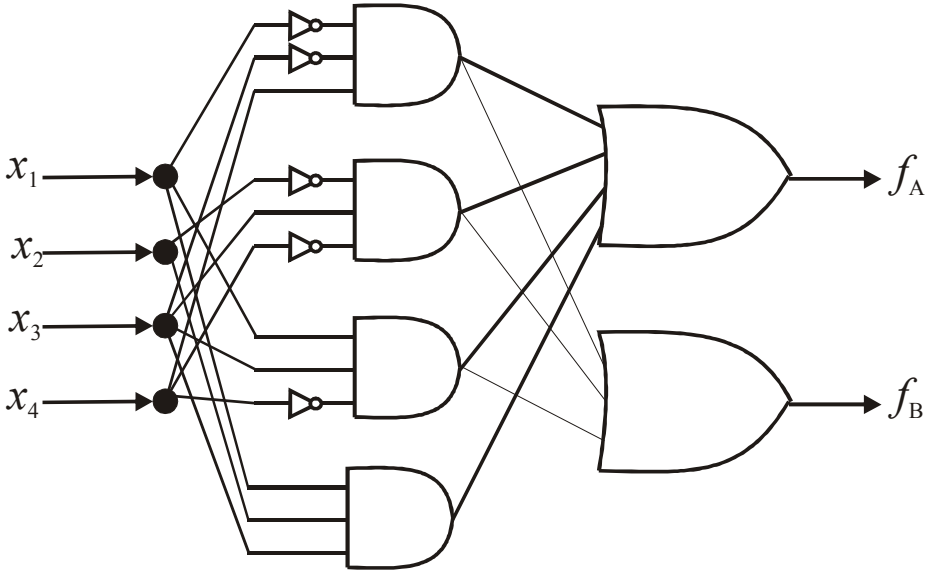
(1) majú jednotkovú Hammingovu vzdialenosť,

(2) prienik indexov funkčných hodnôt je neprázdna množina, pričom výsledný súčet je označený týmto prienikom.

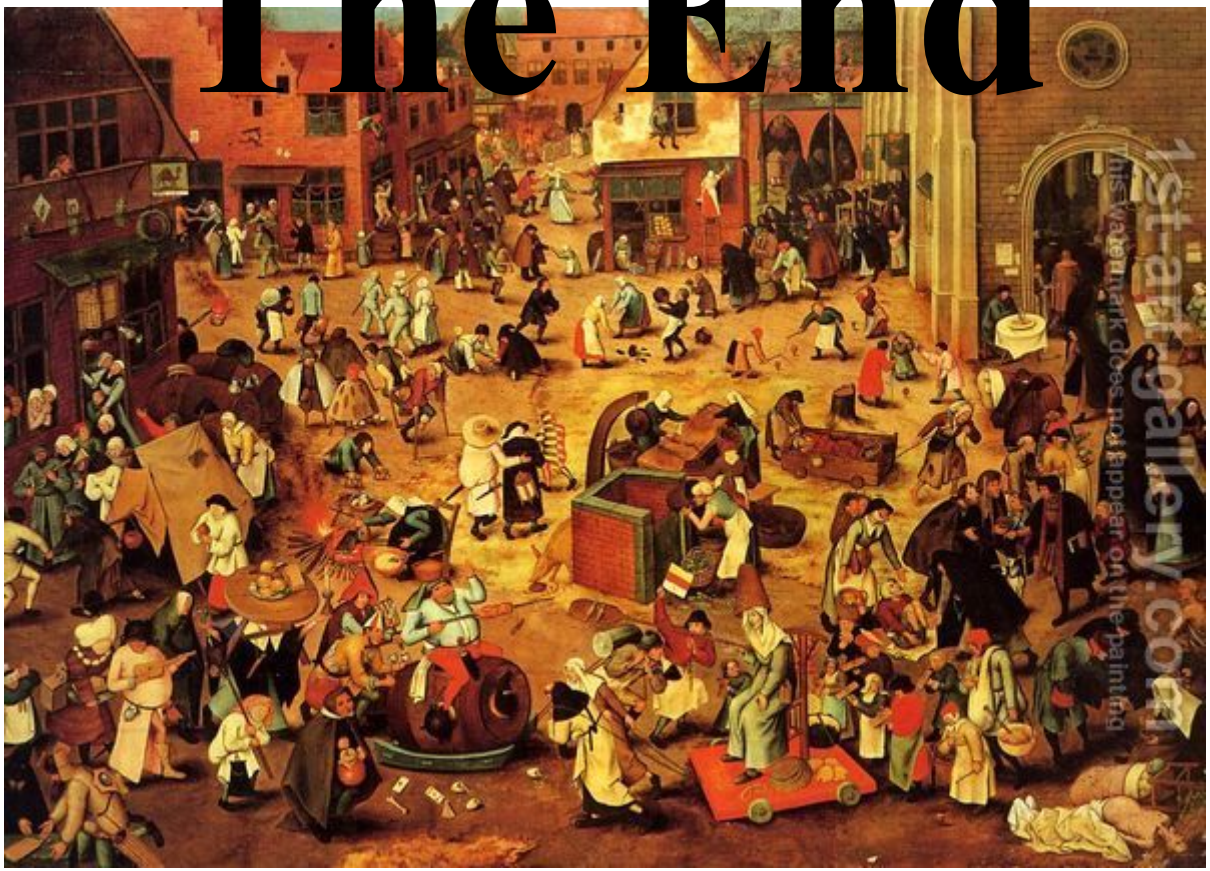


$$f_A = 0\#01 + \#010 + 1\#10 + 111\#$$

$$f_B = 0\#01 + \#010 + 1\#10$$



The End



Prvé znázornenie metafory multiagentové systému holandským maliarom Pieter Bruegel Sr. (1525-1569) olejomaľbou „*Batle between Lent and Carnival* “