Princípy počítačov

Daniel Olejár

9. októbra 2013
Verzia 2.0

Obsah

Predhovor

Tento text vychádza obsahovo z prednášok z predmetu Princípy počítačov, ktoré odzneli v zimnom semestri školského roka 2012/13 pre študentov odboru Informatika na FMFI UK. Prednášky boli v tomto dokumente doplnené o ďalšie rozširujúce informácie, ktoré sa z časových dôvodov nedali odprednášať a námety pre samostatné štúdium. Texty sú rozdelené do tématických celkov, ktoré však nutne nezodpovedajú jednotlivým prednáškam. V každej kapitole je uvedené, čo sú základné a čo rozširujúce informácie a čo by si študent mal povinne osvojiť. (Časom budú jednotlivé kapitoly doplnené o úlohy, ktorých riešením by študent sa študent mal naučiť používať získané vedomosti a hlbšie pochopiť problematiku). Do pripravovanej knižky sú zaradené dve kapitoly, ktoré vznikli úpravou textov dvoch iných našich kníh - základné pojmy sme prebrali z Teórie kódovania a booleovské funkcie z Úvodu do diskrétnych štruktúr. Úpravy ešte nie sú definitívne, pretože aktuálnym cieľom bolo rýchle poskytnúť text na prípravu na skúšku a dolaďovanie obsahu a štylistiky ponechávame na neskoršie obdobie.

Kapitola 1

Základné pojmy a označenia

V prednáške budeme používať mnohé pojmy, ktoré sa stali súčasťou bežného jazyka a ľudia ich často používajú bez toho, aby si uvedomovali ich presný význam. V bežnej komunikácii to natoľko neprekáža, ale pri odbornom výklade by rozličná interpretácia základných pojmov mohla viesť k nedorozumeniam. Aby sme sa v ďalšom výklade vyhli zbytočným nedorozumeniam, vybudujeme exaktne potrebný pojmový aparát.

1.1 Abecedy, slová a jazyky

Abeceda je ľubovoľná konečná neprázdna množina. Prvky abecedy budeme nazývať *znakmi* alebo *symbolmi*. Abecedu budeme označovať symbolom Σ; ak bude potrebné rozlišovať rozličné abecedy, budeme symbol Σ indexovať ($\Sigma_1, \Sigma_2, \dots$). Ľubovoľná konečná postupnosť znakov z abecedy Σ sa nazýva *slovom nad abecedou* Σ. Ak nebude podstatné o akú abecedu ide alebo z kontextu bude známe, o ktorú abecedu sa jedná, budeme kvôli stručnosti slová nad abecedou Σ vynechávať. Zjednodušíme aj zapisovanie slov; symboly v postupnosti nebudeme oddeľovať čiarkami a slovo (napr.) a, b, e, c, e, d, a budeme zapisovať v tvare, ako sa slová v textoch štandardne zapisujú; t.j. $abeceda$. Nech je w slovo nad abecedou Σ, potom počet znakov slova w nazveme *dĺžkou slova* w. Dĺžku slova w budeme označovať symbolom $l(w)$. Tak napríklad $l(slovo) = 5$, $l(abeceda) = 7$, $l(a) = 1$. Postupnosť znakov nad abecedou Σ môže byť aj prázdna. Takáto postupnosť sa nazýva *prázdnym slovom* a označuje sa symbolom ε. Pre dĺžku prázdneho slova platí $l(\varepsilon) = 0$. Teraz definujeme operácie nad slovami, pomocou ktorých bude možné zo známych slov vytvárať nové slová. Nech sú u, v dve slová nad abecedou Σ; $u = a_1 \dots a_n; \quad v = b_1 \dots b_m$. *Zreťazením slov* u, v je slovo $w = uv = a_1 \dots a_n b_1 \dots b_m$ nad abecedou Σ. (Je zrejmé, že operácia zreťazovania slov je asociatívna, ale vo všeobecnosti nie je komutatívna; prázdne slovo ε je obojstranným neutrálnym prvkom vzhľadom na operáciu zreťazenia slov: pre ľubovoľné slovo w platí $w\varepsilon = \varepsilon w = w$). Slovo možno zreťaziť aj so sebou samým, napr. $uu = a_1 \dots a_n a_1 \dots a_n$. Pre ľubovoľné slovo w a ľubovoľné číslo $k \in \mathcal{N}$ definujeme:

1. $w^0 = \varepsilon$,
2. $w^{k+1} = w^k w$.

Nech $u = a_1 \dots a_n$ je ľubovoľné slovo, potom súvislú podpostupnosť $z = a_i a_{i+1} \dots a_{i+k-1}$; $1 \leq i, i+k < n$ nazveme *podslovom slova* u. Ak $0 < k < n$, slovo z nazveme *vlastným podslovom slova* u. Slová u a ε sú triviálnymi podslovami slova u a v kódovaní sa nimi zvlášť zaoberať nebudeme. Zato však v teórii kódovania zohrávajú dôležitú úlohu podslová, ktoré sú začiatkom alebo koncom nejakého slova. Zavedieme pre ne špeciálne pomenovania. Nech $u = a_1 \dots a_n$ je ľubovoľné slovo, slovo $z = a_1 \dots a_k, \quad 0 < k \leq n$ nazveme *počiatočným podslovom (prefixom)* slova u a slovo $x = a_i a_{i+1} \dots a_n; \quad 1 \leq i = n$ nazveme *koncovým podslovom (sufixom) slova* u. Znaky v slove možno aj preusporiadať. Dôležitým prípadom preusporiadania znakov je otočenie slova: *zrkadlovým obrazom* slova $u = a_1 \dots a_n$ nazveme slovo $u^R = a_n \dots a_1$.

Slová môžeme zoskupovať do množín. Takéto množiny slov budeme nazývať jazykmi. Presnejšie, ľubovoľnú množinu slov nad abecedou Σ nazveme *jazykom nad abecedou* Σ. Keďže jazyky sú množiny slov, možno z existujúcich jazykov vytvárať nové jazyky pomocou množinových operácií, ako sú zjednotenie, doplnok, rozdiel, prienik, symetrická diferencia množín a prípadne iné. Pre slová sme zaviedli operáciu zreťazovania (slov). Zavedieme teraz užitočné operácie s jazykmi, ktoré sú založené na zreťazovaní slov. Nech sú $\mathcal{L}_1, \mathcal{L}_2$ jazyky nad abecedou Σ, potom $\mathcal{L} = \mathcal{L}_1\mathcal{L}_2$ je jazyk nad abecedou Σ definovaný nasledovne: $\mathcal{L} = \{uv;\ u \in \mathcal{L}_1, v \in \mathcal{L}_2\}$. Jazyk možno zreťazovať so sebou samým; pre ľubovoľný jazyk $\mathcal{L}$ a ľubovoľné číslo $k \in \mathcal{N}$ definujeme:

1. $\mathcal{L}^0 = \{\varepsilon\}$,
2. $\mathcal{L}^{k+1} = \mathcal{L}^k\mathcal{L}$.

Na záver uvedieme ešte dve operácie nad jazykmi, ktoré nám umožnia popísať množinu všetkých možných slov, ktoré sa dajú vytvoriť pomocou operácie zreťazovania jazyka. Nech $\mathcal{L}$ je ľubovoľný jazyk, potom jazyky $\mathcal{L}^+ = \bigcup_{i>0}^{\infty} \mathcal{L}^i$ a $\mathcal{L}^* = \bigcup_{i\geq 0}^{\infty} \mathcal{L}^i$ sa nazývajú *kladná,* resp. *nezáporná iterácia jazyka* $\mathcal{L}^i$. Všimnite si, že abecedu Σ možno chápať aj ako jazyk pozostávajúci zo všetkých slov dĺžky 1 nad abecedou Σ a Σ^* predstavuje množinu všetkých slov nad abecedou Σ.

Ilustrujeme zavedené pojmy na príkladoch.

Príklad.

1. Binárna abeceda Σ_1 je ľubovoľná dvojprvková množina. Znaky binárnej abecedy najčastejšie označujeme číslicami $0, 1$; binárnu abecedu budeme v tomto prípade chápať ako množinu $\Sigma_1 = \{0, 1\}$.
2. Na zápis prirodzených čísel vystačíme s abecedou $0, 1$; $\Sigma_2 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
3. Racionálne čísla možno zapísať v podobe slov nad abecedou $\Sigma_3 = \Sigma_2 \cup \{"+", "-", "."\}$.
4. $\Sigma_4 = \{a, b, c, d, e, f, g, i, j, k, l, m, n, o, p, q, r, s, t, v, w, x, y, z\}$ je abeceda pozostávajúca z malých písmen anglickej abecedy.

5. Abecedu $\Sigma_5 = \{A, B, C, D, E, F, G, I, J, K, L, M, N, O, P, Q, R, S, T, V, W, X, Y, Z\}$ tvoria veľké písmená anglickej abecedy.

6. $\Sigma_6 = \Sigma_4 \bigcup \Sigma_5$.

7. Abecedu $\Sigma_7 = \{\alpha, \beta, \gamma, \delta, \epsilon, \varepsilon, \zeta, \eta, \theta, \vartheta, \iota, \kappa, \lambda, \mu, \nu, \xi, o, \pi, \varpi, \rho, \varrho, \sigma, \varsigma, \tau, \upsilon, \phi, \varphi, \chi, \psi, \omega\}$ tvoria malé písmená gréckej abecedy.

8. Ďalšími užitočnými abecedami by mohli byť rozličné znakové sady, napr. všetky znaky kódov ASCII. V teórii kódovania budeme často pracovať s abecedami, ktorých symboly sú prvkami konečných polí. Tieto symboly budeme zapisovať pomocou prirodzených čísel; $\Sigma_8 = Z_m = \{0, 1, \ldots m-1\}$.

9. Slovo 2.78128 je slovom nad Σ_3, ale nie je slovom nad abecedou Σ_2 (pretože obsahuje symbol ".", ktorý sa v abecede Σ_2 nenachádza).

10. Zreťazením slov $w_1 = pismeno$ a $w_2 = male$ dostávame slová (napr. nad abecedou Σ_4) $w_1w_2 = pismenomale$ a $w_2w_1 = malepismeno$.

11. Nech je dané slovo $w_1 = pismeno$ nad abecedou Σ_4, počiatočné a koncové podslová tohto slova sú uvedené v nasledujúcej tabuľke:

prefix	sufix
ε	pismeno
p	ismeno
pi	smeno
pis	meno
pism	eno
pisme	no
pismen	o
pismeno	ε

12. Nech je dané slovo $w_1 = pismeno$ nad abecedou Σ_4, zrkadlový obraz slova w_1 je slovo $w_1^R = onemsip$ nad abecedou Σ_4.

13. Nech sú $\mathcal{L}_1 = \{ne, pre, po, vy\}$, $\mathcal{L}_2 = \{mysli, hovor, pis, padni\}$ jazyky nad abecedou Σ_4. Jazyk $\mathcal{L}_1\mathcal{L}_2$ je uvedený v nasledujúcej tabuľke:

$\mathcal{L}_1/\mathcal{L}_2$	ne	pre	po	vy
mysli	nemysli	premysli	pomysli	vymysli
hovor	nehovor	prehovor	pohovor	vyhovor
pis	nepis	prepis	popis	vypis
padni	napadni	prepadni	popadni	vypadni

14. Uvažujme binárnu abecedu $\Sigma_1 = \{0, 1\}$. Uvedieme množiny slov Σ_1^k pre niekoľko

počiatočných hodnôt k:

k	Σ_1^k
0	$\{\varepsilon\}$
1	$\{0, 1\}$
2	$\{00, 01, 10, 11\}$
3	$\{000, 001, 010, 011, 100, 101, 110, 111\}$
4	$\{0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111\}$
5	$\{00000, 00001, 00010, 00011, 00100, 00101, 00110, 00111, 01000, 01001, 01010, 01011, 01100, 01101, 01110, 01111, 10000, 10001, 10010, 10011, 10100, 10101, 10110, 10111, 11000, 11001, 11010, 11011, 11100, 11101, 11110, 11111\}$

1.2 Údaje, informácia a komunikácia

V bežnom živote sa pojem informácie používa voľne a v rozličných významoch; hovorí sa o rozličných druhoch informácie (obrazová, knižná, zvuková, genetická, novinová) a informácii sa pripisujú rozličné atribúty (overená, čerstvá, aktuálna, pochybná, škandalózna a i.) V teórii kódovania nás nebude zaujímať pôvod, význam ani hodnotenie informácie; jediné čo pre nás bude podstatné je množstvo informácie. Budeme pracovať s údajmi a správami, ktoré budú obsahovať nejakú informáciu, tieto údaje budeme spracovávať a budeme sa snažiť nájsť pre zápis informácie, ktorú údaje obsahujú formu ktorá je z hľadiska spracovania údajov/informácie najvhodnejšia.

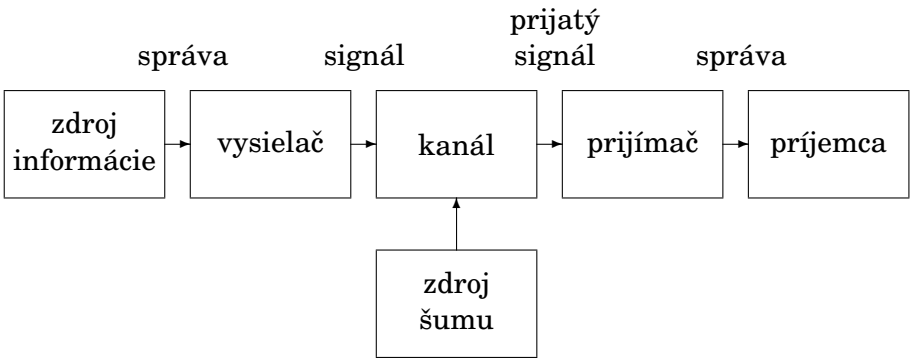
V hovorovom jazyku sa pojmy informácia, správa a údaje chápu ako synonymá; ukážeme, že tieto pojmy majú odlišný význam. Ilustrujeme rozdiel medzi pojmami údaje a informácia na príklade. Predstavme si

1. binárny reťazec 0000000000001010,
2. výraz $0^{12}(01)^2$,
3. slovné spojenie „dvanásť núl, jednotka, nula, jednotka, nula“,
4. 10,
5. A,
6. 000A.

Vo všetkých prípadoch ide o jednoznačné určenie tej istej binárnej postupnosti dĺžky 16; v prvom prípade je popisom explicitné vymenovanie členov postupnosti, v druhom jej zápis pomocou regulárneho výrazu, v treťom slovný popis vo štvrtom vyjadrenie číselnej hodnoty binárneho čísla v desiatkovej sústave (predpokladáme, že informácia o dĺžke slova je známa), v piatom ide o hexadecimálny zápis toho istého čísla (s vynechaním počiatočných núl) a napokon posledný výraz je hexadecimálny zápis binárneho reťazca, vrátane prvých troch nulových hexadecimálnych číslic. Všetky popisy majú spoločné to, že umožňujú v množine všetkých binárnych reťazcov (v našom prípade dĺžky

16) jednoznačne identifikovať daný reťazec; t.j. obsahujú rovnakú informáciu. *Údaje* teda predstavujú záznam informácie; *informácia* je obsahom údajov. Niekedy sa pojem informácia spája aj so sémantikou (významom) údajov, ale toto spojenie chápanie informácie komplikuje, pretože do pojmu informácia zavádza subjektívny aspekt (toho, kto údaje interpretuje a kontext). Preto sa budeme pridržiavať vyššie uvedeného chápania informácie ako obsahu údajov.[1] Pojem *správa* sa zvykne používať na označenie údajov, ktoré majú istý formát a sú prenášané z jedného miesta na druhé. Okrem prenosu údajov v priestore sa údaje často prenášajú aj v čase: zapíšu sa na nejaké médium a po čase sa z neho čítajú. Pod *komunikáciou* budeme rozumieť činnosť dvoch alebo viacerých entít (účastníkov komunikácie), ktorá pozostáva z prenosu údajov/správ od jedného účastníka (odosielateľa) k druhému/iným (príjemca/príjemcovia). Existuje mnoho spôsobov komunikácie, ktoré závisia tak od použitých komunikačných prostriedkov (pošta, telefón, telegraf, televízia, rozhlas, a i.), typu údajov, ktoré sa pri komunikácii prenášajú aj účelu komunikácie. Nebudeme ich rozoberať, namiesto toho zavedieme pomerne všeobecný model komunikačného systému, popíšeme úlohu jeho jednotlivých subsystémov a ukážeme, aké úlohy sa musia pri komunikácii riešiť. Uvedený model použijeme tak na popis prenosu údajov v priestore, ako aj v čase.

Na obrázku 1.1 je uvedený Shannonov model komunikačného systému. Hoci je tento model veľmi všeobecný, hodí sa na popis mnohých komunikačných systémov a pre ďalšie (napríklad komunikačný systém so spätnou väzbou) môže Shannonov model poslúžiť ako základ, ktorý sa dá vhodne upraviť. Podrobnejší model komunikačného systému, odvodený zo Shannonovho modelu je uvedený na obrázku 1.2



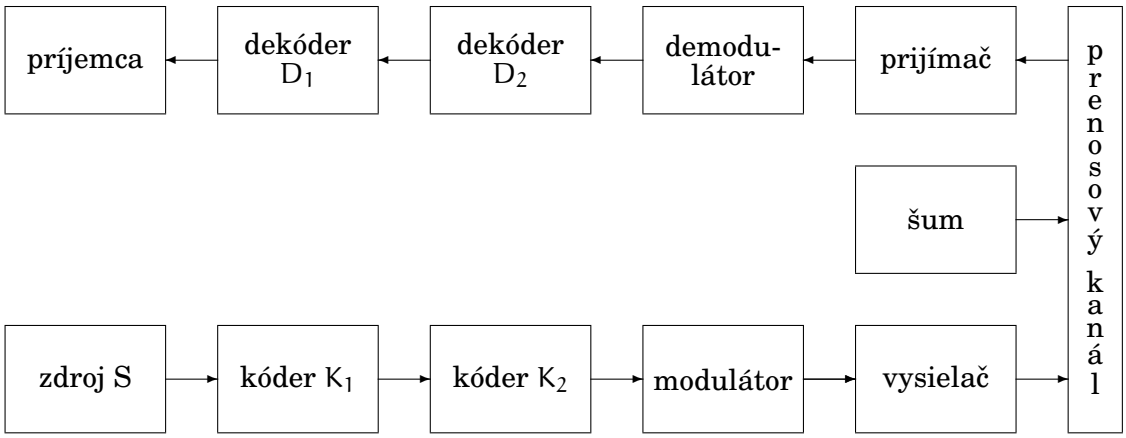
Obr. 1.1: Shannonov model komunikačného systému

Zdroj informácie/údajov. Aby sme sa nemuseli zaoberať tým, odkiaľ údaje (informácia) pochádzajú, budeme predpokladať, že existuje nejaký *zdroj informácie* (údajov), S (Source). Zdroju prislúcha nejaká abeceda, Σ_S, ktorú budeme nazývať *zdrojovou abecedou*, alebo *abecedou zdroja*. Ďalej Σ_S budeme predpokladať, že zdroj S generuje postupnosť znakov $x_i, x_{i+1}, \ldots; x_{i+j} \in \Sigma_S$; napríklad tak že v diskrétnych časových okamihoch (taktoch) sa na výstupe zdroja budú objavovať symboly zo zdrojovej abecedy. Postupnosť znakov zdrojovej abecedy môžeme spracovávať po znakoch, alebo rozdeliť na slová konečnej dĺžky. Bez ujmy na všeobecnosti môžeme predpokladať, že údaje, ktoré budeme

[1] Základné poznatky o meraní množstva informácie v údajoch sú uvedené v kapitole **??**

spracovávať, majú formu postupnosti slov[2] nad abecedou Σ_S.

Kóder K_1 – kódovanie zdrojových údajov. Forma, v ktorej sú zapísané zdrojové údaje, nemusí byť vhodná pre ďašie spracovanie, a preto je postupnosť slov vytvorená zdrojom pred ďalším spracovaním zakódovaná. Toto kódovanie sa nazýva *kódovanie zdrojovej informácie*, alebo *kódovanie zdroja* a realizuje ho kóder K_1. Výsledkom kódovania zdrojovej informácie je postupnosť symbolov kódovej abecedy Σ_C, ktorú budeme nazývať *správou*. Keďže pôvodnú informáciu získavame priamo zo zdroja, kódovanie zdroja nemusí riešiť ochranu údajov pred prípadnými chybami, ale plní inú úlohu—zaisťuje dosiahnutie efektívnosti zápisu zdrojovej informácie. Výsledkom kódovania zdrojových údajov je (v ideálnom prípade) najkratšia správa nad kódovou abecedou, na základe ktorej možno v plnom rozsahu zrekonštruovať zdrojové údaje v pôvodnej podobe. Požiadavka na efektívnosť kódovania správy sa dá vyjadriť tak, že vo výslednej (kódovanej) správe sa ľubovoľná k-tica znakov kódovej abecedy bude vyskytovať s rovnakou pravdepodobnosťou[3].



Obr. 1.2: Zovšeobecnený model komunikačného systému

Na tomto mieste sa na chvíľu zastavíme. Shannonov model komunikačného systému predpokladá, že v ideálnom prípade sa príjemcovi podarí zrekonštruovať správu v pôvodnom tvare. Aj keď sa v reálnych systémoch používa kódovanie zdroja, ktoré realizuje tzv. *bezstratovú kompresiu (data compaction)*, údaje generované zdrojom častokrát obsahujú informáciu, ktorú príjemca nedokáže využiť. Bezstratová kompresia takýchto údajov by viedla ku správam, ktoré by boli zbytočne rozsiahle. Ak dokážeme určiť, ktorá informácia obsiahnutá v zdrojových údajoch je podstatná a ktorá nie, môžeme na kódovanie zdrojových údajov použiť efektívnejšie kódovanie, založené na odfiltrovaní tak redundancie, ako aj nepodstatnej informácie obsiahnutej v zdrojových údajoch. Takéto kódovanie zdroja, pri ktorom dochádza k istej strate informácie sa nazýva *(kompresia so stratou informácie, data compression)*. Príkladom môže byť kódovanie hudby, ktoré využíva skutočnosť, že údaje obsahujú informáciu ktorú príjemca nedokáže využiť (nepočuteľné zvuky); táto informácia sa pri kódovaní zdroja jednoducho odfiltruje a tým

[2] v krajnom prípade slov dĺžky 1, teda znakov zdrojovej abecedy

[3] Zmyslom kódovania zdroja je odstrániť *redundanciu* (nadbytočnosť) pôvodného zápisu. Požiadavka na rovnakú pravdepodobnosť výskytu všetkých k-tic kódovej abecedy znamená, že v kódovanej správe už nebude možné objaviť nejakú zákonitosť, ktorá by sa dala využiť na ďalšie zefektívnenie zápisu.

zvýši efektívnosť zápisu (porovnajte nejakú hudobnú skladbu zapísanú na audio CD a zápis tej istej skladby vo formáte MP3, príp. iných). Na druhej strane mechanické použitie kompresie so stratou informácie nebude asi použiteľné pri spracovávaní exe súborov (hoci inteligentná revízia zdrojových textov tých istých programov by nepochybne odhalila možnosti optimalizácie textu.)

Správa Kódovanú správu rozdelíme na bloky vhodnej dĺžky k. (O výbere k budeme hovoriť neskôr.) Pripomíname, že ak bol kóder K_1 dostatočne kvalitný a kódovaná správa dostatočne dlhá, všetky slová dĺžky k by sa v nej mali vyskytovať s rovnakými pravdepodobnosťami.

Kóder K_2 Na rozdiel od kódovania zdroja, kde nebolo treba rátať so šumom a úlohou kódera K_1 bolo redukovať redundanciu zdrojových údajov, správu budeme čoskoro posielať cez komunikačný kanál, na ktorý pôsobí šum. Úlohou druhého kódera je transformovať slová dĺžky k nad kódovou abecedou Σ_C na slová dĺžky n (kvôli jednoduchosti predpokladajme, že nad tou istou kódovou abecedou Σ_C) tak, aby sa len mierne zvýšila redundancia a príjemca bol schopný odhaliť/opraviť chyby, ktoré vzniknú pri prenose prenosovým kanálom. Najprv budeme uvažovať kóder bez pamäte. Tento kóder realizuje injektívne zobrazenie

$$\mathrm{ENC} : \Sigma_C^k \to \Sigma_C^n.$$

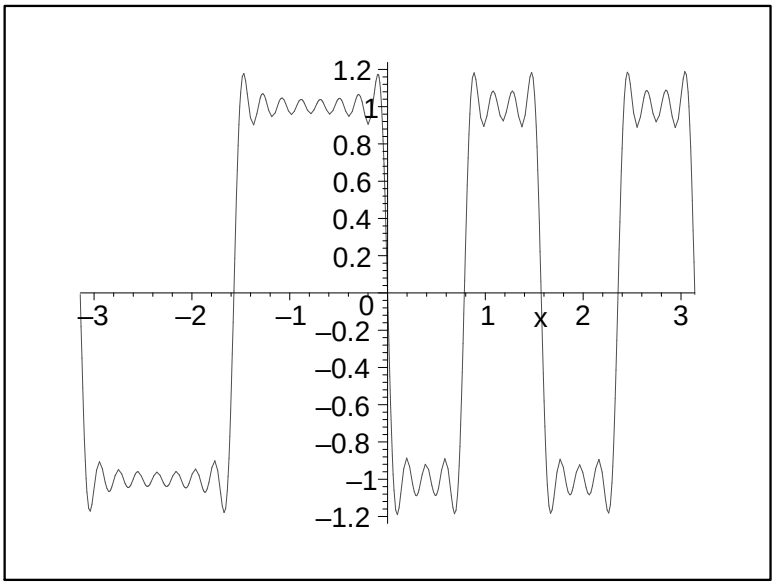
Kódery bez pamäte sa používajú na kódovanie pomocou blokových kódov a vyznačujú sa tým, že nezohľadňujú žiadne vzťahy medzi k-ticami vstupných údajov; to isté slovo (dĺžky k) sa zakaždým zobrazí na to isté slovo (dĺžky n). Existujú aj kódery s pamäťou, ktoré pri kódovaní znaku (zväčša kódujú znak po znaku) zohľadňujú aj predchádzajúce symboly. Tieto kódery sa používajú pri tzv. *konvolučných kódoch*.

Modulátor Správy sa prenášajú z jedného miesta na druhé pomocou fyzikálnych veličín, ktoré sa dokážu šíriť cez vhodné prostredie. Fyzikálna reprezentácia správy sa nazýva signál. (My budeme pomocou jedného signálu reprezentovať menšie časti správy, napríklad slová, alebo znaky kódovej abecedy.) Zariadenie, ktoré transformuje fyzikálnu veličinu tak, aby predstavovala príslušný signál, sa nazýva *modulátor*. Predstavme si napríklad rádiovú vlnu so sínusovým priebehom a amplitúdou 1 a binárnu kódovú abecedu $\Sigma = \{0, 1\}$. Symbolu 0 priradíme hodnotu -1 a symbolu 1 hodnotu $+1$. Postupnosť $0, 0, 1, 1, 0, 1, 0, 1$ bude reprezentovaná signálom, ktorého priebeh je uvedený na odrázku 1.3. Pre zaujímavosť uvedieme aj hodnoty signálov reprezentujúcich jednotlivé bity:

$$\begin{array}{cccc} -0.9479054106 & -0.9450567393 & 0.9450567393 & 0.9479054110 \\ -0.9180252682 & 0.9222918778 & -0.9222918783 & 0.9180252668 \end{array}$$

Vysielač je ďalším prvkom komunikačného systému. Jeho úlohou je generovať signály dostatočne silné na to, aby prekonali cestu k príjemcovi.

Prenosový kanál Signály sa môžu šíriť v rôznorodých prostrediach; napríklad kozmickým priestorom, po kovovom kábli, optickom vlákne a pod. Médium umožňujúce prenos signálov budeme nazývať *prenosovým kanálom*. Predpokladáme, že prenosový kanál je vystavený vplyvom okolitého prostredia, ktoré ovplyvňujú správy prenášané kanálom. Faktorov, ktoré môžu pôsobiť na prenosový kanál je tak veľa, že sa dosť dobre nedá



Obr. 1.3: Signál

skúmať vplyv jednotlivých faktorov, ale namiesto toho sa skúmajú dôsledky ich spoločného pôsobenia. Rôzne rušivé faktory vplývajúce na prenosový kanál, budeme nazývať zdrojmi šumu a výsledok ich pôsobenia—šumom . Budeme predpokladať, že šum má podobu signálov, ktoré ovplyvňujú signály prenášajúce správu (napríklad sa s nimi skladajú), v dôsledku čoho dôchádza k zmenám signálov, ktoré sa v prenášanej správe prejavujú ako chyby troch základných typov:

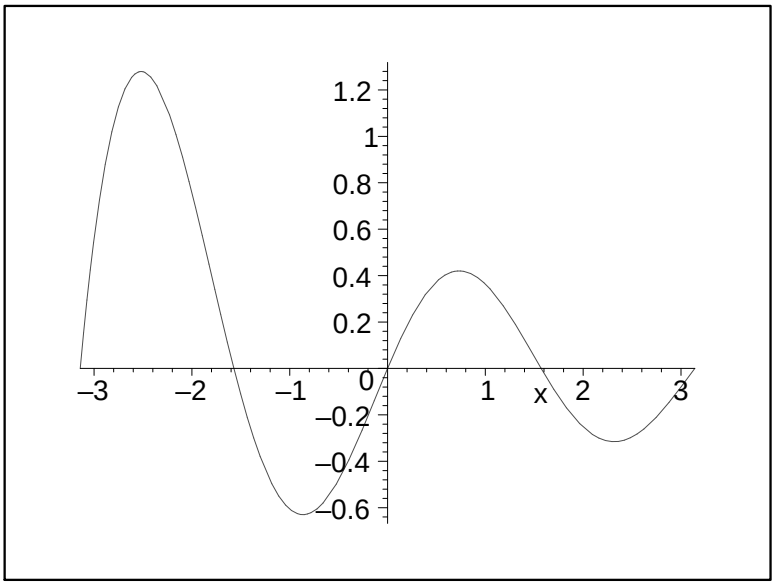
1. nahradenie jedného symbolu prenášanej správy iným symbolom (kódovej abecedy);
2. zmazaním symbolu (čo môžeme chápať tak, že symbol prenášanej správy je nahradený symbolom, ktorý nepatrí do kódovej abecedy);
3. výpadkom/doplnením nového symbolu (kódovej abecedy) do prenášanej správy (porucha synchronizácie).

Šumový signál je zobrazený na obrázku 1.4

V tejto knihe sa budeme zaoberať kódmi, ktoré umožnia riešiť chyby prvého a druhého druhu; t.j. odhaľovať a opravovať chyby. Poruchami synchronizácie sa nebudeme zaoberať, čitateľovi odporúčame

TO DO

Signály prenášané prenosovým kanálom zachytáva prijímacia strana pomocou **prijímača** (napríklad anténa mobilného telefónu). Abstrahujeme od transformácií signálov, ktoré realizuje prijímač a predpokladáme, že prijaté signály vstupujú do **demodulátora**, ktorý transformuje signály na postupnosť znakov kódovej abecedy. Demodulátor už v podstate robí prvú korekciu chýb. V dôsledku pôsobenia šumu na kanál prijaté



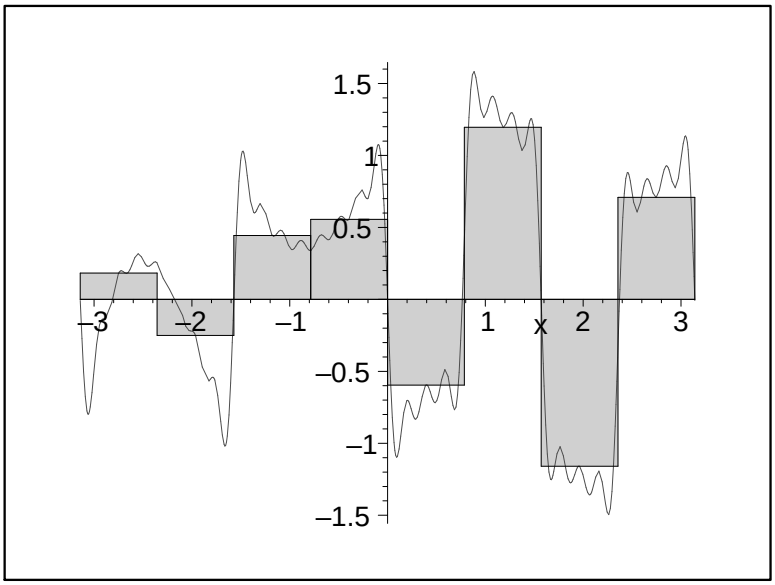
Obr. 1.4: Šum

signály nebudú mať zďaleka ideálny priebeh, obr. 1.5. Postupnosť $0,0,1,1,0,1,0,1$ je reprezentovaná postupnosťou hodnôt[4]

bit	pôvodný signál	prijatý signál	interpretácia	
			tvrdá	mäkká
0	−0.9479054106	0.1824649004	1	?
0	−0.9450567393	−0.2506249324	0	?
1	0.9450567393	0.4439007163	1	1
1	0.9479054110	0.5558633849	1	1
0	−0.9180252682	−0.5960788882	0	0
1	0.9222918778	1.195406403	1	1
0	−0.9222918783	−1.159436952	0	0
1	0.9180252668	0.7084777992	1	1

Ani jedna z prijatých hodnôt (signálov) nepatrí do množiny $\{-1,1\}$. Aby dekóder mohol transformovať prijaté signály na znaky kódovej abecedy, musí použiť pružnejšie pravidlo; napríklad, signály s nezápornými hodnotami budú reprezentovať 1 a ostatné signály budú reprezentovať kódový znak 0. Pri použití tohto pravidla sa značne deformované signály transformujú na postupnosť $1,0,1,1,0,1,0,1$. Demodulátor môže byť navrhnutý tak, aby zakaždým prijal rozhodnutie o interpretácii signálu (*hard quantization*). To by však mohlo viesť k nesprávnej interpretácii signálov blízkych k 0 demodulátorom a problém (identifikáciu a opravenie chyby) by musel riešiť dekóder. Napriek tomu, že pri "tvrdej"transformácii spojitého signálu na diskrétne hodnoty sa dosahuje najvyššia pravdepodobnosť správneho priradenia, často sa používa alternatívne riešenia, tzv. *soft quantization*. V krajnom prípade demodulátor neinterpretuje žiadne signály ako

[4]tieto predstavujú hodnoty signálu v strede príslušných intervalov.



Obr. 1.5: Prijatý signál

kódové znaky a posunie dekóderu vypočítané číselné hodnoty signálov na jednotlivých časových intervaloch. Tým sa prakticky celé spracovanie prijatej správy presunie na dekóder, čo však zvyšuje nároky na jeho výkonnosť (zložitosť a zrejme aj cenu). Rozumné je preto kompromisné riešenie, kedy demodulátor predspracuje prijaté signály napríklad tak, že jednoznačne interpretuje tie signály, ktorých hodnoty dostatočne dobre zodpovedajú hodnotám reprezentujúcim jednotlivé znaky kódovej abecedy, problematické hodnoty signálu bude reprezentovať nejakým novým symbolom a tieto výsledky odovzdá dekóderu. Presnejšie, nech s_k označuje priemernú hodnotu signálu v takte k, a a_k je hodnota symbolu kódovej abecedy zodpovedajúca signálu s_k. Potom pravidlo pre binárnu kódovú abecedu by mohlo vyzerať napríklad takto

$$a_k = \begin{cases} 1 & s_k \geq 0.3, \\ 0 & s_k \leq -0.3, \\ ? & -0.3 < s_k < 0.3. \end{cases}$$

V našom prípade by demodulátor postupnosť prijatých signálov interpretoval ako postupnosť $?, ?, 1, 1, 0, 1, 0, 1$. Ak počas prenosu došlo ku chybe (nahradenie jedného znaku kódového slova iným), informácia, ktorú dostal dekóder od demodulátora by mu umožnila odhadnúť najpravdepodobnejšie miesta, na ktorých mohlo dôjsť ku chybe (v našom príklade sú podozrivé prvé dva symboly), čím by sa (ako uvidíme neskôr) značne zjednodušilo dekódovanie.

Dekódery D_2 a D_1 Hlavnou úlohou dekódera D_2 je čo najlepšie rekonštruovať odvysielanú správu. Predpokladajme, že poznáme všetky kódové slová (budeme ich označovať ako $\mathbf{u}_i$), všetky možné prijaté slová $\mathbf{v}_j$ a podmienené pravdepodobnosti $p_{i,j} = p(\mathbf{v}_j|\mathbf{u}_i)$. Pravdepodobnosť $p_{i,j}$ vyjadruje pravdepodobnosť toho, že po odvysielaní kódového slova $\mathbf{u}_i$ bolo prijaté (nejaké) slovo $\mathbf{v}_j$. Základom pre rozhodovanie dekódera D_2 je nasledujúci

princíp *dekódovania na základe maximálnej pravdepodobnosti* (*Maximum Likelihood Decoding, MLD*):

> Prijaté slovo $\mathbf{v}_j$ dekódujeme na také kódové slovo $\mathbf{u}_i$, pre ktoré je podmienená pravdepodobnosť $p(\mathbf{v}_j|\mathbf{u}_i)$ maximálna.

Dekódovanie na základe maximálnej pravdepodobnosti vždy dáva výsledok (kódové slovo). Takéto dekódovanie sa nazýva *úplné dekódovanie* . Ako sa dá očakávať, okrem úplného dekódovania bude existovať aj nejaké alternatívne riešenie, ktoré budeme nazývať *neúplným dekódovaním*. Pri neúplnom dekódovaní môžu nastať dva prípady:

1. dekóder dekóduje prijaté slovo t.j. priradí prijatému slovu kódové slovo,
2. dekóder namiesto kódového slova vypíše nejaký dohodnutý symbol (napríklad ∞).

Druhý prípad nastane vtedy, keď dekóder našiel v prijatom slove chybu, ale nebol ju schopný opraviť. Takúto situáciu nemôžeme vylúčiť, pretože dekóder nie je schopný opraviť slová, ktoré boli výrazne modifikované. V takom prípade je lepšie požiadať o opätovné zaslanie informácie, ako sa pokúšať opraviť prijaté slovo a dekódovať ho nesprávne. Ani takýto prístup však nezaručuje, že dekódované slovo sa zhoduje s odvysielaným kódovým slovom. Keďže dekóder rozhoduje na základe syntaxe (napr. zoznamu kódových slov) a nie sémantiky prijatých správ, ak počas prenosu kódového slova nastala chyba, ktorá ho transformovala na iné **kódové slovo**, dekóder takúto chybu nedokáže identifikovať. Preto dekóder postavený na princípe MLD bude mať síce najvyššiu pravdepodobnosť správneho dekódovania, ale ak sa pomýli, tak je dekódovaná správa zaťažená chybou, ktorú je ťažko odhaliť. Preto väčšina dekóderov, ktorými sa budeme zaoberať, vychádza z trocha slabšieho princípu, nazývaného *IMLD* (*Incomplete Maximum Likelihood Decoding*); *neúplné dekódovanie na základe maximálnej pravdepodobnosti:*

> Prijaté slovo $\mathbf{v}_j$ dekódujeme buď na také kódové slovo $\mathbf{u}_i$, pre ktoré je podmienená pravdepodobnosť $p(\mathbf{v}_j|\mathbf{u}_i)$ maximálna, alebo na symbol ∞; bola odhalená chyba.

Napriek použitiu samoopravných kódov nedokážeme garantovať správne dekódovanie prijatej správy. Budeme rozlišovať dva problematické prípady: *chyba dekódera* (*decoder error*) nastáva vtedy, keď dekóder nesprávne dekódoval prijaté slovo; t.j. interpretoval ho ako iné kódové slovo, ako bolo to, ktoré bolo odvysielané. *Zlyhanie dekódera* zahŕňa tak chybu dekódera, ako aj ten prípad, keď dekóder odhalil chybu ale nebol ju schopný opraviť.

Dekóder D_1 transformuje dekódovanú správu do podoby, v ktorej ju môže ďalej spracovávať príjemca. Dobrým príkladom je dekódovanie binárne zapísanej zvukovej informácie (hudby) do počúvateľnej podoby, alebo dekódovanie digitálne zapísaných filmov na DVD.

Všimneme si, že hoci sme v modeli prenosového kanála hovorili o prenose správ, tento model možno priamo použiť aj na popis uchovávania a opätovného čítania údajov. Zaznamenávanie údajov a ich opätovné čítanie možno chápať ako prenos informácie v čase, zatiaľ čo pri prenose správ sa jedná o prenos informácie v priestore. Jeden podstatný rozdiel medzi prenosom údajov v čase a priestore však je. Ak pri prenose informácie v priestore dôjde k odhaliteľnej ale neopraviteľnej chybe, príjemca má možnosť požiadať odosielateľa o opätovné zaslatie správy. Ak však dôjde k poškodeniu údajov zapísaných na nejakom pamäťovom médiu, opätovné čítanie neumožní prečítať správne údaje. O to dôležitejšie je pri uchovávaní údajov na pamäťových médiách ochrana ich integrity napríklad pomocou samoopravných kódov. Z hľadiska úloh, ktoré rieši teória kódovania nie je rozdiel informácie v čase a priestore podstatný, a preto sa v ďalšom sa sústredíme na problémy vznikajúcimi pri prenose informácie v priestore.

1.3 Kódovanie

Vráťme sa k modelu komunikačného systému z predchádzajúcej časti. Zostáva vyriešiť problém, ako zapisovať správy, ktoré generuje zdroj S v podobe postupnosti znakov nad abecedou Σ_S pomocou kódovej abecedy Σ_C. Existuje viacero riešení tohto problému. Začneme tým najjednoduchším—kódovaním jednotlivých znakov zdrojovej abecedy. Nech $\Sigma_S = \{s_0, \ldots, s_{m-1}\}$ je zdrojová abeceda a $\Sigma_C = \{b_0, \ldots, b_r\}$ je kódová abeceda a nech sú $\nu_0, \ldots, \nu_{m-1}$ navzájom rôzne slová nad kódovou abecedou Σ_C. Potom zobrazenie

$$\begin{aligned} s_0 &\rightarrow \nu_0 \\ s_1 &\rightarrow \nu_1 \\ &\vdots \\ s_{m-1} &\rightarrow \nu_{m-1} \end{aligned}$$

budeme nazývať *kódovaním symbolov zdrojovej abecedy* slovami nad abecedou Σ_C. Množina $V = \{\nu_0, \ldots, \nu_{m-1}\}$ sa nazýva *kód* a prvky množiny V sa nazývajú *kódovými slovami*. Všimneme si, že kódovanie po písmenách je totálnym (všade definovaným) zobrazením a keďže zdroj S generuje len postupnosti znakov nad abecedou Σ_S, každá správa vytvorená zdrojom S sa dá vyjadriť pomocou postupnosti kódových slov kódu V. Problém však vzniká pri dekódovaní kódovaných správ. Predpokladajme, že je daná nejaká správa $s_{i_1}, \ldots, s_{i_n}$ nad zdrojovou abecedou a jej prislúchajúca kódovaná správa vyjadrená ako postupnosť kódových slov $\nu_{i_1}, \ldots, \nu_{i_n}$. Postupnosť $\nu_{i_1}, \ldots, \nu_{i_n}$ sa však prenáša po znakoch a pred dekódovaním je potrebné ju rozdeliť na kódové slová. Ak sa to podarí, nie je problém dekódovať jednotlivé kódové slová a získať pôvodnú správu $s_{i_1}, \ldots, s_{i_n}$. Nasledujúci príklad ukazuje, že existujú také kódy, pre ktoré sa nie každá postupnosť kódových symbolov dá jednoznačne rozdeliť na kódové slová.

Príklad. Abeceda zdroja $\Sigma_S = \{0, 1, 2, 3\}$ pozostáva zo štyroch symbolov (prvých štyroch desiatkových číslic) a kódová abeceda je binárna; $\Sigma_C = \{0, 1\}$. Kódovanie priraďuje de-

siatkovej číslici jej binárny zápis:

$$\begin{array}{cccccc} 0 & \to & 0 & 1 & \to & 1 \\ 2 & \to & 10 & 3 & \to & 11 \end{array}$$

Uvažujme napr. binárnu postupnosť 001011. Táto postupnosť sa dá interpretovať viacerými spôsobmi, a síce ako binárny zápis desiatkových postupností 001011, 00103, 00211, 0023.

Jednoznačnosť dekódovania je prirodzenou požiadavkou, ktorá sa kladie na kódovanie. Nutným predpokladom jednoznačnosti dekódovania je tzv. *rozdeliteľnosť kódu*.

Definícia 1.3.1. *Kód* $V = \{v_0, \ldots, v_{m-1}\}$ *nad abecedou* Σ_C *sa nazýva rozdeliteľným, ak pre ľubovoľnú rovnosť postupností kódových slov*

$$v_{i_1} \ldots v_{i_k} = v_{j_1} \ldots v_{j_l}$$

platí $l = k, \quad i_1 = j_1, \ldots, i_k = j_k$.

Čo vlastne vyjadruje rozdeliteľnosť kódu? Ak je kód V rozdeliteľný, znamená to, že ľubovoľnú postupnosť nad Σ_C^* buď môžeme rozdeliť na postupnosť kódových slov jednoznačným spôsobom, alebo ju nemôžeme rozdeliť vôbec. Pre rozdeliteľný kód nemôže nastať taká situácia, kedy by sme nejakú postupnosť kódových symbolov mohli rozdeliť na postupnosť kódových slov dvoma rozličnými spôsobmi. Jednoduchým riešením problému rozdeliteľnosti sú *blokové* alebo *rovnomerné kódy*. Blokový kód sa vyznačuje tým, že všetky jeho kódové slová majú rovnakú dĺžku.

Príklad. Rozšírime predchádzajúci príklad a uvedieme dva spôsoby binárneho kódovania desiatkových číslic—rovnomerné a nerovnomerné:

desiatkový zápis	binárny zápis	blokový kód
0	0	0000
1	1	0001
2	10	0010
3	11	0011
4	100	0100
5	101	0101
6	110	0110
7	111	0111
8	1000	1000
9	1001	1001

Dekódovanie postupnosti znakov kódovej abecedy bude v prípade blokového kódu relatívne jednoduché: postupnosť sa najprv rozdelí na slová dĺžky rovnej dĺžke bloku a potom sa (napríklad na základe tabuľky) jednotlivým kódovým slovám priradia symboly zdrojovej abecedy.

Príklad. Postupnosť 100001001100100100110010 rozdelíme na kódové slová:
1000 0100 1100 1001 0011 0010 a dekódujeme pomocou tabuľky z predchádzjúceho príkladu: 843932. Všimneme si, že existujú aj binárne postupnosti, ktoré sa nedajú dekódovať, nakoľko slová 1111, 1110, 1101, 1100, 1011, 1010 nie sú kódové slová.

S rozdeliteľnosťou vznikajú problémy pri použití niektorých kódov, ktoré obsahujú slová nerovnakej dĺžky; tzv. *nerovnomerných kódoch*. Postupnosť kódových symbolov v tomto prípade nemožno mechanicky rozdeliť na bloky rovnakej dĺžky, ale je potrebné určiť kódové slová. To sa v prípade nerovnomerných kódov vo všeobecnosti nemusí dať spraviť (alebo nedá spraviť jednoznačne). Ale aj medzi nerovnomernými kódmi existujú rozdeliteľné kódy. Nakoľko tieto kódy umožňujú zapisovať informáciu častokrát úspornejšie ako blokové kódy, používajú sa najmä na (bezstratovú) kompresiu údajov. Podrobnejšie sa nimi budeme zaoberať v nasledujúcich kapitolách. Vráťme sa teraz ku kódovaniu zdrojovej informácie. Zatiaľ sme kódovali jednotlivé znaky kódovej abecedy, teraz pojem kódovania znakov zdrojovej abecedy zovšeobecníme.

Definícia 1.3.2. *Nech je* Σ_S *zdrojová abeceda, nech je množina* $U = \{u_0, \ldots, u_M\}$ *nejakých slov nad zdrojovou abecedou a nech sú* $v_0, \ldots, v_M$ *slová nad kódovou abecedou* Σ_C. *Zobrazenie*

$$\begin{aligned} u_0 &\rightarrow v_0 \\ u_1 &\rightarrow v_1 \\ &\vdots \\ u_M &\rightarrow v_M \end{aligned}$$

budeme nazývať kódovaním množiny U kódom V.

Všimneme si, že táto definícia kódovania zahŕňa aj kódovanie znakov zdrojovej abecedy; stačí položiť $U = \Sigma_S$.

Príklad. Nech je $\mathcal{U}$ rovná množine všetkých podmnožín množiny prirodzených čísel $\{0, \ldots .99\}$, $V = \{0,1\}^{100}$ je množina binárnych vektorov dĺžky 100. Podmnožine $\{i_0, \ldots, i_k\}$ z $\mathcal{U}$ je priradené slovo v_j[5], ktoré má jednotkové hodnoty na pozíciách $i_0, \ldots, i_k$ a nuly na ostatných pozíciách. Je zrejmé, že V kóduje množinu $\mathcal{U}$ a že toto kódovanie je bijekciou. Poznajúc mohutnosť kódu V vieme určiť aj mohutnosť množiny $\mathcal{U}$: $|\mathcal{U}| = 2^{100}$.

1.4 Formát

Údaje budem v počítačoch zapisovať pomocou slov nad nejakou (najčastejšie binárnou) abecedou. Aby rôzne zariadenia dokázali rovnako interpretovať tie isté údaje, musí existovať zhoda v tom, čo slovo vyjadruje, t.j. akú má dĺžku, na aké podslová sa slovo delí a čo jednotlivé podslová znamenajú. Konvencia o štuktúre slova a interpretácii jeho podslov sa nazýva *formát*. Napríklad, ak 16 bitový vektor predstavuje celé číslo so znamienkom, tak vieme, že najvyšší (najľavejší) bit bude kódovať znamienko a ostávajúcich 15 bitov reprezentuje absolútnu hodnotu čísla. Tým, že formát popisuje štruktúru slova, zároveň definuje aj spôsob vytvárania syntakticky správneho zápisu (kódových slov).

[5] tzv. charakteristický vektor

Kapitola 2

Reprezentácia informácie v počítači

2.1 úvod

Počítač je zariadenie na automatizované spracovanie informácie. V súčasných digitálnych počítačoch je informácia reprezentovaná pomocou binárnych vektorov rôznej, ale spravidla pevne stanovenej dĺžky. Význam binárneho vektora závisí od toho, čo reprezentuje a aký formát sa na zápis danej informácie v počítači používa. V tejto kapitole sa pozrieme na základné typy informácie, ktoré sa v počítači vyskytujú; t.j. ako je reprezentovaná číselná, textová, riadiaca, zvuková a obrazová informácia a na základné aritmetické operácie nad binárnymi číslami v rôznych formátoch.

2.2 Pozičné číselné sústavy

Základom pre pochopenie reprezentácie rôznych údajov v počítači je dokonalé porozumenie princípom binárneho zápisu číselnej informácie v počítači. Začneme výkladom základných princípov pozičných číselných sústav, po ich zvládnutí sa budeme zaoberať kódovaním číselnej a inej informácie.

Bežne používaná desiatková sústava je príkladom *pozičnej číselnej sústavy*. Tento pojem vyjadruje skutočnosť, že číselná hodnota čísla zapísaného v takejto sústave závisí tak od číslic, ako aj ich poradia v čísle, ktoré určuje ich váhu. Váha číslice je vyjadrená pomocou mocnín zvláštneho (prirodzeného) čísla, nazývaného základom danej číselnej sústavy. V desiatkovej sústave je základom číslo 10.

Napr. číslo 723207.123 má hodnotu

$$7 \times 10^5 + 2 \times 10^4 + 3 \times 10^3 + 2 \times 10^2 + 0 \times 10^1 + 7 \times 10^0 + 1 \times 10^{-1} + 2 \times 10^{-2} + 3 \times 10^{-3}.$$

Ak je základom číselnej sústavy číslo z, tak číslo v z-adickej sústave je vyjadrené ako postupnosť číslic z množiny $\{0, \ldots, z-1\}$, ktoré predstavuje hodnotu (celočíselná a zlomková časť sú oddelené rádovou čiarkou/bodkou[1])

$$(a_n a_{n-1} \ldots a_0 . a_{-1} \ldots a_{-m})_z = \sum_{k=-m}^{n} a_k \times z^k$$

2.2.1 Prevod čísel medzi desiatkovou a z-adickou sústavou

Nech je daná pozičná číselná sústava so základom z. Kladné desiatkové číslo $x.y$ s celou časťou $x > 0$ a zlomkovou časťou y prevedieme do z-adickej sústavy. Použijeme pri tom všeobecnejší postup, ktorý sa v matematike dosť často používa: budeme predpokladať, že úloha je vyriešená a potom z predpokladaného výsledku odvodíme postup riešenia. V našom prípade predpokladáme, že číslo je už zapísané v z-adickej sústave (začneme celou časťou):

$$x = a_{n-1} \cdot z^{n-1} + a_{n-2} \cdot z^{n-2} + \cdots + a_1 \cdot z^1 + a_0 \cdot z^0.$$

Číslo x môžeme vyjadriť takto:

$$x = z \cdot \left[a_{n-1} \cdot z^{n-2} + a_{n-2} \cdot z^{n-3} + \cdots + a_1 \cdot z^0 \right] + a_0.$$

Aby sme určili číslicu a_0, stačí nám vydeliť číslo x číslom z. Hľadanú hodnotu a_0 dostávame ako výsledok modulárneho delenia

$$x \mod z = a_0.$$

Všimneme si, že vo výsledku celočíselného delenia $x \div z$ došlo k zníženiu hodôt exponentov a úlohu a_0 z prvého kroku teraz zohráva a_1:

$$x \div z = a_{n-1} \cdot z^{n-2} + a_{n-2} \cdot z^{n-3} + \cdots + a_1 \cdot z^0.$$

Zopakujeme celý postup n-krát a určíme všetky číslice $a_{n-1}, \ldots . a_0$.

Zapíšeme ešte celý postup formálne, pričom predpokladáme, že zápis čísla x v sústave so základom z má n číslic; $n \leq 1 + \log_z x$):

[1] v desiatkovej sústave ju nazývame desatinnou čiarkou

prevod celej časti

```
k:=0;
while (x > 0) & (k < n) do
{
a[k]:= x mod z;       \*  vypocitame hodnotu k-tej cislice
                      \*  celej casti
x:= x div z;
k:= k+1;
}
```

Výsledok je uložený v poli `a[0...n-1]`, pričom číslica uložená v `a[k]` má váhu z^k. Pri prevode zlomkovej časti môže nastať situácia, že číslo, ktoré má konečný rozvoj v jednej číselnej sústave, má nekonečný rozvoj v inej číselnej sústave (napríklad $1/3$ má v desiatkovej sústave zápis $(0.3333\overline{3})_{10}$ ale v trojkovej sústave konečný zápis $(0.1)_3$. Preto pri prevode zlomkovej časti počítame len s presnosťou na daný počet číslic, napríklad m.

prevod zlomkovej časti

Podobne ako pri prevode celej časti budeme aj pri prevode zlomkovej časti predpokladať, že zlomková časť y je už zapísaná v sústave so základom z s presnos5ou na m miest:

$$y = b_{-1} \cdot z^{-1} + b_{-2} \cdot z^{-2} + \cdots + b_{-m} \cdot z^{-m}.$$

Keď vynásobíme číslo y číslom z, dostávame

$$y \cdot z = b_{-1} \cdot z^0 + b_{-2} \cdot z^{-1} + \cdots + b_{-m} \cdot z^{-m+1} = b_{-1} \cdot b_{-2} \ldots b_{-m}.$$

Číslica b_{-1} sa „posunula“ pred rádovú čiarku; jej hodnota zodpovedá dolnej celej časti súčinu $y \cdot z$. Na prvé miesto po rádovej čiarke sa dostala číslica b_{-2}. Odpočítame $y \cdot z - \lfloor y \cdot z \rfloor$ a celý postup zopakujeme ešte $m-1$ krát a určíme zlomkovú časť y v z-adickej sústave s presnosťou na m miest. Formálne algoritmus prevodu zlomkovej časti čísla je uvedený nižšie:

```
k:=1;
while (y > 0) & (k < m) do
{
y:= y*z;
b[k]:= Int(y);        \*  vypocitame hodnotu k-tej cislice
                      \*  zlomkovej casti
y:=y-Int(y);          \*  y priradime zlomkovu cast cisla
k:= k+1;
}
```

Výsledok je uložený v poli `b[1...m]`, pričom číslica uložená v `b[k]` má váhu z^{-k}.

Príklad Prevedieme desiatkové číslo 598 do sústavy so základom 6. Výsledky krokov výpočtu sú uvedené v tabuľke

k	x	$x \mod 6$	$x \div 6$
0	598	3	85
1	85	1	12
2	12	5	1
3	0	1	0

Výsledok $(598)_{10} = (1513)_6$.

Ak by sme náhodou potrebovali previesť číslo napr. zo sústavy so základom 13 do sústavy so základom 37, jednoduchšie bude použiť ako prostredníka desiatkovú sústavu; najprv previesť číslo z trinástkovej sústavy do desiatkovej sústavy a potom výsledok previesť do 37-ovej sústavy. (Experti, ktorí ovládajú násobilku v trinástkovej a tridsaťsedmičkovej sústave, môžu číslo previesť priamo aj bez použitia desiatkovej sústavy.)

2.2.2 Základné aritmetické operácie s binárnymi číslami

V pozičnej binárnej číselnej sústave sa základné aritmetické operácie (sčítanie, odčítanie, násobenie a delenie) vykonávajú podobne ako v desiatkovej sústave. Rozdiel je v tom, že binárna sústava má jednoduchšie pravidlá pre sčítanie/odčítanie a násobilku. Predpokladajme, že a, b sú binárne číslice

a	b	prenos	súčet
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

a	b	súčin
0	0	0
0	1	0
1	0	0
1	1	1

Začneme aditívnymi operáciami s kladnými celými číslami. Kvôli prehľadnosti budeme čísla, s ktorými budeme pracovať, zapisovať v tabuľke a vyznačovať miesta, na ktorých došlo k mimoriadnej situácii.

Sčítanie a odčítanie celých čísel.

a	0	1	1	0	1	1	0	0	1	1	1
b	0	0	1	1	1	0	1	0	1	0	1
a + b	1	0	1	0	0	1	1	1	1	0	0
	—	*	*	*	*	—	—	—	*	*	*

Na miestach označených hviezdičkou došlo k prenostu do vyššieho rádu. Všimnite si, že súčtom dvoch 10-miestnych čísel je 11 miestne číslo. Ak by takáto situácia nastala v počítači, ktorý má na uchovanie čísla obmedzený počet bitov, a výsledok by bol väčší ako

najväčšie zobraziteľné číslo, nezmestil by sa do vymedzeného počtu bitov a nastalo by tzv. pretečenie. Riešením takýchto prípadov sa budeme zaoberať neskôr.

Pri odčítaní porovnáme veľkosť menšenca (čslo, od ktorého očítavame) a menšiteľa (číslo, ktoré odčítavame). Pripomíname, že obe čísla sú kladné. Od väčšieho čísla odčítavame menšie číslo a keď bol pôvodný menšenec menší ako menšiteľ, výsledkom je záporné číslo, v opačnom prípade 0 alebo kladné číslo. Pri odčítavaní si môžeme „požičiavať“ z vyššieho rádu (ale pôžičku musíme nakoniec vrátiť). V nasledujúcom príklade je výsledkom očítania kladné číslo, v poslednom riadku sú hviezdičkami vyznačené rády, v ktorých došlo k „požičiavaniu“.

a	0	1	1	0	1	1	0	0	1	1	1
b	0	0	1	1	1	0	1	0	1	0	1
a − b	0	0	1	1	0	0	1	0	0	1	0
	−	−	*	*	−	−	*	−	−	−	−

Násobenie celých čísel

Násobenie binárnych celých čísel je jednoduché - násobenie dvoma znamená posun čísla o jedno miesto doľava a pripísanie 0 na najnižšie miesto; násobenie viacmiestnym činiteľom sa redukuje na posuny prvého činiteľa a potom spočítanie „posunutých“ exemplárov prvého ščítanca. Jediným problémom je, že pri sčítaní môže dôjsť k prenosu o niekoľko rádov vyššie, na čo si v prípade ručného počítania treba dať pozor. Takisto si je potrebné uvedomiť, že súčinom dvoch n-miestnych čísel môže byť $2n - 1$ miestne číslo

				1	0	0	1	1	0	1	*	1	0	0	1	1
			1	0	0	1	1	0	1							
	1	0	0	1	1	0	1									
	1	0	1	1	0	1	1	0	1	1	1					

Delenie celých čísel

Pri delení celých čísel spisujeme postupne najvyššie bity delenca a spísanú (označenú časť) porovnávame s deliteľom. Ak je spísaná časť menšia ako deliteľ, a na najnižšie miesto výsledku zapíšame 0, v opačnom prípade od spísanej časti delenca odčítame deliteľa a na najnižšie miesto deliteľa zapíšeme 1. Keď spracujeme poslednú číslicu deliteľa, vo výsledku je celočíselný podiel delenca a deliteľa a spísaná časť (menšia ako deliteľ) je zvyšok po delení. Akl by sme pokračovali v delení, mohli by sme vypočítať zlomkovú časť čísla s požadovanou presnosťou.

1✓010111001	1101	0
10✓10111001	1101	00
101✓0111001	1101	000
1010✓111001	1101	0000
10101✓11001	1101	00001
−1101		
10001✓1001	1101	000011
−1101		
1001✓001	1101	0000110
10010✓01	1101	00001101
−1101		
1010✓1	1101	000011010
10101✓	1101	0000110101
−1101		
1000		

2.3 Kódovanie celých čísel

Celé čísla paria medzi základné typy údajov. Keďže sa v počítačoch a iných digitálnych zariadeniach využívajú na rôzne účely, bývajú zapísané v rôznych formátoch. V tejto časti vysvetlíme najdôležitejšie formáty zápisu celých čísel. Príjmeme jeden zjednodušujúci predpoklad. Hoci je dôležitým parametrom formátu dĺžka slova, z principiálneho hľadiska nie je podstatný rozdiel napr. medzi celým číslom so znamienkom dĺžky 16 a 32, a preto budeme predpokladať, že celé čísla sú zapísané pomocou binárnych slov (vektorov) dĺžky n.

2.3.1 Celé čísla bez znamienka (unsigned integer)

Formát celé číslo bez znamienka je najjednoduchší formát zápisu celých čísel, z ktorého navyše sú odvodené ostatné formáty. V tomto formáte vektor $a_{n-1}, a_{n-2}, \ldots, a_0$ reprezentuje binárne číslo $\sum_0^{n-1} a_i \cdot 2^i$. Napríklad vektor 01100111 predstavuje binárny zápis čísla 103. V tomto formáte (pri dĺžke slova n) je možné zobraziť všetky celé čísla z intervalu $\langle 0, \ldots, 2^n - 1 \rangle$, teda celkovo 2^n celých čísel.

2.3.2 Celé čísla so znamienkom (integer)

Tento formát je prirodzeným rozšírením predchádzajúceho formátu, tak aby bolo možné zapisovať kladné aj záporné čísla. Najvyšší bit čísla sa rezervuje na označenie znamienka; kladné znamienko reprezentuje 0, záporné 1. Ostatné bity čísla predstavujú celé číslo bez znamienka. Vektor 01100111 má rovnakú interpretáciu ako v prípade keby predstavoval celé číslo bez znamienka, t.j. číslo $+103$. Číslo 11000000 predstavuje vo formáte celé číslo so znamienkom -64, zatiaľ čo vo formáte celého čísla bez znamienka

predstavuje číslo 192. Pomocou n-bitového vektora je vo formáte celé číslo so znamienkom možné zobraziť všetky celé čísla z intervalu $-2^{n-1}+1,\ldots,2^{n-1}-1$. Zvláštnosťou tohto formátu je, že sa 0 reprezentuje dvomi rôznymi spôsobmi: vektor $00\ldots0$ predstavuje kladnú a vektor $10\ldots0$ zápornú reprezentáciu nuly. Vzhľadom na túto anomáliu je pomocou n-bitového vektora možné zapísať vo formáte celé číslo so znamienkom 2^n-1 celých čísel.

Ďalšie tri formáty umožňujú zapisovať celé čísla v pozičných číselných sústavách so základom 10, 8 a 16. Keďže počítače sú schopné spracovávať len binárne zapísanú informáciu, desiatkové, osmičkové a šestnástkové čísla musia byť v konečnom dôsledku zapísané binárne.

2.3.3 Binárne kódované celé čísla (binary coded decimal)

Ako vyplýva z názvu, tento formát slúži na binárny zápis desiatkových čísel. Desiatkové číslice sú reprezentované 4-bitovými celými číslami bez znamienka a desiatkové číslo je zapísané ako postupnosť štvoríc bitov, kódujúcich jeho jednotlivé číslice. Kódové slová pre jednotlivé desiatkové číslice sú uvedené v nasledujúcej tabuľke:

dekadická číslica	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Desiatkové celé číslo 9578 sa v BCD kóde zapisuje ako 1001010101111000. BCD kódovanie využíva zo 16 možných vektorov len 10 a nie je veľmi efektívne, napríklad pomocou 16-bitového vektora dokážeme v BCD kóde zapísať 10000 rôznych celých čísel, ale vo formáte celé číslo bez znamienka až 65536.

2.3.4 Osmičkové čísla (octal numbers)

Celé číslo (bez znamienka) je zapísané v osmičkovej pozičnej číselnej sústave a každá jeho číslica je kódovaná trojbitovým celým číslom bez znamienka.

osmičková číslica	0	1	2	3	4	5	6	7
kódové slovo	000	001	010	011	100	101	110	111

Číslo $(7352)_8$ je binárne kódované vektorom 111011101010. Každý binárny vektor dĺžky $3n$ predstavuje jedno osmičkové číslo, t.j. celkový počet celých čísel zobraziteľných pomocou n-miestnych osmičkových čísel je $8^n = 2^{3n}$. Výhodou zápisu čísel v osmičkovej (podobne v šestnástkovej) sústave je ľahký prevod medzi binárnym a osmičkovým zápisom: pri prevode z osmičkovej do binárnej sústavy stačí nahradiť každú osmičkovú číslicu jej 3-bitovým binárnym zápisom a opačne, binárne celé číslo bez znamienka stačí v prípade potreby doplniť zľava nulami tak, aby jeho dĺžka bola deliteľná tromi, rozdeliť na trojice bitov a každej trojici priradiť osmičkovú číslicu.

2.3.5 Šestnástkové čísla (hexadecimal numbers)

Tento formát sa zakladá na rovnakom princípe ako predchádzajúci: šestnástkové číslice sa zakódujú ako 4-bitové celé čísla bez znamienka. Na označenie prvých 10 číslic sa používajú desiatkové číslice, zostávajúcich 6 šestnástkových číslic je zapísaných pomocou písmen A-F. Kódovanie šestnástkových číslic je uvedené v nasledujúcej tabuľke:

hex	0	1	2	3	4	5	6	7
bin	0000	0001	0010	0011	0100	0101	0110	0111
hex	8	9	A	B	C	D	E	F
bin	1000	1001	1010	1011	1100	1101	1110	1111

Pri prevode šestnástkových čísel do binárnej sústavy stačí nahradiť každú číslicu šestnástkového čísla nahradiť jej 4-bitovým kódovým slovom. Opačne, binárne zapísané celé číslo bez znamienka sa v prípade potreby zľava doplní nulami tak, aby celkový počet číslic bol deliteľný 4, potom sa postupnosť binárnych číslic rozdelí na štvorice a tým sa priradia hexadecimálne číslice:

$$(9D564F)_{16} = (100111010101011001001111)_2.$$

2.3.6 Excess code

Tento formát určuje binárny blokový kód na reprezentáciu celých čísel, ktorý sa najčastejšie používa na reprezentáciu exponentu vo formáte reálne číslo v pohyblivej rádovej čiarke [2]. Pri vytváraní kódového slova reprezentujúceho celé číslo j sa k číslu j pripočíta konštanta a (excess). Hodnota a je pre n-bitový excess kód najčastejšie rovná 2^{n-1}. V nasledujúcej tabuľke je uvedený trojbitový excess 4 kód. Všimnite si dve veci: výsledkom kódovania je formálne celé číslo bez znamienka a namiesto zápornej nuly máme nové číslo, -4, reprezentované nulovým vektorom (slovom dĺžky 3).

desiatkové číslo	binárny zápis	excess desiatkovo	excess binárne
$+3$	011	7	111
$+2$	010	6	110
$+1$	001	5	101
0	000	4	100
-1	101	3	011
-2	110	2	010
-3	111	1	001
-4	$--$	0	000

[2] O tomto formáte budeme hovoriť v časti **??**

2.3.7 Jednotkový doplnok

Jednotkový doplnkový kód celých čísel sa používa ako základ pre binárny doplnkový kód, samostatné použitie je síce možné, ale zriedkavé. Transformácia celého čísla vo formáte celé číslo so znamienkom/bez znamienka na celé číslo vo formáte v jednotlovom kóde je jednoduché, stačí negovať všetky bity vstupného čísla, t.j. nuly nahradiť jednotkami a jednotky nulami. Opačná transformácia je tiež jednoduchá, na prevod čísla z jednotkového doplnkového kódu do pôvodného formátu stačí negovať všetky jeho bity. Teoreticky by bolo možné previesť číslo (dokonca aj ľubovoľný binárny vektor) do jednotkového doplnkového kódu, ale z praktického hľadiska to nemá zmysel, pretože by bolo potrebné vytvárať zakaždým iné obvody na realizáciu aritmetických operácií nad takto zakódovanými číslami.

2.3.8 Binárny doplnok

Binárny doplnkový kód je binárny blokový kód na reprezentáciu kladných a záporných čísel. Kladné čísla v binárnom doplnkovom kóde sú kódované rovnako, ako kladné celé čísla so znamienkom, na prevod záporných sa používa nasledujúca transformácia:

záporné celé číslo	1101
zneguj všetky bity okrem znamienkového	1010
pripočítaj 1	0001
výsledok	1011

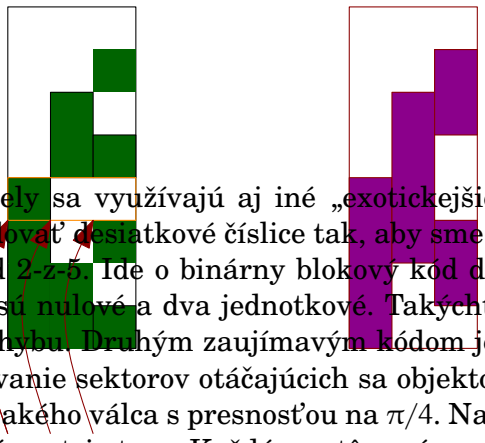
Ak by sme pri prevode záporného čísla použili jeho kladnú reprezentáciu (v našom príklade 0101), druhý krok transformácie by bol: vytvor jednotkový doplnok čísla. V nasledujúcej tabuľke sú uvedené 4-bitové celé čísla so znamienkom a tie isté čísla v jednotkovom aj binárnom doplnkovom kóde.

desiatkový zápis	binárny so znamienkom	jednotkový doplnok	binárny doplnok
$+3$	011	100	011
$+2$	010	101	010
$+1$	001	110	001
$+0$	000	111	000
-1	101	010	111
-2	110	011	110
-3	111	000	101
-4	$--$	$--$	100

Upozorňujeme ešte raz, že binárny doplnkový kód sa vytvára pripočítaním jednotky k jednotkovému doplnku **kladného** čísla a nie záporného; problematickú -4 dostaneme tak, že zoberieme binárne vyjadrenú $4 : 100$, znegujeme všetky jej bity 011, k výsledku pripočítame 1 a dostávame 100. Motiváciou pre zavedenie tohto pomerne zložitého kódovania bolo zmenšenie počtu elementárnych operácií - odčítanie celých čísel bolo možné nahradiť sčítaním.

Obr. 2.1: Prirodzené kódovanie a Grayov zrdkadlový kód

2.3.9 Iné kódovania celých čísel



Na špeciálne účely sa využívajú aj iné „exotickejšie“ formáty zápisu celých čísel. Ak potrebujeme kódovať desiatkové číslice tak, aby sme zaistili ochranu pred chybami, môžeme použiť kód 2-z-5. Ide o binárny blokový kód dĺžky 5, ktorého slová sa vyznačujú tým, že tri bity sú nulové a dva jednotkové. Takýchto kombinácií je $\binom{5}{2} = 10$ a kód dokáže odhaliť 1 chybu. Druhým zaujímavým kódom je tzv. Grayov zrkadlový kód. Ten je vhodný na kódovanie sektorov otáčajúcich sa objektov. Predstavte si, že potrebujete určiť natočenie nejakého válca s presnosťou na $\pi/4$. Na obvode válca je umiestnená čítacia plocha rozdelená na tri stopy. Každá zo stôp má pevne umiestnenú čítaciu hlavu ktorá je schopná rozpoznať, či je oblasť pod ňou svetlá (0) alebo tmavá (1). Na obrázku 2.1 s zobrazené dve riešenia: ľavej strane je na kódovanie pozície použité prirodzené kódovanie celých čísel (000 - 111), na pravej strane Grayov kód dĺžky 3. Problém nastáva ak pri prirodzenom kódovaní hlavice budú čítať z rozhrania dvoch oblastí. Keďže napr. po 000 nasleduje 111 a hlavice môžu prečítať časť informácie z predchádzajúcej a časť z nasledujúcej oblasti, t.j. v tomto prípade môže byť výsledkom akékoľvej binárne číslo dĺžky 3. Grayov zrkadlový kód sa vyznačuje tým, že za sebou idúce kódové slová sa odlišujú práve v jednom znaku, t.j. pri čítaní z hranice oblastí môže vzniknúť len jedna chyba. Grayov zrkadlový kód sa konštruuje rekurzívne 2.1: v prvom stĺpci je Grayov zrkadlový kód dĺžky 1, na končtrukciu Grayovho zrkadlového kódy dĺžky 2 sme okopírovali tabuľku Grayovho zrkadlového kódu dĺžky 1 v opačnom poradí (zrkadlový obraz) a slovám v hornej polovici tabuľky pripísali prefix 0, slovám v dolnej polovici tabuľky prefix 1. Grayov zrkadlový kód dĺžky 3 je v treťom stĺpci.

		000
		001
	00	011
0	01	010
1	11	110
	10	111
		101
		100

Tabuľka 2.1: Grayov zrkadlový kód

2.4 Aritmetika celých čísel

2.5 Reálne čísla

Reálne čísla[3] sa v počítačoch zazanamenávajú v dvoch základných formátoch: v pevnej rádovej čiarke a pohyblivej rádovej čiarke.

2.5.1 Reálne čísla v pevnej rádovej čiarke

Pre reprezentáciu rádovej čiarky sa nezavádza žiaden nový symbol. Reálne číslo vo formáte pevnej rádovej čiarky je binárny reťazec, rozdelený na dve časti - celočíselnú a zlomkovú. Formát určuje, koľko bitov čísla je určených na celočíselnú časť (vrátane znamienkového bitu) a koľko reprezentuje zlomkovú časť. Uvažujme napríklad 12-bitové číslo, ktoré má 8-bitovú celočíselnú a 4-bitovú zlomkovú časť:

$$\underbrace{0}_{\text{znamienko}}\ \underbrace{1100111}_{\text{celá časť}}\ \underbrace{1010}_{\text{zlomková časť}}$$

Poznámka Toto číslo sme vytvorili z celého čísla vo formáte celé číslo so znamienkom tým, že sme ho definovali ako reálne číslo a určili polohu rádovej čiarky. Keďže určenie polohy rádovej čiarky (meranej sprava), napríklad m zodpovedá deleniu pôvodne celého čísla hodnotou 2^m (pre desiatkové 10^m, analogicky pre osmičkové a šestnástkové čísla hodnotami 8^m, resp. 16^m), podobným spôsobom môžeme vytvoriť formáty čísel v pevnej rádovej čiarke aj z iných formátov celých čísel.

Aditívne operácie s reálnymi číslami v pevnej rádovej čiarke si vyžadujú, aby oba argumenty boli v rovnakom formáte, t.j. aby rádové čiarky boli na rovnakej pozícii.

[3]Pojem reálne číslo je zavádzajúci. Počítač má na zobrazenie čísla len konečný počet bitov, a tak všetky čísla, ktoré sa v počítači dajú zaznamenať, majú len konečný binírny rozvoj, a teda sú v skutočnosti racionálne čísla. Pojem reálne číslo sa však natoľko zaužíval, že pokus o korekciu nemá nádej na úspech a viedol by k nedorozumeniam. Preto budeme používať na označenie čísel, ktoré majú aj zlomkovú časť, pojem reálne čísla.

2.5.2 Reálne čísla v pohyblivej rádovej čiarke

Nevýhodou formátu pevnej rádovej čiarky je, že rozsah zobraziteľných čísel je pevný a že sa v ňom nedajú zobrazovať veľmi veľké, ani veľmi malé čísla.Uvažujme napríklad 20 bitové reálne číslo s 12 bitovou celou časťou a 8 bitovou zlomkovou časťou. Najväčšie zobraziteľné číslo je približne $2^{11} = 2096$ a najmenšie (kladné) číslo je $2^{-8} \doteq 0,004$. Ak by sme aj mohli posúvať rádovú čiarku doprava alebo doľava, v tomto formáte nezapíšeme väčšie číslo ako $2^{19} - 1$ a menšie kladné číslo ako $2^{-23} \doteq 1.2 \cdot 10^{-7}$. Riešením je zápis čísel v tzv semilogaritmickom tvare, napr.

$$9.109 \times 10^{-31}$$

Číslo v semilogaritmickom tvare pozostáva zo znamienka (v našom príklade je kladné a nezapisuje sa, hodnoty čísla, tzv. mantisy, zapísanej v tvare reálneho čísla v pevnej rádovej čiarke: 9.109, základu 10 a exponentu, ktorý je v tvare celé číslo so znamienkom: -31. Pri zápise v počítači sa snažíme čo najviac ušetriť tým, že nebudeme zapisovať hodnoty, ktoré sa nemenia. Na začiatok uvedieme základný zjednodušený tvar reálneho čísla v semilogaritmickom tvare, ako je zaznamenaný v počítači.

0	10000000	1	100

- celková dĺžka čísla je 12 bitov,
- prvý bit zľava je znamienkový, 1 predstavuje záporné znamienko a 0 kladné,
- 7 ďalších bitov predstavuje mantisu, zapísanú v podobe reálneho čísla v pevnej rádovej čiarke. Rádová čiarka je umiestnená za znamienkovým bitom, t.j. mantisa je menšia ako 1,
- 4 bity sú venované exponentu, ktorý je zapísaný vo formáte celé číslo so znamienkom, t.j. 9. bit predstavuje znamienko (záporné) a ostatné 3 (t.j. bity s číslami 10-12) reprezentuje hodnotu exponentu,
- základ, ktorý umocňujeme na exponent je 2, ale keďže je nemenný, nepotrebujeme ho nikde zapisovať.

Hodnota čísla z príkladu je

$$+0.5 \times 2^{-4} = 2^{-5} = 0,03125$$

2.6 Zhrnutie

Obsahom tejto kapitoly bola reprezentácia rôznej informácie v počítači. Podrobnejší zoznam a rozsah požadovaných vedomostí uvádzame nižšie

1. Princíp pozičnej číselnej sústavy. Binárna sústava. Prevody čísel medzi binárnou a desiatkovou sústavou, základné aritmetické operácie v binárnej sústave (sčítanie, odčítanie, násobenie, delenie).

   Požadované vedomosti a zručnosti: rozumieť dokonale princípu pozičnej sústavy, vedieť prevádzať čísla z jednej sústavy do druhej, vedieť vykonávať základné aritmetické operácie nad binárne zapísanými celými číslami.

2. Typy údajov v počítači: textové, číselné, riadiace, obrazové, zvukové a iné. Číselné údaje: celé čísla a reálne čísla. Reprezentácia celých čísel: bez znamienka, znamienko absolútna hodnota, jednotkový doplnok, binárny doplnkový kód, exces kód, BCD kód, osmičkové a šestnástkové čísla, Grayov zrkadlový kód. Vyjadrenia celých čísel v rozličných formátoch. Počet zobraziteľných čísel, reprezentácia 0, aritmetické operácie nad celými číslami.

3. Reprezentácia reálnych čísel: pevná rádová čiarka, pohyblivá rádová čiarka (floating point number system, FPNS). FPNS: mantisa, exponent. Normálny tvar mantisy. Reprezentácia exponentu pomocou exces kódu. Reprezentácia 0, najmenšie a najväčšie zobraziteľné (kladné) čísla, počet a hustota zobraziteľných čísel na číselnej osi. Aritmetické operácie s číslami vo FPNS. Úpravy exponentu pri aditívnych operáciách, postnormalizácia. Výnimky z normalizovaného tvaru. Hidden bit technique (technika skrytého bitu); počet a rozsah zobraziteľných čísel, reprezentácia 0, porovnanie so štandardným FPNS.

4. Textová informácia - ASCII a EBCIDIC. Kódovanie špeciálnych znakov (rozšírené kódové množiny).

5. Základná schéma von Neumannovho počítača a princíp činnosti (pamäť: radič pamäte, registre MAR, MBR, pamäťové miesta, I/O systém, CPU (IC, IR, akumulátor), ALU, registre). Inštrukcia: operačný kód a adresová časť. Spôsoby adresácie. Spracovanie inštrukcií. Mikroinštrukcie. Pomocné riadiace údaje: PSW (program status word).

6. Kódovanie zvuku (napr. MP3) a obrazu (rastrová a vektorová grafika, formáty)

Kapitola 3

Booleovské funkcie

V tejto kapitole sa znova budeme zaoberať výrokovou logikou, tentoraz však nebudeme odvodzovať teorémy, ale študovať (zloženým) výrokom priradené logické (Booleovské) funkcie.[1] Ukážeme, že pomocou logických (binárnych) hodnôt možno jednoznačne zapisovať (kódovať) ľubovoľnú informáciu vyjadrenú pomocou reťazcov znakov nad nejakou konečnou abecedou. Ak sa obmedzíme na informáciu zapísanú v podobe konečných reťazcov znakov nad nejakou konečnou abecedou (textovú informáciu), tak spracovanie informácie v podstate predstavuje nejakú transformáciu, ktorá jednému konečnému reťazcu znakov priradí iný (alebo ten istý) konečný reťazec znakov. Ak je takáto transformácia dobre popísaná (napríklad pomocou zobrazenia), tak sa dá popísať aj pomocou logických (Booleovských) funkcií. Booleovské funkcie sa zasa dajú realizovať pomocou fyzikálnych (možno aj biologických) systémov. To znamená, že ak sa dá riešenie nejakého problému popísať pomocou Booleovských funkcií, tak potom (aspoň principiálne[2]) sa dá zostrojiť fyzikálny systém (logický obvod) na riešenie tohto problému. Ako neskôr zistíte pri štúdiu informatiky, Booleovské funkcie nemožno preceňovať; podstata riešenia problému spočíva v nájdení vhodnej transformácie a nie vo vyjadrení tejto transformácie pomocou Booleovských funkcií. Napriek tomuto obmedzeniu sú Booleovské funkcie jedným zo základných objektov, ktoré sa v informatike študujú, a to tak kvôli ich širokému uplatneniu v teoretických disciplínach (výpočtová zložitosť, teória riadiacich systémov, teória testov a iné), ale aj—ako sme už naznačili vyššie—ich úlohe pri návrhu súčasných počítačov a iných digitálnych systémov.

3.1 Základné pojmy

Nech $\mathbb{E} = \{0, 1\}$. *Booleovskou (logickou) premennou* budeme nazývať ľubovoľnú premenú, ktorá nadobúda hodnoty z množiny $\mathbb{E}$ (a žiadne iné). Booleovské premenné budeme označovať symbolmi x, y, z a indexovať. Hodnoty $0, 1$ sa nazývajú *binárne, logické* alebo *Booleovské hodnoty*, alebo aj *logické konštanty*. n*-árnou Booleovskou funkciou* budeme

[1] Táto oblasť diskrétnej matematiky sa nazýva výroková algebra, alebo algebra logiky; zrejme preto, že Booleovské funkcie skúma ako algebraické objekty.

[2] neskôr ukážeme, že takéto riešenia môžu byť tak zložité, že nie sú prakticky realizovateľné

x_1	x_2	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Tabuľka 3.1: Booleovské funkcie dvoch premenných

nazývať ľubovoľné zobrazenie $f : E^n \to E$. n-árna Booleovská funkcia teda priraďuje n-ticiam binárnych hodnôt opäť binárne hodnoty. n-tice vstupných hodnôt Booleovskej funkcie chápeme ako hodnoty n-tice Booleovských premenných. To, že Booleovská funkcia f závisí od premenných $x_1, \ldots, x_n$, zapisujeme symbolicky nasledovne $f(x_1, \ldots, x_n)$. Keďže množina vstupných hodnôt n-árnej Booleovskej funkcie (binárnych vektorov dĺžky n) je konečná a konečná je aj množina hodnôt Booleovskej funkcie, n-árnych Booleovských funkcií je konečne veľa. Označme množinu všetkých n-árnych Booleovských funkcií symbolom $\mathcal{P}_2^n$. Binárnych vektorov dĺžky n je 2^n a n-árnych Booleovských funkcií je toľko, koľko je binárnych vektorov dĺžky 2^n, čiže 2^{2^n}. Pre $n = 2$ teda $|\mathcal{P}_2^2| = 16$ a všetky Booleovské funkcie dvoch premenných sú uvedené v tabuľke 3.1.

Zavedieme niektoré konvencie, ktoré nám zjednodušia zápis Booleovských funkcií. Nech je i prirodzené číslo, $0 \leq i < 2^n$, potom symbolom $\sigma(n, i)$ budeme označovať n-bitový binárny zápis čísla i. Aby sme mohli určiť hodnoty jednotlivých bitov čísla $\sigma(n, i)$, označíme symbolom $\sigma(n, i, j)$, $1 \leq j \leq n$ j-ty bit čísla $\sigma(n, i)$. Tak napríklad $\sigma(4, 12) = 1100, \sigma(4, 7) = 0111, \sigma(3, 7) = 111$; resp. $\sigma(4, 12, 1) = \sigma(4, 12, 2) = 1, \sigma(4, 12, 3) = \sigma(4, 12, 4) = 0$. Je zrejmé, že ak je daný binárny vektor $(x_1, \ldots, x_n)$, tak tomuto vektoru zodpovedá binárne číslo $\sum_{k=1}^{n} x_k * 2^{n-k}$. Na druhej strane binárnemu číslu i môžeme priradiť n-ticu (binárny vektor) $(\sigma(n, i, 1), \sigma(n, i, 2), \ldots, \sigma(n, i, n))$. Túto vzájomne jednoznačnú korešpondenciu medzi binárnymi vektormi dĺžky n a binárnymi číslami budeme využívať tak, že v prípadoch, kde to nepovedie k nedorozumeniu, nebudeme rozlišovať medzi binárnym zápisom n-bitového čísla a jemu zodpovedajúcim n-bitovým vektorom. Potom napríklad $f(\sigma(2, 3))$ bude predstavovať zápis $f(1, 1)$.

Všimnite si tabuľku 3.1, ktorá predstavuje štandardnú tabuľku pravdivostných hodnôt Booleovských funkcií. Riadky tabuľky sú usporiadané vzostupne podľa číselných hodnôt vektorov hodnôt jednotlivých premenných. Ak sa dohodneme na konvencii, že n-árnu Booleovskú funkciu budeme zadávať pomocou štandardnej tabuľky pravdivostných hodnôt 3.2, tak potom je zbytočné zapisovať premenné a ich hodnoty a n-árnu Booleovskú f funkciu možno jednoznačne zadať vektorom hodnôt $(f(\sigma(n, 0)), \ldots, \ldots, f(\sigma(n, 2^n - 1)))$. (Takýto zápis je úsporný a vhodný pre teoretické výpočty a najmä na reprezentáciu Booleovských funkcií v počítači, ale pri väčších hodnotách n je trocha neprehľadný, a

$\tilde{x}$	$f(\tilde{x})$
$\sigma(n, 0)$	$f(\sigma(n, 0))$
$\vdots$	$\vdots$
$\sigma(n, 2^n - 1)$	$f(\sigma(n, 2^n - 1))$

Tabuľka 3.2: Pravdivostná tabuľka n-árnej Booleovskej funkcie

preto budeme na zápis Booleovskej funkcie pri „ručnom“ spracovaní väčšinou používať štandardnú pravdivostnú tabuľku.) Aj vektor hodnôt Booleovskej funkcie predstavuje prirodzené číslo; túto skutočnosť sme využili aj v tabuľke 3.1, kde vektor hodnôt Booleovskej funkcie f_i je $\sigma(4, i)$, čo nám zjednoduší odvolávanie sa na jednotlivé funkcie.

Prikročíme teraz ku skúmaniu Booleovských funkcií dvoch premenných. Ak sa pozrieme na tabuľku 3.1, zistíme, že obsahuje Booleovské funkcie, ktoré „nereagujú“ na zmeny svojich vstupných premenných. Krajným prípadom sú funkcie f_0, f_{15}; prvá nadobúda pre všetky vstupné hodnoty 0, druhá 1. Tieto funkcie nezávisia od hodnôt svojich premenných a predstavujú *konštantné funkcie*, alebo stručne—*konštanty* 0 a 1. Iným príkladom je funkcia f_3, ktorá na výstupe kopíruje hodnoty vstupnej premennej x_1. To znamená, že medzi Booleovskými funkciami dvoch premenných existujú aj funkcie, ktoré by sa dali zapísať ako Booleovské funkcie jednej, resp. žiadnej premennej.

Definícia 3.1.1. *Nech je daná n-árna Booleovská funkcia $f(x_1, \ldots, x_n)$. Potom budeme hovoriť, že Booleovská funkcia $f(x_1, \ldots, x_n)$* podstatne závisí *od premennej x_i, $i \in \{1, \ldots, n\}$, ak existuje taký vektor hodnôt $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n$ premenných $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$, že*

$$f(a_1, \ldots, a_{i-1}, 0, a_{i+1}, \ldots, a_n) \neq f(a_1, \ldots, a_{i-1}, 1, a_{i+1}, \ldots, a_n).$$

Ak Booleovská funkcia f podstatane závisí od premennej x_i, tak premennú x_i nazývame podstatnou premennou *Booleovskej funkcie f. V opačnom prípade premennú x_i nazývame fiktívnou premennou Booleovskej funkcie f.*

Príklad 3.1. *Funkcia $f_{12}(x_1, x_2)$ má podstatnú premennú x_1, lebo $f_{12}(0, 0) = 1$ a $f_{12}(1, 0) = 0$, ale premenná x_2 je fiktívna, lebo $f_{12}(0, x_2) = 1$ a $f_{12}(1, x_2) = 0$.*

Úloha 3.1. *Zistite, ktoré z Booleovských funkcií $f_0, \ldots, f_{15}$ podstatne závisia od dvoch, jednej a žiadnej premennej!*

Príklad 3.2. *Čo sa stane, ak n-árnu Booleovskú funkciu, ktorá podstatne závisí od všetkých svojich premenných. rozšírime o jednu fiktívnu premennú? Ilustrujeme to na príklade, ktorý potom zovšeobecníme. Uvažujme napríklad Booleovskú funkciu $f = f_7$ s premennými x, y a pridajme k nim tretiu premennú, z. Pridanie fiktívnej premennej spôsobilo, že sa každý riadok pôvodnej pravdivostnej tabuľky nahradil dvoma riadkami, v ktorých sú hodnoty premenných x, y rovnaké a tretia premenná z v prvom riadku nadobúda hodnotu 0 a v druhom hodnotu 1, ale hodnota funkcie f závisí len od hodnôt premenných x, y, tabuľka 3.3.*

x	y	z	$g(x, y, z)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Tabuľka 3.3: Pridanie fiktívnej premennej

Úloha 3.2. *Koľko je ternárnych Booleovských funkcií? Koľko z nich podstatne závisí od všetkých troch premenných?*

Zistili sme, že pridávaním, či vyškrtávaním fiktívnych premenných sa síce zväčšuje (zmenšuje) veľkosť tabuľky pravdivostných hodnôt funkcie, ale samotná funkcia sa nemení: pre ten istý súbor hodnôt svojich podstatných premenných dáva vždy tú istú hodnotu bez ohľadu na to, aké hodnoty nadobúdajú jej fiktívne premenné. To nám umožňuje zjednodušiť skúmanie Booleovských funkcií; namiesto toho, aby sme uvažovali Booleovské funkcie s rozličným počtom premenných, môžeme ich doplniť fiktívnymi premennými a skúmať ich všetky napríklad ako n-árne Booleovské funkcie.

Skôr ako prikročíme ku skúmaniu vlastností Booleovských funkcií, ukážeme ako možno pomocou Booleovských funkcií zapísať funkciu definovanú na konečných, ale nie binárnych množinách.

Príklad 3.3. *Nech je* $A = \{a, b, c, d, e, f\}, B = \{✠, ♠, †, ♢\}$ *a zobrazenie* $F : A \to B$ *je definované pomocou nasledujúcej tabuľky:*

x	a	b	c	d	e	f
$F(x)$	✠	♠	†	♢	†	✠

Každému prvku množiny A *priradíme binárny vektor dĺžky* 3 *a prvky množiny* B *zakódujeme pomocou binárnych vektorov dĺžky* 2*:*

x	a	b	c	d	e	f
	000	001	010	011	100	101

y	✠	♠	†	♢
	00	01	10	11

Zobrazenie $F : A \to B$ *vyjadríme pomocou dvoch ternárnych Booleovských funkcií* g_1, g_2*: Všimnite si posledné dva riadky tabuľky 3.4. Definičný obor zobrazenia* F *je 6 prvkový. Na rozlíšenie 6 hodnôt potrebujeme binárne vektory dĺžky 3. Ale binárnych vektorov dĺžky 3 je 8. Prvých 6 riadkov tabuľky 3.4 popisuje pomocou dvoch ternárnych funkcií* g_1, g_2 *zobrazenie* F*, posledné dva riadky zodpovedajú tým vektorom hodnôt premenných* x_1, x_2, x_3*, na ktorých nie sú funkcie* g_1, g_2 *definované. Ako sa takýto problém rieši, budeme skúmať pri konštrukcii disjuntívnych normálnych foriem pre neúplne zadané Booleovské funkcie.*

x	x_1	x_2	x_3	g_1	g_2	$F(x)$
a	0	0	0	0	0	✠
b	0	0	1	0	1	♠
c	0	1	0	1	0	†
d	0	1	1	1	1	♢
e	1	0	0	1	0	†
f	1	0	1	0	0	✠
–	1	1	0	–	–	–
–	1	1	1	–	–	–

Tabuľka 3.4: Zobrazenie F vyjadrené pomocou Booleovských funkcií g_1, g_2

a_1	a_0	b_1	b_0	c_2	c_1	c_0	$a+b=c$		
0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0	1	1
0	0	1	0	0	1	0	0	2	2
0	0	1	1	0	1	1	0	3	3
0	1	0	0	0	0	1	1	0	1
0	1	0	1	0	1	0	1	1	2
0	1	1	0	0	1	1	1	2	3
0	1	1	1	1	0	0	1	3	4
1	0	0	0	0	1	0	2	0	2
1	0	0	1	0	1	1	2	1	3
1	0	1	0	1	0	0	2	2	4
1	0	1	1	1	0	1	2	3	5
1	1	0	0	0	1	1	3	0	3
1	1	0	1	1	0	0	3	1	4
1	1	1	0	1	0	1	3	2	5
1	1	1	1	1	1	0	3	3	6

Tabuľka 3.5: Súčet binárne kódovaných čísel

V úvode tejto kapitoly sme sa zaoberali reprezentáciou celých čísel pomocou binárnych vektorov. V nasledujúcom príklade popíšeme sčítanie prirodzených čísel pomocou Booleovských funkcií 4 premenných.

Príklad 3.4. *Nech* $0 \leq a, b \leq 3$ *sú dve prirodzené čísla; a nech* $a+b=c$ *Zapíšeme všetky tri čísla binárne;* $a=(a_1a_2)_2; b=(b_1b_2)_2$ *a* $c=(c_2c_1c_0)_2$. *Súčet binárne kódovaných čísel* a, b *je popísaný v tabuľke 3.5*

3.2 Skladanie Booleovských funkcií

Ako sme videli v predchádzajúcom príklade, pomocou Booleovských funkcií je možné popísať sčítanie celých čísel. Podobne by sme mohli realizovať násobenie celých čísel, celočíselné delenie (DIV) a modulárne delenie (MOD). Vhodným kódovaním (napríklad vyhradením jedného bitu na kódovanie znamienka) by sme mohli zaviesť záporné čísla a rozšíriť aritmetiku z prirodzených čísel na obor celých čísel. Ak by sme sa dohodli, že (napríklad) v $2n$-bitovom vektore bude prvých n bitov reprezentovať celočíselnú časť a ostávajúcich n bitov—zlomkovú časť čísla, môžeme kódovať racionálne čísla[3] a pomocou Booleovských funkcií popísať aritmetické operácie nad racionálnymi číslami. Principiálne s tým nie sú žiadne problémy. Tie vznikajú, keď by sa teoretické riešenie malo prakticky implementovať. Pomocou 16 bitov môžeme zapísať celé čísla bez znamienka z intervalu $0, \ldots, 65535$, čo na praktické účely asi nebude stačiť. Ak by sme chceli popísať sčítanie dvoch 16 bitových čísel spôsobom uvedeným v príklade 3.5, potrebovali by sme na to 17 Booleovských funkcií o 32 premenných a tabuľku, ktorá by mala $2^{32} = 4.294.967.296$ riadkov. To asi nie je schodné riešenie. Naviac, ak by sa mal pre každú Booleovskú funkciu

[3] v programovacích jazykoch sa označujú ako reálne čísla, t.j. typ REAL

funkcia	# premenných	názov	formula	označenie
$f_0(x,y)$	0	konštanta 0	0	0
$f_1(x,y)$	2	konjunkcia	$x\&y$	AND
$f_3(x,y)$	1	identita/projekcia	x	n.a.
$f_6(x,y)$	2	súčet modulo 2	$x \oplus y$	XOR
$f_7(x,y)$	2	disjunkcia	$x \vee y$	OR
$f_8(x,y)$	2	Pierceova funkcia	$\neg(x \vee y)$	NOR
$f_9(x,y)$	2	ekvivalencia	$x \equiv y$	n.a.
$f_{12}(x,y)$	1	negácia	$\neg x$	NOT
$f_{13}(x,y)$	2	implikácia	$x \Rightarrow y$	n.a.
$f_{14}(x,y)$	2	Shefferova funkcia	$\neg(x\&y)$	NAND
$f_{14}(x,y)$	0	konštanta 1	1	1

Tabuľka 3.6: Elementárne Booleovské funkcie

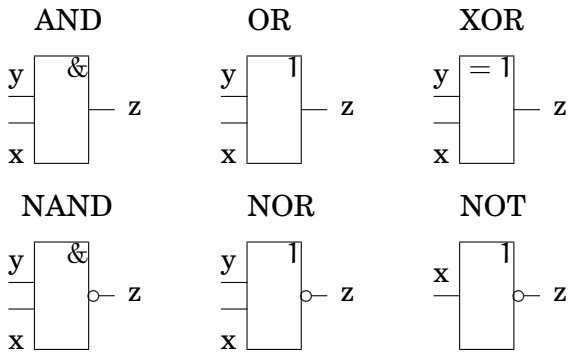
navrhovať systém (logický obvod), ktorý by ju realizoval, bolo by potrebné navrhnúť a vyrobiť veľký počet rozličných špecifických obvodov. To je neekonomické a prakticky nerealizovateľné. V praxi[4] sa preto používa iný prístup: hľadá sa nejaká rozumná množina základných Booleovských funkcií, pomocou ktorých je možné vyjadriť všetky ostatné Booleovské funkcie. Pre tieto základné Booleovské funkcie sa zostroja *logické obvody*, ktoré ich realizujú. Pre Booleovskú funkciu, ktorú potrebujeme realizovať, sa najprv nájde vyjadrenie pomocou elementárnych Booleovských funkcií a na základe tohto vyjadrenia sa z logických obvodov realizujúcich príslušné základné Booleovské funkcie poskladá logický obvod realizujúci danú Booleovskú funkciu. V ďalších častiach tejto kapitoly budeme hľadať odpovede na dve základné otázky

1. ako vybrať množinu základných Booleovských funkcií, aby pomocou nich bolo možné vyjadriť všetky ostatné Booleovské funkcie;
2. keď už je daná množina základných Booleovských funkcií, ako nájsť čo najefektívnejšie vyjadrenie Booleovskej funkcie pomocou základných Booleovských funkcií?

Niektoré spomedzi 16 Booleovských funkcií dvoch premenných[5] sa v matematike a informatike bežne používajú a považujú sa za *elementárne (Booleovské funkcie)*. Ide o Booleovské funkcie $f_0, f_1, f_3, f_6, f_7, f_{12}, f_{13}, f_{15}$ z tabuľky 3.1. Pre jednotlivé elementárne Booleovské funkcie sa zaužívali menej formálne označenia, ktoré budeme aj my v ďalšom výklade používať. Booleovské funkcie f_0, f_{15} budeme nazývať logickými konštantami a označovať symbolmi 0, resp. 1; funkciu f_1 budeme nazývať konjunkciou, atď; kvôli stručnosti a prehľadnosti uvádzame elementárne Booleovské funkcie, ich názvy a symbolické označenia príslušných operátorov v tabuľke 3.6. V tabuľke 3.6 nie sú funkcie f_5, f_{10}, predstavujúce identitu $f_5(x,y) = y$ a negáciu $f_{10}(x,y) = \neg y$, pretože tieto sú už zastúpené funkciami f_3, f_{12}. Na druhej strane, kvôli úplnosti sme medzi elementárne funkcie zaradili aj funkcie f_8, f_{14}, ktorými sa budeme zaoberať až v poslednej časti tejto

[4] ani teória sa neuspokojila konštatovaním, že nejaká úloha je principiálne riešiteľná, ale skúmala zložitosť úloh a hľadala metódy ich optimálneho riešenia.

[5] v skutočnosti medzi nimi sú aj konštanty a funkcie jednej premennej



Obr. 3.1: Hradlá realizujúce elementárne Booleovské funkcie

kapitoly. Operátory pre Shefferovu | a Pierceovu funkciu ↓ sa bežne nepoužívajú, a preto sme obe funkcie vyjadrili pomocou konjunkcie, disjunkcie a negácie. Pre väčšinu elementárnych Booleovských funkcií existujú logické obvody (hradlá, logické členy), ktoré ich realizujú. To znamená, že ak napríklad na vstupy logického obvodu AND (ktorý má dva vstupy a jeden výstup) privedieme signály zodpovedajúce logickým hodnotám p, q, na výstupe AND sa objaví signál zodpovedajúci logickej hodnote $p\&q$. Schématické označenie jednotlivých hradiel je uvedené na obrázku 3.1.

Zložitejšie Booleovské funkcie môžeme dostať skladaním základných Booleovských funkcií.

Definícia 3.2.1. *Nech sú* $f(x_1, x_2, \ldots, x_n), g_1(y_1, \ldots, y_m), g_2(y_1, \ldots, y_m), \ldots, g_n(y_1, \ldots, y_m)$ *Booleovské funkcie. Booleovskú funkciu*

$$F(y_1, \ldots, y_m) = f(g_1(y_1, \ldots, y_m), g_2(y_1, \ldots, y_m), \ldots, g_n(y_1, \ldots, y_m)) \tag{3.1}$$

nazveme zloženou funkciou funkcií $f, g_1, g_2, \ldots, g_n$. *Booleovská funkcia* $f(x_1, x_2, \ldots, x_n)$ *sa nazýva vonkajšou a Booleovské funkcie* $g_1(y_1, \ldots, y_m), g_2(y_1, \ldots, y_m), \ldots, g_n(y_1, \ldots, y_m)$ *vnútornými funkciami zloženej Booleovskej funkcie* F.

Poznámka. Keďže Booleovská funkcia je zobrazenie $\{0,1\}^n \to \{0,1\}$ skladaním Booleovských funkcií dostaneme opäť Booleovskú funkciu. Vnútorné funkcie zloženej Booleovskej funkcie by mohli mať rozličné počty rôznych premenných. Zjednotením množín vstupných premenných jednotlivých Booleovských funkcií sme dostali množinu $\{y_1, \ldots, y_m\}$ a potom sme jednotlivé Booleovské funkcie g_i doplnili fiktívnymi premennými na m-árne Booleovské funkcie. Preto sme pri definícii skladania Booleovských funkcií mohli predpokladať, že všetky vnútorné Booleovské funkcie majú rovnaký počet rovnakých premenných (aj keď v niektorých prípadoch fiktívnych).

Zloženú Boleovskú funkciu 3.1 môžeme opäť zadať pomocou tabuľky pravdivostných hodnôt, ktorú vyplníme nasledovne: aby sme vypočítali hodnotu zloženej Booleovskej

x_1	x_2	f_6	f_{11}	f_{14}	F
0	0	0	1	1	0
0	1	1	0	1	1
1	0	1	1	1	0
1	1	0	1	0	1

Tabuľka 3.7: Zložená Booleovská funkcia

funkcie $F(y_1, \ldots, y_m)$ na vektore vstupných hodnôt $\sigma_1, \ldots, \sigma_m$ potrebujeme najprv vypočítať hodnoty $g_1(\sigma_1, \ldots, \sigma_m), \ldots, g_n(\sigma_1, \ldots, \sigma_m)$, dosadiť ich za premenné $x_1, \ldots, x_n$ funkcie f a potom (napríklad pomocou tabuľky pravdivostných hodnôt Booleovskej funkcie f) vypočítame hodnotu zloženej Booleovskej funkcie.

Príklad 3.5. *Nech* $F(x_1, x_2) = f_6(f_{11}(x_1, x_2), f_{14}(x_1, x_2))$. *Pravdivostné tabuľky čiastkových funkcií* f_6, f_{11}, f_{14} *a zloženej funkcie* F *sú uvedené v tabuľke 3.7*

Všimnite si, že zložená funkcia $F(x_1, x_2) = f_5(x_1, x_2)$*; t.j. skladaním Booleovských funkcií, ktoré záviseli od oboch premenných sme dostali Booleovskú funkciu s jednou fiktívnou premennou.*

Úloha 3.3. *Zopakujte si základné vlastnosti elementárnych Booleovských funkcií z 2. kapitoly (asociatívnosť, komutatívnosť, idempotentnosť a i.) a preskúmajte, ktoré z nich sa zachovávajú pri skladaní Booleovských funkcií!*

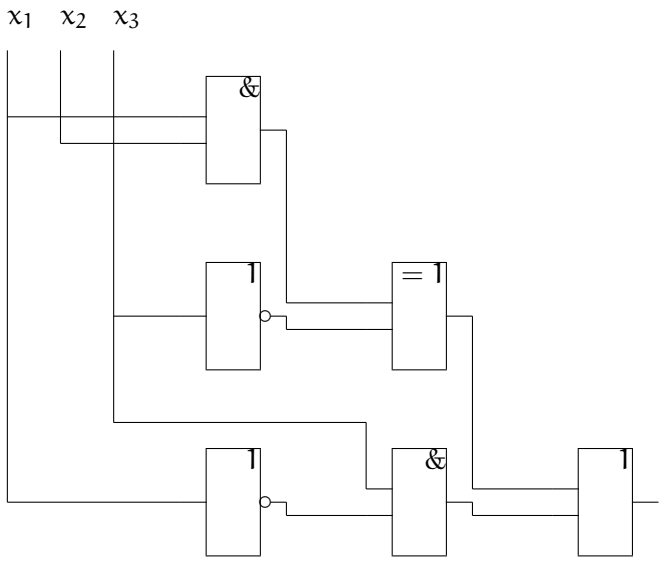
V zápise Booleovskej funkcie pomocou pravdivostnej tabuľky sa stráca informácia o čiastkových Booleovských funkciách, pomocou ktorých je daná Booleovská funkcia vyjadrená. Táto informácia môže však byť užitočná pri skúmaní vlastností Booleovskej funkcie ako aj pri konštruovaní logického obvodu, ktorý ju realizuje. Preto sa popri zápise Booleovských funkcií pomocou pravdivostných tabuliek používa aj ich vyjadrenie v podobe formúl nad nejakou množinou základných Booleovských funkcií. Zavedieme najprv pojem *formuly (algebry logiky)*[6] a potom sa pozrieme na vzťah medzi Booleovskými funkciami a formulami.

Definícia 3.2.2. *Nech je daná množina Booleovskch funkcií* $\mathcal{D} \subseteq \mathcal{P}_2$.

1. *Každý výraz tvaru* $f(x_1, \ldots, x_n)$*, kde* $f \in \mathcal{D}$ *sa nazýva formulou nad* $\mathcal{D}$
2. *Nech je* $f(x_1, \ldots, x_n)$ *Booleovská funkcia z* $\mathcal{D}$ *a* $\mathbf{A_1}, \ldots, \mathbf{A_1}$ *sú formuly nad* $\mathcal{D}$*, potom aj výraz* $f(\mathbf{A_1}, \ldots, \mathbf{A_1})$ *je formulou nad* $\mathcal{D}$
3. *Formulou nad* $\mathcal{D}$ *je ľubovoľný konečný výraz, spĺňajúci podmienky (1) alebo (2). Iné formuly nad* $\mathcal{D}$ *nie sú.*

Formuly algebry logiky budeme označovať písmenami $\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots$. Ak budeme potrebovať vyjadriť, z akých funkcií formula skladaním vznikla, uvedieme ich zoznam v hranatých zátvorkách za názvom formuly; ak potrebujeme vedieť, aké premenné obsahuje, uvedieme ich zoznam v okrúhlych zátvorkách za názvom formuly. Napríklad, ak

[6] V tejto kapitole sa budeme takmer výlučne zaoberať foremulami algebry logiky. Ak to však nepovedie k nedorozumeniu, budem slová „algebry logiky“ kvôli stručnosti vynechávať



Obr. 3.2: Logický obvod počítajúci Booleovskú funkciu F

zloženej funkcii F (3.1) priradíme formulu **A**, tak potom formulu **A** môžeme zapísať ako $\mathbf{A}[f, g_1, \ldots, d_n]$ alebo $\mathbf{A}(y_1, \ldots, y_m)$. Ak ako množinu $\mathcal{D}_1$ zoberieme podmnožinu elementárnych Booleovských funkcií, napríklad

$$\mathcal{D}_1 = \{x\&y, x \vee y, x \oplus y, \neg x\},$$

tak potom formuly nad $\mathcal{D}_1$ zodpovedajú zloženým výrokom, v ktorých premenné x_i zohrávajú úlohu logických premenných alebo elementárnych výrokov. Pravdivostnú hodnotu zložených výrokov vieme bez problémov určiť. Naviac, ak sa nám podarí vyjadriť nejakú funkciu pomocou formuly nad danou množinou $\mathcal{D}_1$, tak potom vieme navrhnúť logický obvod, ktorý bude počítať hodnoty danej Booleovskej funkcie.

Príklad 3.6. *Uvažujme Booleovskú funkciu* $F(x_1, x_2, x_3)$ *zadanú nasledujúcou formulou:*

$$F(x_1, x_2, x_3) = ((x_1\&x_2) \oplus \neg x_3) \vee (x_3\&\neg x_1).$$

Hodnoty tejto Booleovskej funkcie bude počítať logický obvod uvedený na obrázku 3.2.

Teraz upresníme intuitívne chápanie súvislosti medzi Booleovskými funkciami a formulami algebry logiky. Upravíme najprv definíciu hĺbky formuly, ktorú sme zaviedli pre formuly výrokového počtu:

1. Nech $\mathbf{A}(x_1, \ldots, x_n) = f(x_1, \ldots, x_n)$, kde $f(x_1, \ldots, x_n) \in \mathcal{D}$. Potom $\mathbf{hl}(\mathbf{A}) = 0$.
2. Nech $\mathbf{A}(x_1, \ldots, x_n) = f(\mathbf{A}_1, \ldots \mathbf{A}_m)$, kde $f \in \mathcal{D}$. Potom $\mathbf{hl}(\mathbf{A}) = \max_i \mathbf{hl}(\mathbf{A})_i + 1$.

Teraz matematickou indukciou vzhľadom na hĺbku formuly ukážeme, že každej formule algebry logiky možno jednoznačne priradiť Booleovskú funkciu. Nech $\mathbf{A}(x_1, \ldots, x_n)$ je formula nad $\mathcal{D}$.

1. Ak $\mathbf{hl}(\mathbf{A}) = 0$, potom existuje funkcia $f(x_1, \ldots, x_n) \in \mathcal{D}$ taká, že $\mathbf{A}(x_1, \ldots, x_n) = f(x_1, \ldots, x_n)$. Funkcia $f(x_1, \ldots, x_n)$ je Booleovská funkcia priradená formule $\mathbf{A}$.

2. Predpokladajme, že každej formule $\mathbf{A}(x_1, \ldots, x_n)$ nad $\mathcal{D}$, ktorá má hĺbku menšiu ako N dokážeme jednoznačne priradiť Booleovskú funkciu.

3. Nech $\mathbf{hl}(\mathbf{A}) = N$, potom $\mathbf{A}(x_1, \ldots, x_n) = f(\mathbf{A}_1, \ldots, \mathbf{A}_m)$, kde $f(x_1, \ldots, x_m) \in \mathcal{D}$ a $\mathbf{A}_1, \ldots, \mathbf{A}_m$ sú formuly nad $\mathcal{D}$. Keďže hĺbka formúl $\mathbf{A}_1, \ldots, \mathbf{A}_m$ je menšia ako N, každej z týchto formúl dokážeme jednoznačne priradiť Booleovskú funkciu. Označme tieto funkcie (v poradí) $g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n)$. Potom zložená Booleovská funkcia $f(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$, ktorá je zadaná jednoznač ne funkciami $f, g_1, \ldots, g_m$ je Booleovskou funkciou priradenou formule $\mathbf{A}(x_1, \ldots, x_n)$.

Nech $\mathbf{A}(x_1, \ldots, x_n)$ je formula algebra logiky a $f(x_1, \ldots, x_n)$—Booleovská funkcia priradená formule $\mathbf{A}(x_1, \ldots, x_n)$. Potom pre ľubovoľný súbor hodnôt $\sigma_1, \ldots, \sigma_n$ premenných $x_1, \ldots, x_n$ platí $\mathbf{A}(\sigma_1, \ldots, \sigma_n) = f(\sigma_1, \ldots, \sigma_n)$. Preto hovoríme, že formula $\mathbf{A}$ realizuje Booleovskú funkciu $f(x_1, \ldots, x_n)$. Ukážeme teraz na príklade, ako sa formule priraďuje Booleovská funkcia.

Príklad 3.7. *Uvažujme formulu* $\mathbf{A}(x_1, x_2, x_3) = f_7(f_2(x_1, x_3), f_9(x_1, f_{10}(x_2, x_3)))$ *nad množinou všetkých binárnych Booleovských funkcií z tabuľky 3.1. Nájdeme Booleovskú funkciu* $f(x_1, x_2, x_3)$ *priradenú formule* $\mathbf{A}$ *a zapíšeme ju pomocou tabuľky pravdivostných hodnôt. Budeme postupovať nasledovne*

- *najprv vytvoríme tabuľku pravdivostných hodnôt Booleovských funkcií* $f_2(x_1, x_3)$ *a* $f_{10}(x_2, x_3)$. *Obe funkcie doplníme na funkcie troch premenných fiktívnymi premennými (v prvom prípade premennou* x_2*, v druhom —* x_1*) a rozšírené (ternárne) funkcie označíme symbolmi* $y_1(x_1, x_2, x_3) = f_2(x_1, x_3)$*, resp.* $y_2(x_1, x_2, x_3) = f_{10}(x_2, x_3)$.

- *Pre jednotlivé súbory hodnôt* $\sigma_1, \sigma_2, \sigma_3$ *premenných* x_1, x_2, x_3 *vypočítame hodnoty* $y_2(\sigma_1, \sigma_2, \sigma_3)$ *a zostrojíme tabuľku pravdivostných hodnôt funkcie* $f_9(x_1, f_{10}(x_2, x_3))$*, ktorú kvôli stručnosti označíme symbolom* $y_3(x_1, x_2, x_3) = f_9(x_1, y_2)$.

- *Nakoniec v tabuľke pravdivostných hodnôt nájdeme hodnoty* $y_1(\sigma_1, \sigma_2, \sigma_3)$ *a* $y_3(\sigma_1, \sigma_2, \sigma_3)$*, dosadíme ich do funkcie* $f_7 : \ f_7(y_1, y_3)$ *a na základe hodnôt, ktoré pre hodnoty* $\sigma_1, \sigma_2, \sigma_3$ *funkcia* f_7 *nadobúda, zostrojíme pravdivostnú tabuľku funkcie* $f(x_1, x_2, x_3)$.

Každej formule nad množinou $\mathcal{D}$ možno teda jednoznačne priradiť Booleovskú funkciu. Ak by sme dokázali realizovať jednotlivé funkcie z množiny $\mathcal{D}$ logickými obvodmi, potom by sme dokázali vytvoriť aj logický obvod, ktorý by realizoval (počítal hodnoty) danej Booleovskej fukncie. Z tohoto hľadiska je zaujímavá nasledujúca otázka—možno pre danú Booleovskú funkciu nájsť formulu nad množinou $\mathcal{D}$, ktorá ju realizuje? Odpoveď na túto otázku budeme hľadať v nasledujúcich častiach tejto kapitoly.

x_1	x_2	x_3	y_1	x_1	y_2	y_3	f_7
0	0	0	0	0	1	0	0
0	0	1	0	0	1	0	0
0	1	0	0	0	0	1	1
0	1	1	0	0	0	1	1
1	0	0	1	1	1	1	1
1	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0
1	1	1	0	1	0	0	0

Tabuľka 3.8: Tabuľka pravdivostných hodnôt funkcie priradenej formule $\mathbf{A}(x_1, x_2, x_3)$

3.3 Rozklad Booleovských funkcií podľa premenných. Normálne formy

Uvažujme množinu Boleovských funkcií $\mathcal{D}_2 = \{x\&y, x \vee y, \neg x\}$. Ukážeme, že pre ľubovoľnú Booleovskú funkciu možno zostrojiť formulu nad $\mathcal{D}_2$, ktorá danú Booleovskú funkciu realizuje. Podobne ako pre formuly výrokového počtu zavedieme aj pre logické premenné nasledujúce označenie:

$$x^\sigma = x\&\sigma \vee (\neg\sigma)\&(\neg x), \text{kde } \sigma \in \{0, 1\}.$$

Platí

$$x^\sigma = \begin{cases} \neg x & \text{ak } \sigma = 0, \\ x & \text{ak } \sigma = 1, \end{cases}$$

čo sa dá vyjadriť aj nasledovne:

$$x^\sigma = \begin{cases} x & \text{ak } x = \sigma, \\ \neg x & \text{ak } x \neq \sigma. \end{cases}$$

Veta 3.3.1. *(O disjunktívnom rozklade Booleovskej funkcie podľa premenných) Nech je* $f(x_1, \ldots, x_n)$ *ľubovoľná Booleovská funkcia a nech* $0 < m \leq n$. *Potom platí*

$$\begin{aligned} &f(x_1, \ldots, x_m, x_{m+1}, \ldots, x_n) = \\ &\quad = \bigvee_{\sigma_1, \ldots, \sigma_m} x_1^{\sigma_1} \& \ldots \& x_m^{\sigma_m} \& f(\sigma_1, \ldots, \sigma_m, x_{m+1}, \ldots, x_n) \end{aligned} \tag{3.2}$$

pričom disjunkcia sa berie cez všetky možné vektory hodnôt premenných $x_1, \ldots, x_m$.

Dôkaz. Nech je $(a_1, \ldots, a_n)$ ľubovoľný vektor hodnôt premenných $x_1, \ldots, x_n$. Dosadením do 3.2 dostávame:

$$\begin{aligned} &f(a_1, \ldots, a_n) = \\ &\quad = \bigvee_{\sigma_1, \ldots, \sigma_m} a_1^{\sigma_1} \& \ldots \& a_m^{\sigma_m} \& f(\sigma_1, \ldots, \sigma_m, a_{m+1}, \ldots, a_n). \end{aligned} \tag{3.3}$$

Ak pre nejaké i, $1 \leq i \leq m$ $a_i \neq \sigma_i$, tak $a_i^{\sigma_i} = 0$ a každá konjunkcia vo formule 3.3, ktorá obsahuje člen $a_i^{\sigma_i}$ nadobúda hodnotu 0. Pre disjunkciu platí $x \vee 0 \equiv 1$. Ak je funkcia $f(x_1, \ldots, x_n)$ identicky rovná 0, tak vďaka členu $f(\sigma_1, \ldots, \sigma_m, a_{m+1}, \ldots, a_n)$ je každý člen disjunkcie 3.3 nulový a tvrdenie vety je v tomto prípade pravdivé.

Nech $f(x_1, \ldots, x_n)$ nie je konštanta 0. Z disjunkcie vypadnú všetky konjunkcie obsahujúce členy $a_i^{\sigma_i}$ také, že $a_i \neq \sigma_i$ a zostane v nej len člen

$$a_1^{\sigma_1} \& \ldots \& a_m^{\sigma_m} \& f(\sigma_1, \ldots, \sigma_m, a_{m+1}, \ldots, a_n), \text{ kde } a_i = \sigma_i,\ i = 1, \ldots, m.$$

Potom však $a_1^{\sigma_1} \& \ldots \& a_m^{\sigma_m} \equiv 1$ a

$$f(a_1, \ldots, a_n) = 1 \& f(a_1, \ldots, a_n) = f(a_1, \ldots, a_n).$$

□

Poznámka. Hoci sme rozklad Booleovskej funkcie robili vzhľadom na premenné $x_1, \ldots, x_m$, rovnako dobre sme mohli Booleovskú funkciu rozložiť aj podľa premenných $x_{i_1}, \ldots, x_{i_m}$.

Príklad 3.8. *Rozložíme funkciu $x_1 \Rightarrow x_2$ vzhľadom na obe premenné:*

$$\begin{aligned} x_1 \Rightarrow x_2 &= [x_1^0 \&(0 \Rightarrow x_2)] \vee [x_1^1 \&(1 \Rightarrow x_2)] = \\ &= [\neg x_1 \&(0 \Rightarrow x_2)] \vee [x_1 \&(1 \Rightarrow x_2)]. \end{aligned} \tag{3.4}$$

Ale aj funkcie $(0 \Rightarrow x_2)$ a $(1 \Rightarrow x_2)$ z formuly 3.4 možno rozložiť vzhľadom na premennú x_2:

$$\begin{aligned} 0 \Rightarrow x_2 &= [x_2^0 \&(0 \Rightarrow 0)] \vee [x_2^1 \&(0 \Rightarrow 1)] = \\ &= [\neg x_2 \&(0 \Rightarrow 0)] \vee [x_2 \&(0 \Rightarrow 1)]. \end{aligned} \tag{3.5}$$

Podobne

$$\begin{aligned} 1 \Rightarrow x_2 &= [x_2^0 \&(1 \Rightarrow 0)] \vee [x_2^1 \&(1 \Rightarrow 1)] = \\ &= [\neg x_2 \&(1 \Rightarrow 0)] \vee [x_2 \&(1 \Rightarrow 1)]. \end{aligned} \tag{3.6}$$

Dosadíme teraz formuly 3.6 a 3.5 do formuly 3.4 a upravíme:

$$\begin{aligned} (\cancel{x_1} \Rightarrow x_2) &= [\neg x_1 \&[(\neg x_2 \&(0 \Rightarrow 0)) \vee (x_2 \&(0 \Rightarrow 1))] \vee [x_1 \&[(\neg x_2 \&(1 \Rightarrow 0)) \vee (x_2 \&(1 \Rightarrow 1))] \\ &= [\neg x_1 \& \neg x_2 \&(0 \Rightarrow 0)] \vee [\neg x_1 \& x_2 \&(0 \Rightarrow 1)] \vee [x_1 \& \neg x_2 \&(1 \Rightarrow 0)] \vee [x_1 \& x_2 \&(1 \Rightarrow 1)]. \end{aligned}$$

Keďže $(0 \Rightarrow 0) \equiv (1 \Rightarrow 1) \equiv (0 \Rightarrow 1) \equiv 1$, $(1 \Rightarrow 0) \equiv 0$, $(p \& 0) \equiv 0$, $(p \& 1) \equiv p$ a napokon $(p \vee 0) \equiv p$, pre funkciu $(x_1 \Rightarrow x_2)$ po posledných úpravách dostávame nasledujúcu formulu:

$$(x_1 \Rightarrow x_2) = (\neg x_1 \& \neg x_2) \vee (x_1 \& \neg x_2) \vee (x_1 \& x_2).$$

Poznámka. Zápis negácie a konjunkcie v predchádzajúcich formulách je trochu neprehľadný. Vo výrokovej algebre sa používa jednoduchšie označenie. Operátor konjunkcie sa zvykne tam kde to nespôsobí nedorozumenie vynechávať a konjunkcia $x\&y$ sa zapisuje ako xy. Negácia premennej (ale vo všeobecnosti aj formuly) sa označuje vodorovnou čiarou nad premennou (formulou), na ktorú sa negácia vzťahuje: tak napríklad $\neg x_1$ sa zapisuje ako $\overline{x_1}$. Formula, ktorú dostaneme rozkladom implikácie z predchádzajúceho príkladu podľa oboch premenných, bude mať tvar:

$$(x_1 \Rightarrow x_2) = \overline{x_1}\,\overline{x_2} \vee \overline{x_1}x_2 \vee x_1x_2.$$

Dôsledky vety 3.3.1 Uvažujme dva krajné prípady vzhľadom na počet premenných, podľa ktorých sa robí rozklad:

$m = 1$ (Rozklad funkcie podľa jednej premennej)

$$\begin{aligned} &f(x_1,\ldots,x_i,\ldots,x_n) = \\ &\quad = \overline{x_i}\&f(x_1,\ldots,x_{i-1},0,x_{i+1},\ldots,x_n) \vee x_i\&f(x_1,\ldots,x_{i-1},1,x_{i+1},\ldots,x_n). \end{aligned} \tag{3.7}$$

$m = n$ (Rozklad funkcie podľa všetkých premenných)

$$f(x_1,\ldots,x_n) = \bigvee_{\sigma_1,\ldots,\sigma_n} x_1^{\sigma_1}\ldots x_n^{\sigma_n}\&f(\sigma_1,\ldots,\sigma_n). \tag{3.8}$$

Ak do Booleovskej funkcie $f(x_1,\ldots,x_n)$ dosadíme za všetky premenné nejaké hodnoty, napríklad $\sigma_1,\ldots,\sigma_n$, dostávame nulárnu funkciu, čiže konštantu: $f(\sigma_1,\ldots,\sigma_n)$. Ak však $f(\sigma_1,\ldots,\sigma_n) \equiv 0$, potom aj

$$x_1^{\sigma_1}\ldots x_n^{\sigma_n}\&f(\sigma_1,\ldots,\sigma_n) \equiv 0$$

a preto z formuly 3.8 môžeme vynechať všetky konjunkcie, pre ktoré $f(\sigma_1,\ldots,\sigma_n) \equiv 0$. V disjunkcii zostávajú tie členy (konjunkcie),

$$x_1^{\sigma_1}\ldots x_n^{\sigma_n}\&f(\sigma_1,\ldots,\sigma_n) \text{ pre ktoré } f(\sigma_1,\ldots,\sigma_n) = 1.$$

Ale $p\&1 \equiv p$, a preto môžeme výraz $f(\sigma_1,\ldots,\sigma_n)$ v konjunkcii

$$x_1^{\sigma_1}\ldots x_n^{\sigma_n}\&f(\sigma_1,\ldots,\sigma_n)$$

vynechať. Po týchto úpravách dostávame pre Booleovskú funkciu $f(x_1,\ldots,x_n)$ formulu

$$f(x_1,\ldots,x_n) = \bigvee_{\substack{(\sigma_1,\ldots,\sigma_n)\\ f(\sigma_1,\ldots,\sigma_n)=1}} x_1^{\sigma_1}\ldots x_n^{\sigma_n}, \tag{3.9}$$

ktorá sa nazýva *úplnou disjunktívnou normálnou formou (ÚDNF) Booleovskej funkcie* $f(x_1,\ldots,x_n)$.

x_1	x_2	x_3	f	elementárna konjunkcia
0	0	0	1	$\overline{x}_1\overline{x}_2\overline{x}_3$
0	0	1	0	
0	1	0	1	$\overline{x}_1 x_2\overline{x}_3$
0	1	1	1	$\overline{x}_1 x_2 x_3$
1	0	0	0	
1	0	1	1	$x_1\overline{x}_2 x_3$
1	1	0	0	
1	1	1	0	

Tabuľka 3.9: Konštrukcia ÚDNF pre Booleovskú funkciu $f(x_1, x_2, x_3)$

Skôr ako ukážeme, ako sa pre Booleovskú funkciu zostrojuje ÚDNF, zavedieme niekoľko užitočných pojmov, ktoré budeme pri popise a skúmaní disjunktívnych normálnych foriem používať. Výraz $x_i^{\sigma_i}$ ktorý predstavoval premennú x_i alebo jej negáciu $\overline{x}_i$ budeme nazývať *literálom*. Nech sú $x_1, \ldots, x_n$ premenné (nejakej Booleovskej funkcie) a nech

$$x_{i_1}^{\sigma_{i_1}}, \ldots, x_{i_r}^{\sigma_{i_r}}, \text{ kde } 0 \leq r \leq n,$$

sú ľubovoľné literály premenných $x_1, \ldots, x_n$. Potom budeme výraz

$$K = x_{i_1}^{\sigma_{i_1}} \& \ldots \& x_{i_r}^{\sigma_{i_r}}$$

nazývať *elementárnou konjunkciou*[7]. Tam, kde to nepovedie k nedorozumeniu, budeme operátory konjunkcie (&) vynechávať. Hodnotu r nazveme *rangom konjunkcie* K, ak $r = 0$, hovoríme, že konjunkcia K je prázdna a jej hodnota je 1. Nech sú $K_1, \ldots, K_m$ navzájom rôzne konjunkcie. Výraz $\mathbf{D} = K_{i_1} \vee \cdots \vee K_{i_m}$ sa nazýva *disjunktívnou normálnou formou (DNF)*. Hodnota m sa nazýva *dĺžkou disjunktívnej normálnej formy* $\mathbf{D}$. Ak $m = 0$, DNF $\mathbf{D}$ je prázdna a jej hodnota je rovná 0. Úplná disjunktívna normálna forma sa vyznačuje tým, že všetky jej konjunkcie sú elementárne a obsahujú literály všetkých n premenných. V nasledujúcom príklade ukážeme, ako sa konštruuje ÚDNF Booleovskej funkcie.

Príklad 3.9. *Nech je daná Booleovská funkcia* $f(x_1, x_2, x_3) = 10110100$. *Zapíšeme Booleovskú funkciu* f *pomocou tabuľky pravdivostných hodnôt. Jednotkovým (nulovým) riadkom pravdivostnej tabuľky nazveme riadky zodpovedajúce tým vektorom hodnôt premenných, na ktorých Booleovská funkcia* f *nadobúda pravdivostnú hodnotu* 1 *(resp.* 0*). Jednotkovému riadku zodpovedajúcemu vektoru* $\sigma_1, \sigma_2, \sigma_3$ *priradíme elementárnu konjunkciu* $x_1^{\sigma_1} x_2^{\sigma_2} x_3^{\sigma_3}$. *Tieto elementárne konjunkcie pospájame disjunkciami a formula, ktorú takto vytvoríme, je úplnou disjunktívnou normálnou formou Booleovskej funkcie* $f(x_1, x_2, x_3)$

$$f(x_1, x_2, x_3) = \overline{x}_1\overline{x}_2\overline{x}_3 \vee \overline{x}_1 x_2\overline{x}_3 \vee \overline{x}_1 x_2 x_3 \vee \overline{x}_2 x_3.$$

Všimnite si, že elementárna konjunkcia $x_1^{\sigma_1} x_2^{\sigma_2} x_3^{\sigma_3}$ *nadobúda pravdivostnú hodnotu* 1 *len na jedinom vektore;* $\sigma_1, \sigma_2, \sigma_3$. *Úplná disjunktívna normálna forma Booleovskej funkcie*

[7] Elementárnosť konjunkcie spočíva v tom, že jej operandami sú literály. Ak by operandom konjunkcie bola iná zložená formula, výraz by predstavoval konjunkciu, ktorá by však nebola elementárnou konjunkciou, napríklad: $x_1 \& (x_2 \Rightarrow x_3$.)

x_1	x_2	x_3	$\overline{x}_1\overline{x}_2\overline{x}_3$	$\overline{x}_1x_2\overline{x}_3$	$\overline{x}_1x_2x_3$	$x_1\overline{x}_2x_3$	$f(x_1,x_2,x_3)$
0	0	0	1	0	0	0	1
0	0	1	0	0	0	0	0
0	1	0	0	1	0	0	1
0	1	1	0	0	1	0	1
1	0	0	0	0	0	0	0
1	0	1	0	0	0	1	1
1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0

Tabuľka 3.10: ÚDNF Booleovskej funkcie $f(x_1,x_2,x_3)$

x_1	x_2	x_3	$\overline{x}_1\overline{x}_3$	$\overline{x}_1x_2$	$x_1\overline{x}_2x_3$	$f(x_1,x_2,x_3)$
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	1	0	1	1	0	1
0	1	1	0	1	0	1
1	0	0	0	0	0	0
1	0	1	0	0	1	1
1	1	0	0	0	0	0
1	1	1	0	0	0	0

Tabuľka 3.11: Realizácia Booleovskej funkcie $f(x_1,x_2,x_3)$ pomocou DNF $\mathbf{D}^*$

v podstate predstavuje zoznam vektorov hodnôt, na ktorých daná Booleovská funkcia nadobúda hodnotu 1. *Pre Booleovskú funkciu* $f(x_1,x_2,x_3)$ *to názorne ukazuje tabuľka 3.10.*

Je úplná disjunktívna normálna forma jedinou formulou, ktorá realizuje Booleovskú funkciu $f(x_1,x_2,x_3)$ z predchádzajúceho príkladu? Ukážeme, že nie. Keďže

$$\overline{x}_1\overline{x}_2\overline{x}_3 \vee \overline{x}_1x_2\overline{x}_3 = \overline{x}_1\overline{x}_3(x_2 \vee \overline{x}_2) = \overline{x}_1\overline{x}_3$$

a

$$\overline{x}_1x_2\overline{x}_3 \vee \overline{x}_1x_2x_3 = \overline{x}_1x_2,$$

Ako je dobre vidieť z tabuľky 3.11 Booleovskú funkciu $f(x_1,x_2,x_3)$ možno realizovať aj pomocou disjunktívnej normálnej formy

$$\mathbf{D}^* = \overline{x}_1\overline{x}_3 \vee \overline{x}_1x_2 \vee x_1\overline{x}_2x_3.$$

Úloha 3.4. *Zostrojte logický obvod realizujúci Booleovskú funkciu* $f(x_1,x_2,x_3)$*!*

Úloha 3.5. *Zostrojte ÚDNF aspoň pre 5 rozličných Booleovských funkcií troch premenných!*

Keďže pre väčšinu Booleovských funkcií existuje viacero disjunktívnych normálnych foriem, ktoré ich realizujú, na realizáciu danej Booleovskej funkcie sa zvyčajne vyberá optimálna DNF. Existuje viacero kritérií, pomocou ktorých možno posudzovať DNF. Uvedieme dve najčastejšie používané: DNF realizujúca danú Booleovskú funkciu f sa nazýva

1. *minimálnou*, ak zo všetkých DNF realizujúcich danú Booleovskú funkciu obsahuje minimálny počet literálov;

2. *najkratšou*, ak zo všetkých DNF realizujúcich danú Booleovskú funkciu má minimálnu dĺžku (obsahuje minimálny počet konjunkcií).

Zjednodušene povedané, výber minimálnej DNF minimalizuje počet spojení medzi hradlami a výber najkratšej DNF zasa minimalizuje počet hradiel v logickom obvode realizujúcoum danú Booleovskú funkciu f.[8] Základmi optimalizácie DNF sa budeme zaoberať v nasledujúcej časti tejto kapitoly.

Úloha 3.6. *Zostrojte minimálne a najkratšie DNF pre funkcie*

1. $x_1 \Rightarrow x_2$,

2. $f(x_1, x_2, x_3) = (01011011)$,

3. $f(x_1, x_2, x_3) = (11011001)$.

Čo spravíme v prípade, keď máme realizovať konštantu 0? Pre tú síce neexistuje ÚDNF (ak za ÚDNF nepovažujeme prázdnu DNF), ale konštantu 0 môžeme vyjadriť napríklad pomocou formuly $x\&\overline{x}$, Týmto sme uzavreli dôkaz nasledujúcej vety:

Veta 3.3.2. *Ľubovoľnú Booleovskú funkciu možno realizovať pomocou formuly nad množinou Booleovských funkcií* $\{x_1\&x_2, x_1 \vee x_2, \neg x\}$.

Okrem disjunktívnej normálnej formuly existujú aj iné spôsoby vyjadrenia Booleovských funkcií v podobe formúl nad množinou $\{x_1\&x_2, x_1 \vee x_2, \neg x\}$. Jednou z nich je tzv. *konjuktívna normálna forma.*

Veta 3.3.3. *(O konjuktívnom rozklade Booleovskej funkcie) Nech je* $f(x_1, \ldots, x_n)$ *ľubovoľná Booleovská funkcia a nech* $1 \leq m \leq n$. *Potom platí*

$$f(x_1, \ldots, x_m, x_{m+1}, \ldots, x_n) = \\ = \bigwedge_{\sigma_1, \ldots, \sigma_m} x_1^{\overline{\sigma}_1} \vee \cdots \vee x_m^{\overline{\sigma}_m} \vee f(\sigma_1, \ldots, \sigma_m, x_{m+1}, \ldots, x_n) \tag{3.10}$$

kde symbol $\wedge$ *označuje konjunkciu a konjunkcia sa berie cez všetky možné vektory hodnôt premenných* $x_1, \ldots, x_n$.

Dôkaz. Veta sa dokazuje analogický ako veta 3.3.1. □

[8] zjednodušenie spočíva v predpoklade, že sa v logickom obvode použijú hradlá, ktoré majú dostatočný počet vstupov na realizáciu elementárnej konjunkcie, resp. disjunkcie všetkých konjunkcií danej DNF. Ak sa použijú hradlá AND a OR s dvoma vstupmi a jedným výstupom, vzťah medzi dĺžkou DNF a počtom hradiel bude zložitejší.

Dôsledok. Pre ľubovoľnú Booleovskú funkciu $f(x_1,\ldots,x_n) \neq 1$ platí

$$f(x_1,\ldots,x_n) = \bigwedge_{\substack{\sigma_1,\ldots,\sigma_n \\ f(\sigma_1,\ldots,\sigma_n)=0}} x_1^{\overline{\sigma}_1} \vee \cdots \vee x_n^{\overline{\sigma}_n}. \tag{3.11}$$

Formula 3.11 sa nazýva ú*plná konjuktívna forma* (ÚKNF) Booleovskej funkcie $f(x_1,\ldots,x_n)$. ÚKNF pre danú Booleovskú funkciu zostrojíme tak, že každému nulovému riadku pravdivostnej tabuľky (t.j. riadku, prislúchajúcemu vektoru $(\sigma_1,\ldots,\sigma_n)$, pre ktorý $f(\sigma_1,\ldots,\sigma_n) = 0$) priradíme tzv. elementárnu disjunkciu $x_1^{\overline{\sigma}_1} \vee \cdots \vee x_n^{\overline{\sigma}_n}$ a potom tieto elementárne disjunkcie pospájame konjunkciami. ÚKNF pre Booleovskú funkciu z príkladu 3.16 má tvar

$$(x_1 \vee x_2 \vee \overline{x}_3)\&(\overline{x}_1 \vee x_2 \vee x_3)\&(\overline{x}_1 \vee \overline{x}_2 \vee x_3)\&(\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3).$$

Úloha 3.7. *Vyberte si aspoň 5 Booleovských funkcií troch premenných a zostrojte pre ne ÚKNF!*

Úloha 3.8. *Zostrojte konjunktívny rozklad funkcie $x_1 \Rightarrow x_2$ podľa oboch premenných!*

Ako je výhodnejšie zadávať Booleovské funkcie—formulami, alebo pravdivostnými tabuľkami? Formuly nad $\{x_1\&x_2, x_1 \vee x_2, \neg x\}$ nebudú rozhodne zložitejšie ako pravdivostné tabuľky, pretože ÚDNF a ÚKNF zložitosťou približne zodpovedajú pravdivostným tabuľkám. Existujú však Booleovské funkcie, ktoré sa zapisujú pomocou formúl podstatne jednoduchšie v porovnaní so zápisom pomocou pravdivostných tabuliek. Napríklad funkcia $f(x_1,\ldots,x_{100}) = x_1\&\ldots\&x_{100}$. Formula pre realizujúca túto funkciu má 100 literálov a 99 znakov konjunkcie, zatiaľ čo pravdivostná tabuľka (ktorá mimochodom obsahuje jediný jednotkový riadok) má $2^{100} = 1267650600228229401496703205376$ riadkov.

3.4 Minimalizácia disjunktívnych normálnych foriem

V predchádzajúcej časti tejto kapitoly sme ukázali dve podstatné skutočnosti: každú[9] Booleovskú funkciu možno realizovať pomocou disjunktívnej normálnej formy a pre danú Booleovskú funkciu spravidla existuje viacero DNF, ktoré ju realizujú. Keďže priemerná n-árna Booleovská funkcia má približne polovicu jednotkových vektorov, jej ÚDNF bude mať dĺžku $\approx 2^{n-1}$ a bude obsahovať $\approx n2^{n-1}$ literálov. V praktických aplikáciách sa pracuje s binárnymi veličinami o veľkosti niekoľko desiatok bitov (čísla, znaky, inštrukcie). Popísať spracovanie (napríklad) dvoch 32-bitových čísel pomocou Booleovských funkcií a tie realizovať pomocou ÚDNF je príliš zložité a neefektívne. V praxi sa na návrh logických obvodov používa iná metóda: zložitý problém sa rozloží na jednoduchšie, tie sa popíšu pomocou Booleovských funkcií, navrhnú sa pre ne efektívne obvody a riešenie globálneho problému sa „poskladá“ z čiastkových riešení. Rozsah

[9] Pripomíname, že konštantu 0 možno realizovať pomocou prázdnej DNF a konštantu 1 pomocou ÚDNF $x_1 \vee \overline{x}_1$.

a_i	b_i	r_{i-1}	r_i	c_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Tabuľka 3.12: Jednobitová sčítačka

čiastkových problémov je taký malý, že je možná optimalizácia ich riešení. Napríklad sčítanie dvoch n-bitových čísel sa dá realizovať pomocou n jednobitových sčítačiek, z ktorých každá vypočítava súčet troch jednobitových čísel (dvoch jednobitových sčítancov a prenosu z nižšieho rádu). Výsledný $2n$-bitový sumátor bude mať zložitosť (vyjadrenú počtom literálov) $10n$ (pozri príklad 3.10). Ak by sme sčítanie realizovali pomocou logických obvodov vychádzajúcich z DNF $2n$-árnych Booleovských funkcií, potrebovali by sme $n+1$ takýchto Booleovských funkcií a zložitosť takto zostrojeného logického obvodu by bola $\approx n^2 \cdot 2^n$.

Príklad 3.10. *Sumátor so sériovým prenosom. Vstupom sú dve binárne hodnoty* a_i, b_1 *a prenos z predchádzajúceho nižšieho rádu* r_{i-1}. *Výstupom sumátora je súčet* $c_i = a_i \oplus b_1 = a_i\overline{b_i} \vee \overline{a_i}b_i$ *a prenos do vyššieho rádu:* $r_i = a_i b_i \vee a_i r_{i-1} \vee b_i r_{i-1}$. *Počet literálov vo formulách pre výstupnú hodnoty* c_i, r_i *je zhora ohraničený*[10] *číslom* 10.

Ako však nájdeme optimálnu DNF pre danú Booleovskú funkciu? Uvažujme n-árnu Booleovskú funkciu $f(x_1, \ldots, x_n)$. Existuje 3^n rozličných elementárnych konjunkcií premenných $x_1, \ldots, x_n$; pretože pre ľubovoľné $1 \leq i \leq n$ elementárna konjunkcia

- premennú x_i neobsahuje, alebo
- obsahuje premennú x_i, alebo
- obsahuje negáciu premennej x_i; $\overline{x}_i$.

Každá disjunktívna normálna forma je jednoznačne určená výberom konjunkcií—je zrejmé, že každá z 3^n rozličných elementárnych konjunkcií buď patrí alebo nepatrí do DNF. Celkovo je teda možných 2^{3^n} DNF premenných $x_1, \ldots, x_n$. Teoreticky by sme teda mohli postupovať nasledovne: usporiadali by sme všetky DNF (napríklad najprv podľa počtu literálov a potom lexikograficky) a potom by sme postupne preverovali, či DNF realizuje zadanú Booleovskú funkciu alebo nie. Prvá DNF, ktorú by sme takýmto spôsobom našli, by bola minimálna DNF danej Booleovskej funkcie. Žiaľ, principiálne jednoduchá metóda je použiteľná nanajvýš pre funkcie troch premenných, pretože množina konjunkcií, ktoré môžu patriť do DNF je príliš veľká. Prakticky sa tento problém rieši tak, že sa

[10] riešenie, ktoré sme zostrojili, potrebovalo 10 literálov. Nie je vylúčené, že existuje lepšie riešenie, ktoré vystačí s menším počtom literálov. Ale naša konštrukcia zaručuje, že 10 literálov bude určite stačiť.

najprv podstatne zúži množina kandidátov na konjunkcie v DNF a potom sa na zúženú množinu aplikujú sofistikované metódy preberania. Optimalizácia (minimalizácia) DNF je mimoriadne zaujímavá tak z praktického (konštrukcia logických obvodov) ako aj teoretického hľadiska. Mnohé optimalizačné problémy možno previesť na minimalizáciu DNF a tak nájdenie efektívnej metódy na minimalizáciu DNF by pomohlo vyriešiť aj celý rad ťažkých, užitočných a zaujímavých optimalizačných úloh. Preto sa minimalizácia DNF intenzívne študuje od polovice minulého storočia a návrhom rozličných schém realizujúcich Booleovské funkcie a skúmaniu ich zložitosti sa zaoberá samostatná disciplína. My sa teoretických poznatkov o zložitosti Booleovských funkcií dotkneme len okrajovo a sústredíme sa na prezentáciu dvoch metód konštrukcie optimálnych a suboptimálnych DNF Booleovských funkcií; metódu Quine-McCluskey a metódu založenú na Karnaughových mapách.

3.4.1 Geometrické princípy minimalizácie DNF

Minimalizcia DNF má veľmi názornú geometrickú interpretáciu—zodpovedá konštrukcii pokrytia grafu Booleovskej funkcie špeciálnymi podgrafmi. Zavedieme najprv niektoré potrebné pojmy a potom sa budeme zaoberať minimalizáciou DNF.

Definícia 3.4.1. *Nech* $\mathbf{u} = (a_1, \ldots, a_n), \mathbf{v} = (b_1, \ldots, b_n)$ *sú binárne vektory dĺžky* n.

(a) *Počet jednotkových zložiek vektora* $\mathbf{u}$ *budeme nazývať Hammingovou váhou vektora a označovať symbolom* $\mathbf{wt}(\mathbf{u})$.

(b) *Počet zložiek, v ktorých sa vektory* $\mathbf{u}, \mathbf{v}$ *odlišujú, budeme nazývať Hammingovou vzdialenosťou vektorov* $\mathbf{u}, \mathbf{v}$ *a označovať symbolom* $\mathbf{d}(\mathbf{u}, \mathbf{v})$.

Poznámka. Je zrejmé, že $= \mathbf{wt}(\mathbf{u} \oplus \mathbf{v})$.

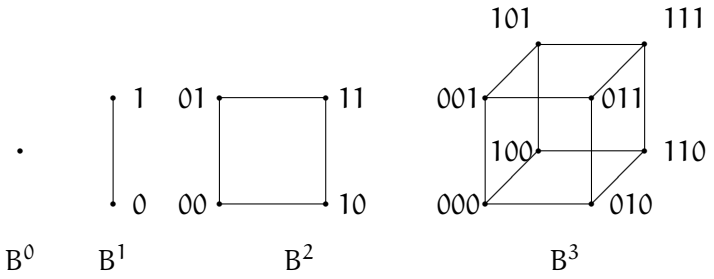
Definícia 3.4.2. *(Graf Booleovskej funkcie).*

(a) n*-rozmernou Booleovskou kockou nazveme graf* $B^n = (V, U)$*, kde* $V = \{\sigma(n, 0), \ldots, \ldots, \sigma(n, 2^n - 1)\}$ *a* $U = \{(v_i, v_j); \mathbf{d}(\sigma(\mathbf{n}, \mathbf{i}), \sigma(\mathbf{n}, \mathbf{j}) = \mathbf{1})\}$.

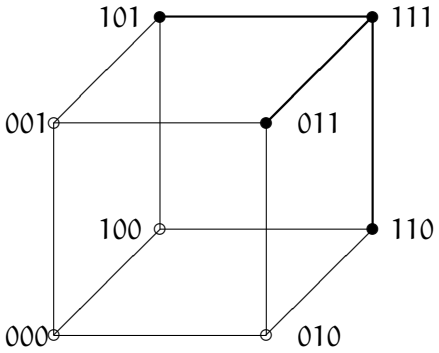
(b) *Nech* $f(x_1, \ldots, x_n)$ *je* n*-árna Booleovská funkcia,* $N_f = \{(a_1, \ldots, a_n); f(a_1, \ldots, a_n) = 1\}$ *je množina jednotkových vektorov funkcie* f. *Vrcholom* $\sigma(n, i)$ *Booleovskej kocky* B^n *priradíme hodnoty* $f(\sigma(n, i))$*; vrcholy zodpovedajúce vektorom z množiny* N_f *nazveme jednotkovými vrcholmi. Grafom Booleovskej funkcie* f *nazveme podgraf indukovaný množinou jej jednotkových vrcholov.*[11]

Príklad 3.11. *Na obrázku 3.16 sú uvedené* $0, 1, 2, 3$ *rozmerné Booleovské kocky. Uvažujme napríklad funkciu* r_i *z tabuľky 3.23. Graf tejto funkcie je znázornený na obrázku 3.4 hrubými čiarami.*

[11] Nech je daný graf $G = (V, U)$ a $V_1 \subseteq V$ je nejaká podmnožina nbožiny vrcholov grafu G. Podgrafom grafu G indukovaným množinou vrcholov V_1 je graf $G_1 = (V_1, U_1)$, kde $U_1 = (V_1 \times V_1) \cap U$ t.j. hranami podgrafu G_1 sú práve tie hrany grafu G, ktoré sú incidentné s vrcholmi z množiny V_1. Graf Booleovskej funkcie je potom $B^n(f) = (V(f), U(f))$; $V(f) = \{\sigma(n, i) \in V; \sigma(n, i) \in N_f$, $U(f) = U \cap V(f) \times V(f)$.



Obr. 3.3: Booleovské kocky



Obr. 3.4: Graf Booleovskej funkcie r_i

Pozrieme sa, ako sa operácie s funkciami prejavia na ich grafoch. Nech sú f, g dve n-árne Booleovské funkcie premenných $x_1, \ldots, x_n$; $B^n(f) = (V_f, U_f)$ a $B^n(g) = (V_g, U_g)$ sú grafy týchto funkcií. Potom

(a) grafom funkcie $\neg f$ je podgraf B^n indukovaný množinou vrcholov $\{0,1\}^n - V_f$;

(b) grafom funkcie $f\&g$ je podgraf B^n indukovaný množinou vrcholov $V_f \cap V_g$;

(c) grafom funkcie $f \vee g$ je podgraf B^n indukovaný množinou vrcholov $V_f \cup V_g$.

Ako sme ukázali v predchádzajúcom príklade, graf Booleovskej funkcie predstavuje iný spôsob reprezentácie Booleovskej funkcie. Určiť hodnotu Booleovskej funkcie f na vektore $\sigma_1, \ldots, \sigma_n$ znamená, zistiť, či jej graf $B^n(f)$ obsahuje vrchol $\sigma_1, \ldots, \sigma_n$. Efektívnosť takéhoto výpočtu závisí od toho, ako efektívne sa podarí charakterizovať graf $B^n(f)$. Jeden z možných spôsobov je zostaviť zoznam všetkých jednotkových vrcholov grafu $B^n(f)$. Takýto prístup však (vzhľadom na počet jednotkových vektorov priemernej Booleovskej funkcie) už pre relatívne malé hodnoty n vedie k značne rozsiahlym zoznamom. Preto sa hľadajú vzťahy medzi jednotkovými vrcholmi grafu $B^n(f)$, ktoré by umožnili zaradiť ich do takých skupín (podmnožín) spĺňajúcich nasledujúce prirodzené požiadavky

- každý jednotkový vrchol grafu $B^n(f)$ patrí do aspoň jednej skupiny vrcholov,
- každá skupina vrcholov sa dá jednoducho charakterizovať a príslušnosť vrcholu do skupiny sa dá efektívne určiť.

Implicitne sa predpokladá, že žiadna skupina neobsahuje nejaký nulový vrchol, pretože to by znamenalo zmenu pôvodnej Booleovskej funkcie.

Takéto riešenie zodpovedá konštrukcii zvláštneho pokrytia grafu $B^n(f)$.

Definícia 3.4.3. *(Vrcholové pokrytie) Nech je daný graf* $G = (V, U)$ *a nech sú* $G_i = (V_i, U_i), i = 1 \ldots, k$ *podgrafy grafu* G. *Potom hovoríme, že*

(a) *graf* $G_i = (V_i, U_i)$ *pokrýva množinu vrcholov* V_i *grafu* G,

(b) *grafy* G_i *tvoria (vrcholové) pokrytie grafu* G, *ak* $\bigcup_i V_i = V$.

Pokrytie na prvý pohľad pripomína rozklad množiny vrcholov na triedy ekvivalencie. Od rozkladu sa však odlišuje tým, že jednotlivé skupiny vrcholov nemusia byť disjunktné.

Teraz je dôležité nájsť podgrafy, pomocou ktorých by bolo možné pokryť graf $B^n(f)$. Pozrieme sa najprv na to, aký je vzťah medzi formulou (DNF) a grafom Booleovskej funkcie, resp. čo zodpovedá elementárnym konjunkciám z DNF Booleovskej funkcie v jej grafe $B^n(f)$. Kvôli názornosti budeme pracovať s trojrozmernou Booleovskou kockou z obrázka 3.16 a budeme vytvárať DNF a grafy rozličných Booleovských funkcií. Začneme konštantou 0. Konštanta 0 nemá žiadne jednotkové vektory a teda jej graf neobsahuje žiadne vrcholy a je prázdny. Ľubovoľnej elementárnej konjunkcii $x_1^{\sigma_2} x_2^{\sigma_2} x_3^{\sigma_3}$ zodpovedá jediný jednotkový vektor $\sigma_1, \sigma_2, \sigma_3$ a jemu prislúchajúci jednotkový vrchol v Booleovskej

kocke B^3. Zaujímavá situácia nastane vtedy, keď sa v kocke vyskytujú susedné (t.j. spojené hranou) jednotkové vrcholy. Napríklad, dvojici susedných (jednotkových) vrcholov $111, 110$ v kocke B^3 odpovedajú elementárne konjunkcie $x_1x_2x_3$ a $x_1x_2\overline{x}_3$ a ÚDNF danej funkcie by mala tvar $x_1x_2x_3 \vee x_1x_2\overline{x}_3$. Použijeme distributívny zákon a upravíme ÚDNF nasledovne:

$$x_1x_2x_3 \vee x_1x_2\overline{x}_3 = x_1x_2(x_3 \vee \overline{x}_3) = x_1x_21 = x_1x_2.$$

Z hľadiska pokrytia to znamená, že dvojicu (jednotkových) vrcholov $111, 110$ môžeme pokryť hranou (jednorozmernou Booleovskou kockou). Zoberne napokon štvoricu (jednotkových) vrcholov ležiacich napríklad na prednej stene kocky B^3: $000, 010, 011, 001$. Táto štvorica je pokrytá dvojrozmernou kockou. V ÚDNF bude uvedenej štvorici jednotkových vrcholov zodpovedať formula

$$\overline{x}_1\overline{x}_2\overline{x}_3 \vee \overline{x}_1x_2\overline{x}_3 \vee \overline{x}_1x_2x_3 \vee \overline{x}_1\overline{x}_2x_3$$

Opakovaným použitím distributívneho zákona túto formulu upravíme na tvar

$$\overline{x}_1(\overline{x}_2\overline{x}_3 \vee x_2\overline{x}_3 \vee \overline{x}_2x_3 \vee x_2x_3) = \overline{x}_1.$$

Teraz už vieme dosť na to, aby sme mohli popísať vzťahy medzi geometrickým modelom Booleovskej funkcie a jej DNF.

(a) elementárnej konjunkcii $x_{i_1}^{\sigma_{i_1}}, \ldots, x_{i_r}^{\sigma_{i_r}}$ zodpovedá množina vrcholov/vektorov tvoriacich jednotkovú podkocku dimenzie $n - r$. Vrcholy/vektory tejto podkocky majú fixované zložky $i_1, \ldots, i_r$, na ktorých nadobúdajú hodnoty $\sigma_{i_1}, \ldots, \sigma_{i_r}$.

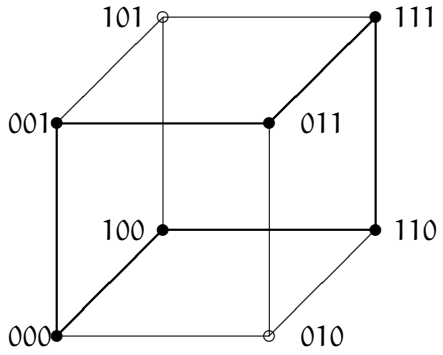
(b) DNF Booleovskej funkcie zodpovedá pokrytie vrcholov jej grafu jednotkovými podkockami.

(c) ÚDNF Booleovskej funkcie zodpovedá pokrytie vrcholov jej grafu jednotkovými podkockami dimenzie 0.

Konštrukcia DNF teda zodpovedá konštrukcii pokrytia vrcholov grafu Booleovskej funkcie jednotkovými podkockami. Jednotkový vrchol však môže byť pokrytý podkockami rozličných dimenzií. Uvažujme napríklad Booleovskú funkciu $f(x_1, x_2, x_3) = (01111111)$. Potom vrchol 111 možno pokryť 0-rozmernou podkockou zodpovedajúcou elementárnej konjunkcii $x_1x_2x_3$, 1-rozmernou podkocou (hranou) zodpovedajúcou elementárnej konjunkcii x_1x_2, dvojrozmernou podkockou zodpovedajúcou elementárnej konjunkcii x_1.[12] Videli sme, že čím väčšia je dimenzia podkocky, tým jednoduchšia je jej prislúchajúca konjunkcia a tým viac jednotkových vrcholov pokrýva. Ak máme možnosť pokryť nejaký jednotkový vrchol podkockami viacerých rozmerov, vyberieme z nich preto podkocku maximálneho rozmeru. Tento pojem je pre ďalší výklad taký dôležitý, že si zaslúži formálnu definíciu.

Definícia 3.4.4. *Nech je* $B^n(f)$ *graf (nejakej) Booleovskej funkcie* f. *Jednotková podkocka* C *grafu* $B^n(f)$ *sa nazýva maximálnou jednotkovou podkockou grafu* $B^n(f)$, *ak v grafe* $B^n(f)$ *neexistuje jednotková podkocka* C' *taká, že* C *je podgrafom* C'.

[12] vrchol 111 možno pokryť aj inými podkockami rozličných rozmerov.



Obr. 3.5: Maximálne podkocky

V teórii DNF sa elementárne konjunkcie zodpovedajúce jednotkovým podkockám grafu $B^n(f)$ nazývajú aj *implikanty*. Tento názov vyplýva z toho, že ak K_i je elementárna konjunkcia zodpovedajúca jednotkovej podkocke C_i grafu $B^n(f)$, tak potom je implikácia $K_i \Rightarrow f$ pravdivá. Uvažujme postupnosť jednotkových podkociek $C_1, \ldots, C_m$ grafu $B^n(f)$, pričom C_i je zároveň podkockou C_{i+1} a postupnosť im prislúchajúcich elementárnych konjunkcií $K_1, \ldots, K_m$. Potom platia implikácie $K_i \Rightarrow K_{i+1}, 1 \leq i < m$ a $K_m \Rightarrow f$. Ak v grafe $B^n(f)$ neexistuje jednotková podkocka, ktorá by obsahovala podkocku C_m, tak potom je C_m maximálna podkocka. Elementárna konjunkcia zodpovedajúca maximálnej podkocke sa nazýva *prostým implikantom* (prime implicant). Význam pojmu implikant si najlepšie uvedomíme vtedy, keď v implikácii $K_i \Rightarrow f$ nahradíme elementárnu konjunkciu K_i elementárnou konjunkciou $K_i' \neq K_i$. Nech $K_i'(a_1 \ldots, a_n) = 1$ a $K_i(a_1 \ldots, a_n) = 0$, potom je implikácia $K_i \Rightarrow f$ na vektore $a_1 \ldots, a_n)$ pravdivá ($0 \Rightarrow 0 \equiv 1$), ale implikácia $K_i' \Rightarrow f$ na vektore $a_1 \ldots, a_n)$ je nepravdivá ($1 \Rightarrow 0 \equiv 0$). Dobrou stratégiou (aspoň na prvý pohľad) pri konštrukcii minimálnej DNF by mohlo byť vytvorenie zoznamu všetkých prostých implikantov Booleovskej funkcie f (resp. maximálnych podkociek grafu $B^n(f)$). Vzhľadom na to, ako sme definovali implikanty, resp. prosté implikanty Booleovskej funkcie, tvorí aj ich disjunkcia DNF Booleovskej funkcie.

Definícia 3.4.5. *Nech je f ľubovoľná Booleovská funkcia a $K_1, \ldots, K_s$ je množina všetkých prostých implikantov funkcie f. Disjunktívna normálna forma*

$$K_1 \vee \cdots \vee K_s$$

sa nazýva skrátenou disjunktívnou normálnou formou Booleovskej funkcie f.

Príklad 3.12. *Na obrázku 3.5 je uvedený graf Booleovskej funkcie $f(x_1, x_2, x_3) = (11011011)$. Booleovská funkcia má 6 prostých implikantov a jej skrátená DNF vyzerá nasledovne:*

$$f(x_1, x_2, x_3) = x_1x_2 \vee x_2x_3 \vee \overline{x}_1x_3 \vee \overline{x}_1\overline{x}_2 \vee \overline{x}_2\overline{x}_3 \vee x_1\overline{x}_3.$$

Predchádzajúci príklad ukazuje, že všetky prosté implikanty zo skrátenej DNF nemusia byť potrebné na realizáciu Booleovskej funkcie. Booleovskú funkciu z predchádzajúceho príkladu by bolo možné realizovať DNF obsahujúcou tri prosté implikanty:

$$f(x_1, x_2, x_3) = x_1x_2 \vee \overline{x}_1x_3 \vee \overline{x}_2\overline{x}_3.$$

Na vylúčenie nadbytočných prostých implikantov zo skrátenej DNF a na konštrukciu novej DNF sa používa nasledujúca induktívna metóda, pri ktorej využijeme korešpondenciu medzi prostými implikantami K_i a maximálnymi podkockami C_i grafu príslušnej Booleovskej funkcie:

1. vyberieme nejaký prostý implikant K_{i_1} a zaradíme ho do DNF;
2. predpokladáme, že v DNF sú už nejaké implikanty $K_{i_1}, \ldots, K_{i_{m-1}}$, $m > 1$ a nech K_{i_m} je nejaký prostý implikant zo skrátenej DNF. Ak existuje vrchol podkocky C_{i_m} ktorý nie je pokrytý podkockami $C_{i_1}, \ldots, C_{i_{m-1}}$, zaradíme K_{i_m} do DNF, v opačnom prípade ho do DNF nezaradíme. Tento krok opakujeme pre všetky prosté implikanty zo skrátenej DNF.

Po ukončení vyššie uvedenej procedúry dostávame DNF, z ktorej nemožno vyradiť žiaden prostý implikant, pretože výsledná DNF by potom už nerealizovala danú Booleovskú funkciu. Takáto DNF, v ktorej sa nevyskytujú nadbytočné prosté implikanty, sa nazýva *iredudantnou DNF*. Je zrejmé, že iredudantných DNF danej Booleovskej funkcie môže byť viacero (pozri 3.5). Napriek tomu konštrukcia minimálnej DNF vyzerá principiálne jednoducho—najprv zostrojíme ÚDNF, potom z ÚDNF skrátenú DNF a eleináciou nadbytočných prostých implikantov dostaneme niekoľko iredudantných DNF, medzi ktorými bude jedna alebo niekoľko minimálnych. Teória je však neúprosná—principiálne jednoduchá metóda sa vo všeobecnom prípade nedá použiť, pretože

- skrátená DNF je veľmi zložitá,
- iredudantných DNF je veľmi veľa,
- univerzálne a efektívne metódy preberania množiny prostých implikantov nie sú známe.

K týmto výsledkom sa ešte vrátime v závere tejto časti, zatiaľ sa však nimi nebudeme zaťažovať a pozrieme sa na prakticky použiteľné metódy konštrukcie skrátenej DNF.[13]

3.4.2 Quine-McCluskeyova metóda

Quineho-McCluskey-ova metóda sa výhodne používa pre funkcie závisiace od 5 a väčšieho počtu premenných. My ju vysvetlíme na príklade funkcie 4 premenných. Nech je daná Booleovská funkcia $f(x_1, x_2, x_3, x_4)$ zadaná pravdivostnou tabuľkou 3.13

Z konštrukcie pokrytí grafov Boleovských funkcií vieme, že spájať možno len susedné vrcholy, t.j. také, ktorých Hammingovská vzdialenosť je 1. Zostrojíme preto zoznam všetkých jednotkových vektorov Booleovskej funkcie f usporiadaných podľa ich Hammingovej váhy. V zozname budeme mať 5 skupín vrcholov: vrcholy/vektory váhy $0, \ldots, 4$, pozri tabuľka 3.14. Výhodou tohto usporiadania je, že susedné vrcholy/vektory

[13] Existencia takýchto metód vôbec nespochybňuje pesimistické teoretické výsledky. Znamená len to, že existujú prípady, pre ktoré sa minimálna DNF dá efektívne zostrojiť a že mnohé praktické problémy patria do tejto kategórie.

x_1	x_2	x_3	x_4	f	x_1	x_2	x_3	x_4	f
0	0	0	0	1	1	0	0	0	1
0	0	0	1	1	1	0	0	1	1
0	0	1	0	1	1	0	1	0	0
0	0	1	1	1	1	0	1	1	0
0	1	0	0	1	1	1	0	0	1
0	1	0	1	0	1	1	0	1	0
0	1	1	0	1	1	1	1	0	0
0	1	1	1	0	1	1	1	1	1

Tabuľka 3.13: Pravdivostná tabuľka Booleovskej funkcie f.

x_1	x_2	x_3	x_4	implikant	číslo	kontrola
0	0	0	0	$\overline{x}_1\overline{x}_2\overline{x}_3\overline{x}_4$	0	
0	0	0	1	$\overline{x}_1\overline{x}_2\overline{x}_3x_4$	1	✓
0	0	1	0	$\overline{x}_1\overline{x}_2x_3\overline{x}_4$	2	✓
0	1	0	0	$\overline{x}_1x_2\overline{x}_3\overline{x}_4$	4	✓
1	0	0	0	$x_1\overline{x}_2\overline{x}_3\overline{x}_4$	8	✓
0	0	1	1	$\overline{x}_1\overline{x}_2x_3x_4$	3	✓
0	1	1	0	$\overline{x}_1x_2x_3\overline{x}_4$	6	✓
1	0	0	1	$x_1\overline{x}_2\overline{x}_3x_4$	9	✓
1	1	0	0	$x_1x_2\overline{x}_3\overline{x}_4$	12	✓
1	1	1	1	$x_1x_2x_3x_4$	15	

Tabuľka 3.14: Quine-McCluskey

sa nachádzajú v susedných skupinách. Všimneme si, že nám vypadla skupina vektorov váhy 3. Teraz budeme kombinovať vektory: v geometrickej interpretácii to znamená, že budeme susedné jednotkové vrcholy pokrývať hranami. Algebraická interpretácia „kombinovania“ vektorov znamená použitie distributívneho zákona a nahradenie dvojice konjunkcií $xK \vee \overline{x}K$ jednou konjunkciou K. Aby sme nestratili prehľad o tom, ktoré premenné sme vynechali, v príslušnom vektore hodnôt nahradíme hodnotu vynechanej premennej znakom "(don't care). Výsledky 1. kola „kombinovania“ vektorov sú uvedené v tabuľke 3.15. Pripomenieme ešte, že ak sme úspešne skombinovali dvojicu vektorov z tabuľky 3.14, oba vektory označíme znakom ✓.

V 1. kole sme našli všetky dvojice jednotkových vrcholov, ktoré bolo možné pokryť hranami. V 2.kole budeme hľadať možnosti pokrytia vrcholov 2-rozmernými jednotkovými kockami; t.j. možnosti nahradenia dvojice susedných jednotkových hrán dvojrozmernou jednotkovou kockou. To znamená, že v tabuľke 3.15 budeme opäť hľadať dvojice vektorov s Hammingovou vzdialenosťou 1, nahrádzať zložku, v ktorej sa odlišujú symbolom don't care a vyškrtávať z implikantov literál v ktorom sa odlišujú. Kedže vektory v tabuľke 3.15 obsahujú symboly don't care, kombinovať možno len tie vektory, ktoré majú symbol don't care na tom istom mieste, pozri tabuľku 3.16

Výsledky 2. kola kombinovania vektorov uvádzame v tabuľke 3.20. V predchádzajúcej tabuľke 3.15 označíme tie vektory, ktoré boli použité v 2. kole-ide o vektory, ktoré boli pokryté dvojrozmernými kockami. Keďže dvojrozmerná kocka sa skladá z dvoch dvojíc

x_1	x_2	x_3	x_4	implikant	kombinácia	kontrola
0	0	0	–	$\overline{x}_1\overline{x}_2\overline{x}_3$	0, 1	✓
0	0	–	0	$\overline{x}_1\overline{x}_2\overline{x}_4$	0, 2	✓
0	–	0	0	$\overline{x}_1\overline{x}_3\overline{x}_4$	0, 4	✓
–	0	0	0	$\overline{x}_2\overline{x}_3\overline{x}_4$	0, 8	✓
0	0	–	1	$\overline{x}_1\overline{x}_2x_4$	1, 3	✓
0	0	1	–	$\overline{x}_1\overline{x}_2x_3$	2, 3	✓
0	–	1	0	$\overline{x}_1x_3\overline{x}_4$	2, 6	✓
0	1	–	0	$\overline{x}_1x_2\overline{x}_4$	4, 6	✓
–	0	0	1	$\overline{x}_2\overline{x}_3x_4$	1, 9	✓
1	0	0	–	$x_1\overline{x}_2\overline{x}_3$	8, 9	✓
–	1	0	0	$x_2\overline{x}_3\overline{x}_4$	4, 12	✓
1	–	0	0	$x_1\overline{x}_3\overline{x}_4$	8, 12	✓

Tabuľka 3.15: Quine-McCluskey, 1.kolo

x_1	x_2	x_3	x_4	implikant	kombinácia
0	0	0	–	$\overline{x}_1\overline{x}_2\overline{x}_3$	0, 1
0	0	1	–	$\overline{x}_1\overline{x}_2x_3$	2, 3
0	0	–	–	$\overline{x}_1\overline{x}_2$	0, 1, 2, 3

Tabuľka 3.16: kombinácie vektorov obsahujúcich don't care

hrán, možno ju vytvoriť dvoma rozličnými spôsobmi. Preto v každom kroku budeme kontrolovať, či sme nevytvorili implikant, ktorý sa už v našom zozname nachádza; ak áno, nebudeme ho zapisovať ešte raz, ale označíme ako použitú (pokrytú) aj druhú dvojicu vektorov, ktorej kombináciou daný implikant vznikol.

Vektory z tabuľky 3.20 už nemáme s čím kombinovať. (Inak povedané, v grafe Booleovskej funkcie sa nevyskytujú jednotkové podkocky dimenzie 3 a vyššej.) V neoznačených riadkoch predchádzajúcich troch tabuliek 3.14, 3.15 a 3.20 sa nachádzajú všetky prosté implikanty Booleovskej funkcie f. Skrátená DNF Booleovskej funkcie f vyzerá nasledovne:

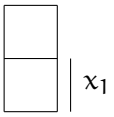
$$\overline{x}_1\overline{x}_2 \vee \overline{x}_2\overline{x}_3 \vee \overline{x}_1\overline{x}_4 \vee \overline{x}_3\overline{x}_4 \vee x_1x_2x_3x_4.$$

x_1	x_2	x_3	x_4	implikant	kombinácia
0	0	–	–	$\overline{x}_1\overline{x}_2$	0, 1, 2, 3
–	0	0	–	$\overline{x}_2\overline{x}_3$	0, 1, 8, 9
0	–	–	0	$\overline{x}_1\overline{x}_4$	0, 2, 4, 6
–	–	0	0	$\overline{x}_3\overline{x}_4$	0, 4, 8, 12

Tabuľka 3.17: Quine-McCluskey, 2. kolo

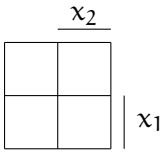
3.4.3 Karnaughove mapy

Pre Booleovské funkcie menšieho počtu premenných (≤ 4) možno na konštrukciu skrátenej DNF a často aj na konštrukciu minimálnej DNF použiť metódu založenú na inom geometrickom modeli Booleovskej funkcie, na tzv. Karnaughových mapách. Karnaughova mapa je tabuľka, do ktorej sa zapisujú hodnoty Booleovskej funkcie. Namiesto toho, aby sme ich definovali formálne, ukážeme, ako sa Karnaughove mapy konštruujú. Tieto mapy má zmysel konštruovať aspoň pre funkcie jednej premennej. Booleovská funkcia jednej premennej má pravdivostnú tabuľku s dvomi riadkami, to znamená, že Karnaughova mapa musí mať dve políčka. Aby sme nemuseli popisovať jednotlivé políčka tabuľky, označíme to políčko, pre ktoré nadobúda jediná premenná Booleovskej funkcie hodnotu 1. Je zrejmé, že na druhom políčku nadobúda hodnotu 0.



Obr. 3.6: (Prázdna) Karnaughova mapa funkcie jednej premennej

Karnaughovu mapu pre funkciu dvoch premenných ($f(x_1, x_2)$) dostaneme tak, že spojíme dve Karnaughove mapy pre funkcie jednej premennej—$f(x_1, 1)$ a $f(x_1, 0)$, obrázok 3.7.



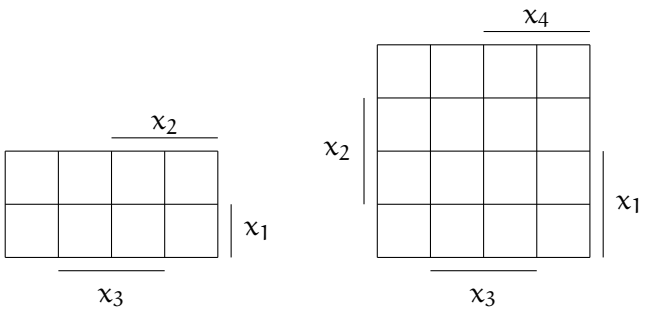
Obr. 3.7: (Prázdna) Karnaughova mapa funkcie dvoch premenných

Karnaughovu mapu pre funkciu troch premenných $f(x_1, x_2, x_3)$ dostaneme spojením dvoch Karnaughových máp pre funkcie dvoch premenných $f(x_1, x_2, 0)$ a $f(x_1, x_2, 1)$, podobne ako Karnaughovu mapu pre funkciu štyroch premenných dostaneme spojením dvoch Karnaughových máp pre funkcie troch premenných, obrázok 3.8.

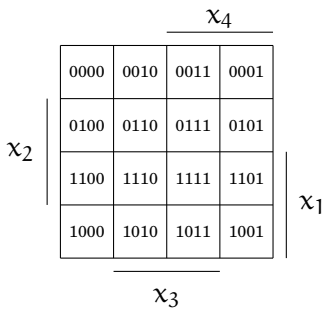
Dajú sa zostrojiť aj Karnaughove mapy pre Booleovské funkcie piatich a väčšieho počtu premenných, ale v týchto mapách už oblasti, v ktorých sú jednotlivé premenné jednotkové, nie sú súvislé. V ďalšom sa budeme podrobnejšie zaberať Karnaughovou mapou pre funkciu 4 premenných. Najprv explicitne uvedieme akým vektorom hodnôt zodpovedajú jednotlivé políčka Karnaughovej mapy, Obr. 3.9.

Zapíšme teraz do Karnaughovej mapy hodnoty Booleovskej funkcie. Podobne ako v prípade grafu Booleovskej funkcie, kde sme sa zaoberali len jednotkovými vrcholmi a ich pokrytím, budeme aj do Karnaughovej mapy zapisovať len jednotkové hodnoty Booleovskej funkcie. Na obrázku 3.10 je Karnaughova mapa funkcie 4 premenných, ktorú sme použili na ilustráciu Quine-McCluskeyovej metódy.

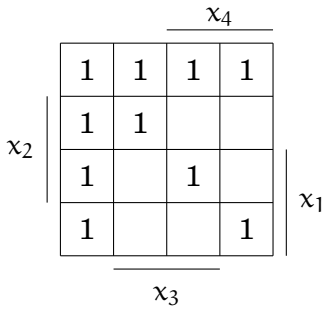
Priamo na základe Karnaughovej mapy možno zostrojiť ÚDNF Booleovskej funkcie.



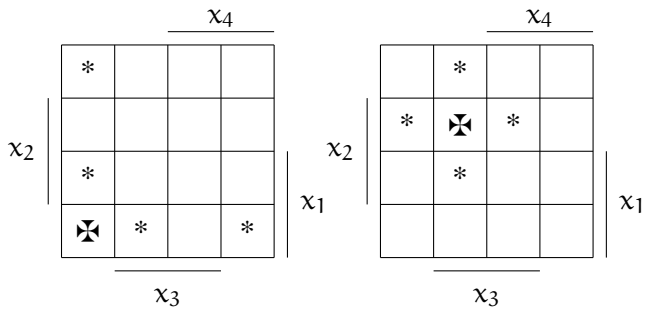
Obr. 3.8: (Prázdne) Karnaughove mapy funkcie troch a štyroch premenných



Obr. 3.9: „Adresy“ políčok v Karnaughovej mape štyroch premenných



Obr. 3.10: Karnaughova mapa Booleovskej funkcie $f(x_1, x_2, x_3, x_4)$



Obr. 3.11: Susednosť v Karnaughovej mape

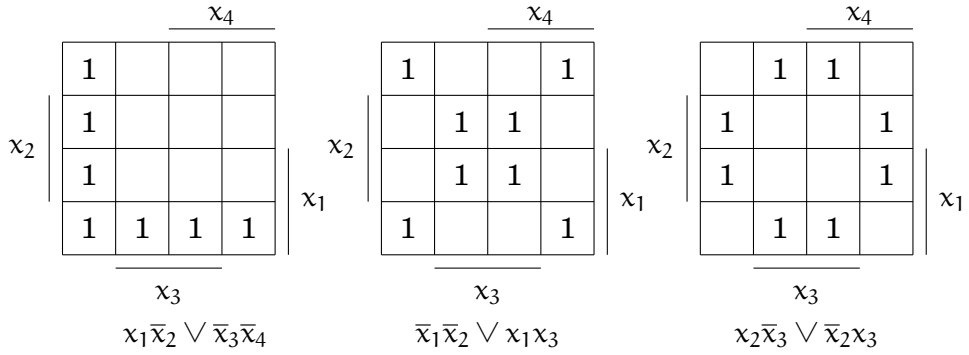
Každému jednotkovému políčku priradíme elementárnu konjunkciu na základe jeho „adresy"—napríklad jednotkové políčko s adresou 0111 leží v prieniku oblastí v ktorých premenné x_2, x_3, x_4 nadobúdajú hodnoty 1 a premenná x_1 hodnotu 0; jednotkovému vektoru 0111 teda priradíme elementárnu konjunkciu $\overline{x}_1 x_2 x_3 x_4$.[14] Na konštrukciu ÚDNF by sme nepotrebovali konštruovať Karnaughovu mapu, ale vystačili by sme s pravdivostnou tabuľkou Booleovskej funkcie. Karnaughova mapa nám umožňuje viac—nájsť prosté implikanty Booleovskej funkcie. Všimneme si vektory-adresy políčok Karnaughovej mapy na obrázku 3.9; každé políčko susedí so štyrmi ďalšími-políčkom nad, pod, napravo a naľavo, pričom susednosť presahuje cez okraj Karnaughovej mapy, prozri obrázok 3.11

Susedné jednotkové políčka možno spojiť do väčších jednotkových oblastí s 2, 4, 8, 16 políčkami. Jednotkovému políčku zodpovedá elementárna konjunkcia (v tomto prípade) rangu 4. Dve susedné jednotkové políčka tvoria oblasť, ktorej prislúcha elementárna konjunkcia rangu 3. Na obrázku 3.12 sú znázornené rozličné jednotkové oblasti so 4 políčkami a im prislúchajúce implikanty. Implikant (elementárnu konjunkciu) zodpovedajúcu jednotkovej oblasti Karnaughovej mapy určíme tak, že do konjunkcie zaradíme literály určujúce oblasti, do prieniku ktorých daná jednotková oblasť patrí. Ak sa v jednotkovej oblasti nachádzajú políčka z oblasti x_i aj $\overline{x}_i$, literál premennej x_i sa v konjunkcii nebude vyskytovať; ak daná jednotková oblasť leží celá v oblasti x_i, jej konjunkcia bude obsahovať premennú x_i a napokon, ak jednotková oblasť leží mimo oblasti x_i jej konjunkcia bude obsahovať negáciu tejto premennej— $\overline{x}_i$.

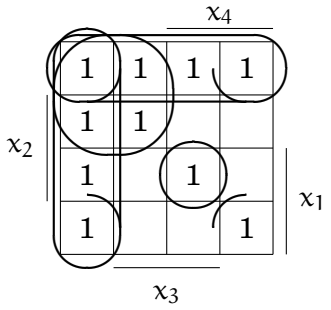
Vrátime sa ku Karnaughovej mape Booleovskej funkcie $f(x_1, x_2, x_3, x_4)$. V Karnaughovej mape vyznačíme jednotkové oblasti a im prislúchajúce prosté implikanty (obrázok 3.13):

1. 1. stĺpec zľava: $\overline{x}_3\overline{x}_4$
2. 1. riadok zhora: $\overline{x}_1\overline{x}_2$
3. ľavý horný kvadrant: $\overline{x}_1\overline{x}_4$

[14] Inak povedané, jednotkovému vektoru $\sigma_1, \sigma_2\sigma_3, \sigma_4$ priradíme (ako sa dalo očakávať) elementárnu konjunkciu $x_1^{\sigma_1} x_2^{\sigma_2} x_3^{\sigma_3} x_4^{\sigma_4}$. Pointa je v tom, že v Karnaughovej mape sú vyznačené len oblasti, v ktorých nadobúdajú jednotlivú premenné jednotkové hodnoty a nie vektory hodnôt premenných.



Obr. 3.12: Jednotkové oblasti v Karnaughovej mape



Obr. 3.13: Prosté implikanty Booleovskej funkcie $f(x_1, x_2, x_3, x_4)$

4. všetky 4 rohy: $\overline{x}_2\overline{x}_3$
5. izolovaná jednotka v pravom dolnom kvadrante: $x_1x_2x_3x_4$

Skrátená a zároveň minimálna DNF Booleovskej funkcie $f(x_1, x_2, x_3, x_4)$ je

$$\overline{x}_1\overline{x}_2 \vee \overline{x}_2\overline{x}_3 \vee \overline{x}_1\overline{x}_4 \vee \overline{x}_3\overline{x}_4 \vee x_1x_2x_3x_4.$$

3.4.4 Výber pokrytia

Predpokladajme, že sme úspešne zostrojili skrátenú DNF; t.j. našli všetky prosté implikanty Booleovskej funkcie. Ďalším krokom je odstránenie nadbytočných prostých implikantov a konštrukcia iredudantnej DNF. Ako sa ukáže v nasledujúcej časti, Booleovská funkcia môže mať veľký počet prostých implikantov a vybrať z nich tie, ktoré tvoria minimálnu DNF je vo všeobecnosti veľmi náročná úloha. Pre malý počet[15] premenných sa pokrytie dá zostrojiť pomocou tabuľky pokrytia a niekoľkých relatívne jednoduchých pravidiel. Tabuľka pokrytia Booleovskej funkcie f vyzerá nasledovne: každému jednotkovému vektoru funkcie f je priradený stĺpec (kvôli zjednodušeniu označenia sa namiesto $\sigma(n, i)$ označuje stĺpec len symbolom m_i a nulové vektory do tabuľky pokrytia nezapisujeme). Riadkom matice pokrytia sú priradené prosté implikanty Booleovskej funkcie f. Ak implikant K_j pokrýva vrchol/vektor m_i, na priesečníku riadka K_j a stĺpca m_i je v tabuľke pokrytia jednotka. V opačnom prípade ponecháme kvôli prehľadnosti políčko tabuľky prázdne. Na ilustráciu uvádzame tabuľku pokrytia funkcie $f(x_1, x_2, x_3, x_4)$ z predchádzajúceho príkladu, tabuľka 3.18.

	m_0	m_1	m_2	m_3	m_4	m_6	m_8	m_9	m_{12}	m_{15}	cena
$\overline{x}_1\overline{x}_2$	1	1	1	1							2
$\overline{x}_2\overline{x}_3$	1	1					1	1			2
$\overline{x}_1\overline{x}_4$	1		1		1	1					2
$\overline{x}_3\overline{x}_4$	1				1		1		1		2
$x_1x_2x_3x_4$										1	4

Tabuľka 3.18: Tabuľka pokrytia Booleovskej funkcie $f(x_1, x_2, x_3, x_4)$

Každému prostému implikantu sme priradili cenu, v tomto prípade vyjadrenú počtom jeho literálov. Cena môže byť stanovená aj ináč a zohráva dôležitú úlohu pri rozhodovaní, ktorý z prostrých implikantov zaradíme do pokrytia a ktorý vylúčime. Je zrejmé, že na to, aby výsledná DNF realizovala danú Booleovskú funkciu, je potrebné vybrať do DNF (resp. jej zodpovedajúceho pokrytia) prosté implikanty tak, že ak ponecháme v tabuľke len riadky zodpovedajúce vybraným prostým implikantom, každom stĺpci tabuľky pokrytia bude aspoň jedna jednotka. Pri výbere prostých implikantov budeme využívať nasledujúce pravidlá:

[15] pojem malý je relatívny. Pred niekoľkými rokmi sa študenti naprogramovali konštrukciu minimálnej DNF. Program sám o sebe nebol zložitý, ale keď ho testovali na vtedy modernej 286-ke, problémy sa prejavili už pri funkciách 8 premenných. Efektívnejšie kódovanie implikantov umožnilo zvýšiť počet premenných o 1, ale podstatné vylepšenie neprinieslo.

Pravidlo podstatného prostého implikantu. Prvé pravidlo je jednoduché a logické: **ak je v niektorom sĺpci tabuľky pokrytia jediná jednotka, príslušný prostý implikant musí byť zaradený do DNF** (a to bez ohľadu na jeho cenu) do DNF, pretože bez neho sa DNF realizujúca danú Booleovskú funkciu zostrojiť nedá. Takýto prostý implikant sa nazýva *podstatný*. Z tabuľky 3.14 vidíme, že sú všetky implikanty podstatné a teda skrátená DNF je zároveň aj minimálnou DNF danej Booleovskej funkcie. Vo všeobecnom prípade to také jednoduché nebude. Nájdenie podstatného implikantu (napríklad K_j) nám však umožní zjednodušiť tabuľku pokrytia—môžeme z nej odstrániť všetky stĺpce, ktoré majú v riadku zodpovedajúcemu K_j jednotky. Vektory/vrcholy prislúchajúce týmto stĺpcom sú už pokryté vybraným prostým implikantom

Pravidlo dominujúceho riadka. Pre pravidlo dominujúceho riadka je podstatné rozmiestnenie jednotkových prvkov v riadku a cena riadka. Nech má riadok r_j jednotkové hodnoty v stĺpcoch $i_1, \ldots, i_s$; cenu c_j a riadok r_k jednotkové hodnoty v stĺpcoch $l_1, \ldots, l_t$ a cenu c_k . Budeme hovoriť, že riadok r_j dominuje nad riadkom r_k práve vtedy, ak $\{l_1, \ldots, l_t\} \subseteq \{i_1, \ldots, i_s\}$ a $c_j \leq c_k$; t.j. riadok r_k pokrýva len nejakú podmnožinu tých vektorov, ktoré pokrýva riadok r_j a má vyššiu alebo rovnakú cenu ako riadok r_j. Pravidlo dominujúceho riadka potom znie **ak v tabuľke pokrytia riadok r_j dominuje nad riadkom r_k, možno z nej riadok r_k vynechať.**

	m_0	m_1	m_2	m_3	m_4	m_6	m_{12}	m_{15}	cena
$\overline{x}_1\overline{x}_2$	1	1	1	1					2
$\overline{x}_2\overline{x}_3$	1	1							2

Tabuľka 3.19: Dominujúci riadok

Na začiatku konštrukcie DNF obsahuje tabuľka pokrytia len prosté implikanty a pravidlo dominujúceho riadka nemožno uplatniť. Po uplatnení iných pravidiel z tabuľky vypadnú niektoré riadky a stĺpce a môžu sa objaviť aj dominujúce riadky. Ilustrujeme to na príklade, ktorý sme dostali modifikáciou[16] tabuľky 3.14. Predpokladajme, že nejakým spôsobom už boli pokryté vektory m_8, m_9. Potom riadok $\overline{x}_1\overline{x}_2$ dominuje nad riadkom $\overline{x}_2\overline{x}_3$ a teda riadok $\overline{x}_2\overline{x}_3$ možno z tabuľky pokrytia vylúčiť.

Pravidlo dominujúceho stĺpca Nech stĺpce m_r, m_s tabuľky pokrytia, majú jednotkové hodnoty v riadkoch $i_1, \ldots, i_k$, resp. $j_1, \ldots, j_l$ Stĺpec m_r dominuje nad stĺpcom m_s práve vtedy, ak $\{i_1, \ldots, i_k\} \subseteq \{j_1, \ldots, j_l$; t.j. každý implikant, ktorý pokrýva m_r, pokrýva aj m_s. Pravidlo dominujúceho stĺpca: **ak stĺpec m_r dominuje nad stĺpcom m_s, tak stĺpec m_s možno z tabuľky pokrytia vyradiť.** Napríklad v tabuľke 3.14 dominuje stĺpec m_3 nad stĺpcami m_0, m_1, m_2, m_6 nad m_4 a pod.

Pri konštrukcii pokrytia (minimalizácii DNF) sa pokúšame opakovane používať jednotlivé pravidlá. Niekedy sa stáva, že sa už žiadne z uvedených pravidielnedá použiť ale konštrukcia DNF nie je ukončená (ostali nepokryté stĺpce a niekoľko prostých implikantov, z ktorých je potrebné vyberať.). Vtedy je potrebné rozumným spôsobom pre-

[16] pripomíname, že teraz už nejde o minimalizáciu pôvodnej Booleovskej funkcie

brať všetky možnosti: vyberieme stĺpec, ktorý obsahuje najmenej jednotiek (napríklad 2), vyberieme prostý implikant zodpovedajúci prvej jednotke, zaradíme ho do DNF a pokračujeme v konštrukcii DNF. Potom vyberieme prostý implikant zodpovedajúci druhej jednotke, zostrojíme DNF, napokon porovnáme obe DNF a vyberieme z nich tú, ktorá má nižšiu cenu.

Pri návrhu logických obvodov bývajú transformácie, ktoré je potrebné realizovať, často popísané pomocou *Booleovských operátorov.* (Pripomíname, že Booleovský operátor je zobrazenie $F^{n,m} : \{0,1\}^n \to \{0,1\}^m$, ktoré sa dá interpretovať ako m-tica n-árnych Booleovských funkcií.) Booleovský operátor by sme mohli realizovať pomocou DNF jeho jednotlivých Booleovských funkcií. Takáto realizácia však nevyužíva to, že čiastkové funkcie Booleovského operátora nemusia byť nezávislé (môžu sa medzi nimi dokonca vyskytovať rovnaké funkcie) a pre každú z nich konštruuuje samostatnú DNF. Quineho-McCluskeyova metóda konštrukcie prostých implikantov a uvedená metóda konštrukcie DNF sa dá upraviť aj na konštrukciu disjunktívnych normálnych foriem realizujúcich Booleovské operátory. Modifikácia Quineho-McCluskeyovej metódy spočíva vo vyhľadávaní všetkých spoločných (skupinových) prostých implikantov pre všetky možné kombinácie čiastkových Booleovských funkcií Boolovského operátora. To potom pri konštrukcii pokrytia (DNF) umožňuje rozhodovať o tom, ktoré jednotkové vrcholy jednotlivých čiasktových Booleovských funkci sa budú pokrývať individuálne a u ktorých sa pokrytie bude zdieľať. Konštrukcia skupinových prostých implikantov nie je principiálne zložitá, problém spočíva v tom, že netriviálnych kombinácii m Booleovských funkcií je $2^m - 2$ a rozličných skupinových prostých implikantov môže byť už pre malé hodnoty n, m netriviálne veľa.

3.4.5 *Odhady parametrov DNF

V predchádzajúcich častiach tejto kapitoly sme neraz narazili na problém, že principiálne jednoduchá konštrukcia nemusí byť prakticky použiteľná. Jednoduchým príkladom je ÚDNF. n-árna Booleovská funkcia je zadaná vektorom svojich pravdivostných hodnôt, ktorý má dĺžku 2^n. Typická Booleovská funkcia f má približne polovicu jednotiek a polovicu núl; presnejší odhad dostaneme pomocou Čebyševovej nerovnosti:

$$2^{n-1} - \phi(n) \cdot 2^{n/2} < |N_f| < 2^{n-1} + \phi(n) \cdot 2^{n/2},$$

kde $\phi(n) \to \infty$ pre $n \to \infty$ je ľubovoľná pomaly rastúca funkcia. To znamená, že dĺžka ÚDNF typickej n-árnej Booleovskej funkcie bude približne 2^{n-1}. Problémy spojené s minimalizáciou DNF viedli k skúmaniu dlžok a počtov rozličných DNF Booleovských funkcií. Dosiahnuté výsledky umožnili vytvoriť si ucelenejšiu predstavu o zložitosti problémov, na ktorý pri minimalizácii DNF narážame a posúdeniu efektívnosti metód minimalizácie. Výsledky sú prebraté z [**?**] a uvádzame ich bez dôkazov. Budeme používať nasledujúce označenie: ak λ označuje nejaký parameter Booleovských funkcií, tak $\lambda(f)$ označuje hodnotu tohto parametra na funkcii f a $\lambda(n) = \max_f \lambda(f)$, pričom maximum sa berie cez všetky n-árne Booleovské funkcie.

Dĺžka skrátenej DNF. Nech $l_S(f)$ označuje dĺžku skrátenej DNF Booleovskej funkcie f. Potom[17]

$$\frac{3^n}{n} \preccurlyeq l_S(n) \preccurlyeq \frac{3^n}{\sqrt{n}}.$$

Pritom pre skoro všetky n-árne Booleovské funkcie platí

$$n^{(1-\delta'_n)\lg\lg n} \cdot 2^n \leq l_S(f) \leq n^{(1+\delta''_n)\lg\lg n} \cdot 2^n,$$

kde $\delta'_n, \delta''_n \to 0$ pre $n \to \infty$.

Počet iredudantných DNF. Nech $t(f)$ označuje počet iredudantných DNF Booleovskej funkcie f. Najprv uvedieme odhad maximálnej hodnoty:

$$\left(2^{2^n}\right)^{c'_n\sqrt{n}} \leq t(n) \leq \left(2^{2^n}\right)^{c''_n n},$$

kde veličiny c'_n (pre $n \geq 3$) sú zdola a c''_n zhora ohraničené kladnými konštantami. Pre skoro všetky n-árne Booleovské funkcie je počet iredudantných DNF ohraničený nasledovne:

$$\left(2^{2^{n-1}}\right)^{(1-\varepsilon'_n)\lg n\lg\lg n} \leq t(f) \leq \left(2^{2^{n-1}}\right)^{(1+\varepsilon''_n)\lg n\lg\lg n},$$

kde $\varepsilon'_n, \varepsilon''_n \to 0$ pre $n \to \infty$. To znamená, že prehľadávanie množiny iredudantných DNF na to, aby sme našli minimálnu DNF je mimoriadne náročné.

Počet najkratších DNF. Najkratšia DNF má minimálnu dĺžku, t.j. počet konjunkcií. Dá sa očakávať, že pre jednu Booleovskú funkciu bude existovať viacero najkratších DNF. Označme symbolom $q(f)$ počet najkratších DNF Booleovskej funkcie f. Je známy len dolný odhad maximálnej hodnoty parametra $q(n)$:

$$\left(2^{2^n}\right)^{c'_n\sqrt{n}} \leq t(n).$$

Dĺžka iredudantnej DNF. Nech $l_T(f)$ je maximálna dĺžka iredudantnej DNF a $l_K(f)$ dĺžka najkratšej DNF. Potom

$$l_K(n) \sim 2^n,$$

a pre skoro všetky Booleovské funkcie

$$l_K(f) \sim 2^{n-1}$$

.

Dĺžka najkratšej DNF. Označme symbolom $l_K(f)$ dĺžka najkratšej DNF. Potom

$$l_K(n) = 2^{n-1},$$

a pre skoro všetky n-árne Booleovské funkcie

$$\frac{2^{n-1}}{\lg n \lg\lg n} \lesssim l_K(f) < \frac{2^n}{\lg n}.$$

[17] Symbol $\preccurlyeq$ vyjadruje vzťah menší alebo rádovo rovný.

Relatívna dĺžka iredudantnej DNF. Pre danú Booleovskú funkciu existuje—ako sme videli—veľa iredudandntých DNF. Bolo by zaujímavé vedieť, do akej miery sa ich dĺžky odlišujú od optima, ktoré predstavuje dĺžka najkratšej DNF. Relatívnou dĺžkou iredudantnej DNF sa nazýva pomer jej dĺžky k dĺžke najkratšej DNF. Pre funkciu f zavádzame parameter $\Upsilon(f)$, nazývaný rozptylom, ktorý je definovaný ako maximálna relatívna dĺžka iredudantnej DNF Booleovskej funkcie f. Pre maximálnu hodnotu rozptylu dĺžok platí

$$\Upsilon(n) = 2^{n(1-\varepsilon)},$$

kde $\varepsilon \to 0$, pre $n \to \infty$. Pre typickú Booleovskú funkciu je rozptyl dĺžok DNF podstatne menší, aj keď rastie vzhľadom na veľkosť n:

$$\lg n \preceq F(f) \precsim \lg n \lg\lg n.$$

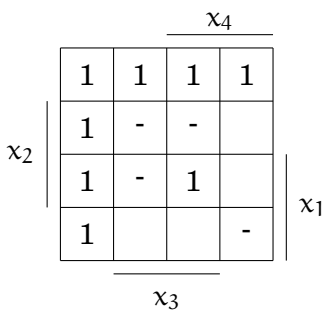
Keďže iredudantných DNF je príliš veľa na úplné preberanie, alternatívou úplného preberania by mohol byť náhodný výber a následné úplné preberanie podmnožiny iredudantných DNF. Posledný odhad hovorí, že takýmto spôsobom by sme mohli dostať iredudantnú DNF, ktorej dĺžka by bola minimálne $\lg n$-krát dlhšia, ako je dĺžka najkratšej DNF.

3.4.6 Neúplne určené Booleovské funkcie

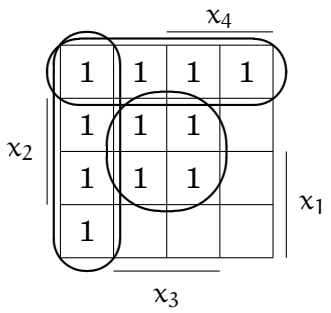
V niektorých úlohách sa stretávame s Booleovskými funkciami, ktoré nie sú definované pre všetky hodnoty svojich vstupných premenných (pozri napríklad tabuľku 3.4). Dalo by sa očakávať, že táto neurčitosť bude pri realizácii Booleovských funkcií spôsobovať problémy. Prekvapujúce je, že nie a dokonca realizácia neúplne určených Booleovských funkcií môže byť jednoduchšia, ako v prípade plne definovaných funkcií. Čo vlastne—z hľadiska realizácie—znamená, že Booleovská funkcia je na nejakom vektore svojich hodnôt neurčená? Môže sa to chápať dvojako: buď je z nejakých dôvodov jedno, akú hodnotu Booleovská funkcia na danom vektore nadobúda; alebo je daná vstupná hodnota zakázaná a tak je v konečnom dôsledku tiež jedno, akú hodnotu bude Booleovská funkcia na zakázanom vstupe nadobúdať. Ošetrenie vstupov formuly alebo obvodu realizujúceho Booleovskú funkciu nie je našou úlohou. Budeme predpokladať, že tento problém je vyriešený a pre nás neurčená hodnota znamená, že ju môžeme definovať ako sa nám hodí. Rozumné je ponechať si možnosť dodefinovať funkciu otvorenú čo najdlhšie. To sa dá spraviť tak, že sa neurčeným hodnotám priradí symbol - (don't care). Pri hľadaní prostých implikantov s "don't care" narábame tak, ako keby predstavoval hodnotu 1, s výnimkou prípadov, keď by sme mali zostrojiť prostý implikant, ktorý by pokrýval samé "don't care" -vektory. To však ošetríme pri konštrukcii pokrytia; v záhlaví tabuľky pokrytia uvedieme len jednotkové vrcholy, a to znamená, že prosté implikanty, ktoré by pokrývali len "don't care" vrcholy sa do tabuľky nedostanú, resp. vypadnú automaticky na základe pravidla o dominujúcom riadku. Metódu ilustrujeme na príklade. Na obrázku 3.14 je uvedená Karnaughova mapa neúplne určenej Booleovskej funkcie 4 premenných.

Na základe Karnaughovej mapy zostrojíme priamo DNF Booleovskej funkcie:

$$\overline{x}_1\overline{x}_2 \vee \overline{x}_3\overline{x}_4 \vee x_2x_3.$$



Obr. 3.14: Karnaughova mapa neúplne určenej Booleovskej funkcie



Obr. 3.15: Doplnenie neúplne určenej Booleovskej funkcie

Tri "don't care" -vektory sme definovali ako jednotkové, jeden ako nulový—obrázok 3.15

Zostrojíme teraz pomocou Qiune-McCluskeyovej metódy zoznam všetkých prostých implikantov neúplne určenej Booleovskej funkcie a potom zostrojíme pokrytie jednotkových vektorov, resp. jemu zodpovedajúcu minimálnu DNF danej Booleovskej funkcie.

Teraz zostrojíme tabuľku pokrytia. Vektor m_{15} je pokrytý jedine prostým implikantom x_2x_3; t.j. implikant x_2x_3 je podstatný. 1. riadok $(\overline{x}_1\overline{x}_2)$ dominuje nad 5. riadkom $(\overline{x}_1x_3)$, 4. riadok $(\overline{x}_3\overline{x}_4)$ nad 6. riadkom $(x_2\overline{x}_4)$. Po odstránení 7. riadka a stĺpca m_{15}, 5. a 6. riadku dostávame nasledujúcu tabuľku pokrytia: Z tabuľky pokrytia vyplýva, že podstatné implikanty sú $\overline{x}_1\overline{x}_2$, ktorý pokrýva vektor m_3 a $\overline{x}_3\overline{x}_4$, ktorý pokrýva vektor m_{12}. Tieto dva implikanty však pokrývajú všetky ostávajúce jednotkové vektory, a teda minimálna DNF pre neúplne určenú Booleovskú funkciu f bude:

$$\overline{x}_1\overline{x}_2 \vee \overline{x}_3\overline{x}_4 \vee x_2x_3.$$

Pripomenieme, že túto istú DNF sme dostali podstatne jednoduchším spôsobom pomocou karnaughovej mapy.

x_1	x_2	x_3	x_4	implikant	číslo	kontrola
0	0	0	0	$\overline{x}_1\overline{x}_2\overline{x}_3\overline{x}_4$	0	✓
0	0	0	1	$\overline{x}_1\overline{x}_2\overline{x}_3x_4$	1	✓
0	0	1	0	$\overline{x}_1\overline{x}_2x_3\overline{x}_4$	2	✓
0	1	0	0	$\overline{x}_1x_2\overline{x}_3\overline{x}_4$	4	✓
1	0	0	0	$x_1\overline{x}_2\overline{x}_3\overline{x}_4$	8	✓
0	0	1	1	$\overline{x}_1\overline{x}_2x_3x_4$	3	✓
0	1	1	0	$\overline{x}_1x_2x_3\overline{x}_4$	6	✓
1	0	0	1	$x_1\overline{x}_2\overline{x}_3x_4$	9	✓
1	1	0	0	$x_1x_2\overline{x}_3\overline{x}_4$	12	✓
0	1	1	1	$\overline{x}_1x_2x_3x_4$	7	✓
1	1	1	0	$x_1x_2x_3\overline{x}_4$	14	✓
1	1	1	1	$x_1x_2x_3x_4$	15	✓
0	0	0	–	$\overline{x}_1\overline{x}_2\overline{x}_3$	0, 1	✓
0	0	–	0	$\overline{x}_1\overline{x}_2\overline{x}_4$	0, 2	✓
0	–	0	0	$\overline{x}_1\overline{x}_3\overline{x}_4$	0, 4	✓
–	0	0	0	$\overline{x}_2\overline{x}_3\overline{x}_4$	0, 8	✓
0	0	–	1	$\overline{x}_1\overline{x}_2x_4$	1, 3	✓
0	0	1	–	$\overline{x}_1\overline{x}_2x_3$	2, 3	✓
0	–	1	0	$\overline{x}_1x_3\overline{x}_4$	2, 6	✓
0	1	–	0	$\overline{x}_1x_2\overline{x}_4$	4, 6	✓
–	0	0	1	$\overline{x}_2\overline{x}_3x_4$	1, 9	✓
1	0	0	–	$x_1\overline{x}_2\overline{x}_3$	8, 9	✓
–	1	0	0	$x_2\overline{x}_3\overline{x}_4$	4, 12	✓
1	–	0	0	$x_1\overline{x}_3\overline{x}_4$	8, 12	✓
0	–	1	1	$\overline{x}_1x_3x_4$	3, 7	✓
0	1	1	–	$\overline{x}_1x_2x_3$	6, 7	✓
–	1	1	0	$x_2x_3\overline{x}_4$	6, 14	✓
1	1	–	0	$x_1x_2\overline{x}_4$	12, 14	✓
–	1	1	1	$x_2x_3x_4$	7, 15	✓
1	1	1	–	$x_1x_2x_3$	14, 15	✓
0	0	–	–	$\overline{x}_1\overline{x}_2$	0, 1, 2, 3	
–	0	0	–	$\overline{x}_2\overline{x}_3$	0, 1, 8, 9	
0	–	–	0	$\overline{x}_1\overline{x}_4$	0, 2, 4, 6	
–	–	0	0	$\overline{x}_3\overline{x}_4$	0, 4, 8, 12	
0	–	1	–	$\overline{x}_1x_3$	2, 3, 6, 7	
–	1	–	0	$x_2\overline{x}_4$	4, 6, 12, 14	
–	1	1	–	x_2x_3	6, 7, 14, 15	

Tabuľka 3.20: Prosté implikanty Booleovskej funkcie f

implikant	m_0	m_1	m_2	m_3	m_4	m_8	m_{12}	m_{15}	cena
$\overline{x}_1\overline{x}_2$	1	1	1	1					2
$\overline{x}_2\overline{x}_3$	1	1				1			2
$\overline{x}_1\overline{x}_4$	1		1		1				2
$\overline{x}_3\overline{x}_4$	1				1	1	1		2
$\overline{x}_1x_3$			1	1					2
$x_2\overline{x}_4$					1		1		2
x_2x_3								1	2

Tabuľka 3.21: Tabuľka pokrytia (1)

implikant	m_0	m_1	m_2	m_3	m_4	m_8	m_{12}	cena
$\overline{x}_1\overline{x}_2$	1	1	1	1				2
ï$\overline{x}_2\overline{x}_3$	1	1				1		2
$\overline{x}_1\overline{x}_4$	1		1		1			2
$\overline{x}_3\overline{x}_4$	1				1	1	1	2

Tabuľka 3.22: Tabuľka pokrytia (2)

3.5 Úplnosť a uzavretosť systému Booleovských funkcií

Podľa vety 3.3.2 stačia Booleovské funkcie $\{x_1\&x_2, x_1 \vee x_2, \neg x\}$ na vyjadrenie všetkých ostatných Booleovských funkcií. Existujú aj iné množiny Booleovských funkcií s touto vlastnosťou? Ak je daná množina Booleovských funkcií, vieme určiť, či sa pre ľubovoľnú Booleovskú funkciu dá vytvoriť formula nad touto množinou, ktorá danú Booleovskú funkciu realizuje? Odpovede na tieto otázky budeme hľadať v tejto časti. Začneme upresnením niektorých základných pojmov.

Definícia 3.5.1. *Nech je $\mathcal{M}$ množina Booleovských funkcií. Budeme hovoriť, že $\mathcal{M}$ je úplná ($\mathcal{M}$ tvorí úplný systém Booleovských funkcií), práve vtedy ak ľubovoľnú Booleovskú funkciu možno realizovať pomocou formuly nad $\mathcal{M}$.*

Príklad 3.13. *Nasledujúce množiny Booleovských funkcí tvoria úplné systémy.*

1. *Množina $\mathcal{P}_2$ všetkých Booleovských funkcií,*
2. *Množina $\{x_1\&x_2, x_1 \vee x_2, \neg x\}$,*
3. *Ľubovoľná množina Booleovských funkcií, ktorá je nadmnožinou množiny $\{x_1\&x_2, x_1 \vee x_2, \neg x\}$ alebo inej úplnej množiny Booleovských funkcií.*

Na druhej strane nie všetky množiny Booleovských funkcií sú úplné. Napríklad $\neg x$ nestačí na vyjadrenie konjunkcie; obe konštanty 0 a 1 sú nulárne (nemajú podstatné premenné), a preto z nich nemožno vyjadriť ani funkciu s apoň jednou podstatnou premennou. Nasledujúca veta dáva návod na to, ako zistiť, či nejaká množina Booleovských funkcí tvorí úplný systém.

Veta 3.5.1. *Nech sú dané dve množiny Booleovských funkcií z $\mathcal{P}_2$, $\mathcal{F} = \{f_1, f_2, \ldots, f_s\}$ a $\mathcal{G} = \{g_1, g_2, \ldots, g_t\}$ také že:*

1. *$\mathcal{F}$ tvorí úplný systém Booleovských funkcií a*
2. *každá funkcia z $\mathcal{F}$ sa dá realizovať pomocou formuly nad množinou $\mathcal{G}$.*

Potom množina $\mathcal{G}$ tvorí úplný systém Booleovských funkcií.

Dôkaz. Nech je h ľubovoľná funkcia z $\mathcal{P}_2$. Potom existuje formula $\mathbf{A}[\mathcal{F}]$, ktorá realizuje funkciu h; $\mathbf{A} = f_{i_0}(f_{i_1}, \ldots, f_{i_m})$, $\forall k f_{i_k} \in \mathcal{F}$. Podľa predpokladu vety možno každú funkciu $f_i \in \mathcal{F}$ realizovať pomocou formuly nad $\mathcal{G}$; $f_j = \mathbf{B}_j[\mathcal{G}]$; $\mathbf{B}_j = g_{j_0}(g_{j_1}, \ldots, g_{j_n})$. Ak vo formule $\mathbf{A}$ nahradíme každú funkciu z $\mathcal{F}$, formulou nad $\mathcal{G}$, ktorá ju realizuje, dostávame formulu $\mathbf{A}'[\mathcal{G}] \equiv \mathbf{A}[\mathcal{F}]$. Keďže $\mathcal{F}$ tvorí úplný systém Booleovských funkcií, každú Booleovskú funkciu z $\mathcal{P}_2$ môžeme realizovať pomocou formuly nad $\mathcal{F}$ a túto formulu zasa môžeme vyjadriť pomocou funkcií z $\mathcal{G}$. To znamená, že aj $\mathcal{G}$ tvorí úplný systém Booleovských funkcií. □

Využijeme tvrdenie vety 3.5.1 a zostrojíme niekoľko ďalších úplných systémov Booleovských funkcií.

Príklad 3.14. *1. Množina Booleovských funkcií $\{\neg x, x_1 \& x_2\}$ tvorí úplný systém, pretože $x_1 \vee x_2 \equiv \neg(\neg x_1 \& \neg x_2)$, podobne*

2. množina Booleovských funkcií $\{\neg x, x_1 \vee x_2\}$ tvorí úplný systém, pretože $x_1 \& x_2 \equiv \neg(\neg x_1 \vee \neg x_2)$.

Úloha 3.9. *Zistite, či nasledujúce množiny Booleovských funkcií tvoria úplné systémy:*

1. $\{\neg x_1, x_1 \Rightarrow x_2\}$;
2. $\{1, x_1 \Rightarrow x_2\}$;
3. $\{0, x_1 \Rightarrow x_2\}$;
4. $\{x_1 \& x_2, x_1 \Rightarrow x_2\}$;
5. $\{f_{14}\}$, *Pierceova funkcia;*
6. $\{f_8\}$, *Shefferova funkcia.*

Zaujímavý úplný systém Booleovských funkcií predstavuje nasledujúca množina:

$$\mathcal{D}_3 = \{x_1 \& x_2, x_1 \oplus x_2, 1\}.$$

Podľa predchádzajúcej vety na dôkaz úplnosti stačí ukázať, že pomocou funkcií z $\mathcal{D}_3$ vyjadríme všetky funkcie nejakého úplného systému, napríklad $\{\neg x, x_1 \& x_2\}$. Množina $\mathcal{D}_3$ obsahuje konjunkciu, ale neobsahuje negáciu. Negácia sa však dá vyjadriť takto:

$$\neg x \equiv x \oplus 1. \tag{3.12}$$

To znamená, že množina $\mathcal{D}_3$ tvorí skutočne úplný systém Booleovských funkcií. Konjunkcia a sčítanie modulo 2 sú komutatívne operácie a platí pre ne distributívny zákon (operátor & kvôli zjednodušeniu zápisu vynechávame):

$$x_1(x_2 \oplus x_3) \equiv x_1 x_2 \oplus x_1 x_3. \tag{3.13}$$

Ak využijeme vzťahy 3.12 a 3.13, môžeme ľubovoľnú formulu[18] nad množinou $\mathcal{D}_3$ vyjadriť v *algebraickej normálnej forme* (ANF), nazývanej aj *Žegalkinovým polynómom*:

$$f(x_1, \ldots, x_n) = \bigoplus_{i_1,\ldots,i_s} a_{i_1,\ldots,i_s} x_{i_1} \& \ldots \& x_{i_s}, \tag{3.14}$$

kde znak $\oplus$ označuje súčet modulo 2 a suma sa berie cez všetky možné podmnožiny množiny $\{1, \ldots, n\}$ a koeficienty $a_{i_1,\ldots,i_s}$ nadobúdajú hodnotu z množiny $\{0, 1\}$.

Skôr, ako sa budeme zaoberať ANF Booleovských funkcií podrobnejšie, ilustrujeme na príklade, čo sa skrýva za trocha neprehľadným zápisom 3.14.

Príklad 3.15. *Zapíšeme všeobecný tvar algebraickej normálnej formy Booleovskej funkcie 3 premenných.*

$$\begin{aligned} &f(x_1, x_2, x_3) = \\ &\quad = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus a_3x_3 \oplus a_{1,2}x_1x_2 \oplus a_{1,3}x_1x_3 \oplus a_{2,3}x_2x_3 \oplus a_{1,2,3}x_1x_2x_3. \end{aligned}$$

Všimnite si, že zložitý index $i_1, \ldots, i_s$ konštanty $a_{i_1,\ldots,i_s}$ nevyjadruje nič iné, len indexy premenných, ktoré vystupujú v príslušnej konjunkcii $a_{i_1,\ldots,i_s} x_{i_1} \& \ldots \& x_{i_s}$. Takéto označenie konštánt je dobré pri teoretickom skúmaní vlastností algebraických normálnych foriem. Pri konštrukcii ANF pre konkrétne Booleovské funkcie menšieho počtu premenných sa obvykle používa jednoduchšie označenie, napríklad ANF Booleovskej funkcie 3 premenných sa dá zapísať aj v tvare:

$$f(x, y, z) = a_0 \oplus a_1x \oplus a_2y \oplus a_3z \oplus a_4xy \oplus a_5xz \oplus a_6yz \oplus a_7xyz.$$

Zápis Booleovskej funkcie v ANF je vhodný na skúmanie viacerých vlastností Booleovských funkcií. My ho budeme používať pri zisťovaní linearity Booleovskej funkcie. Z úplnosti systému $\mathcal{D}_3$ vyplýva, že každú Booleovskú funkciu možno zapísať v ANF. Nasledujúca veta tvrdí, že tento zápis je jednoznačný.

Veta 3.5.2. *Pre ľubovoľnú Booleovskú funkciu* $f \in \mathcal{P}_2$ *existuje práve jedna formula v algebraickej normálnej forme, ktorá realizuje Booleovskú funckiu* f.

Dôkaz. V ANF n-árnej Booleovskej funkcie je 2^n binárnych koeficientov. Ak sa dohodneme na pevnom poradí členov v ANF, môžeme ANF Booleovskej funkcie jednoznačne zadať pomocou binárneho vektora dĺžky 2^n. ANF n-árnych Booloevských funkcií je práve toľko, koľko je n-árnych Booloevských funkcií. Z úplnosti systému $\mathcal{D}_3$ vyplýva, že každú Booleovskú funkciu možno zapísať v ANF. Predpokladajme, že tento zápis nie je jednoznačný, t.j. jednej Booleovskej funkcii by boli priradené dve rozličné formuly v ANF. Potom by

- existovala n-árna Booleovská funkcia, pre ktorú neexistuje formula v ANF. To je v spore s úplnosťou systému $\mathcal{D}_3$.
- alebo by jedna formula v ANF realizovala dve rozličné n-árne Booleovské funkcie, čo je v spore s jednoznačnosťou Booleovskej funkcie realizovanej formulou.

□

[18] a teda aj ľubovoľnú Booleovskú funkciu

Poznámka. Dokážeme jednoznačnosť priradenia ANF Boleovskej funkcii iným spôsobom. Nech sú Booleovskej funkcii $f(x_1,\dots,x_n)$ priradené dve rozličné formuly v ANF:

$$\begin{aligned} f(x_1,\dots,x_n) &= a_0 \oplus a_1x_1 \oplus \cdots \oplus a_nx_n \oplus a_{1,2}x_1x_2 \oplus \cdots \oplus a_{1,\dots,n}x_1\dots x_n \\ f(x_1,\dots,x_n) &= b_0 \oplus b_1x_1 \oplus \cdots \oplus b_nx_n \oplus b_{1,2}x_1x_2 \oplus \cdots \oplus b_{1,\dots,n}x_1\dots x_n. \end{aligned}$$

Sčítame modulo 2 pravé i ľavé strany posledných dvoch rovností. Dostávame

$$0 = c_0 \oplus c_1x_1 \oplus \cdots \oplus c_nx_n \oplus c_{1,2}x_1x_2 \oplus \cdots \oplus c_{1,\dots,n}x_1\dots x_n, \tag{3.15}$$

kde

$$c_{i_1,\dots,i_s} = a_{i_1,\dots,i_s} \oplus b_{i_1,\dots,i_s}.$$

Zistíme, aké hodnoty nadobúdajú koeficienty c_i ANF konštantej funkcie 0.[19] Vo vzťahu 3.15 položíme $x_1 = x_2 = \cdots = x_n = 0$. Potom $c_0 = 0$ a

$$0 = c_1x_1 \oplus \cdots \oplus c_nx_n \oplus c_{1,2}x_1x_2 \oplus \cdots \oplus c_{1,\dots,n}x_1\dots x_n, \tag{3.16}$$

Dosadíme $x_1 = 1$ a $x_2 = \cdots = x_n = 0$ do vzťahu 3.16. Dostávame $c_1 = 0$ a

$$0 = c_2x_2 \oplus \cdots \oplus c_nx_n \oplus c_{1,2}x_1x_2 \oplus \cdots \oplus c_{1,\dots,n}x_1\dots x_n,$$

Podobným spôsobom určíme hodnoty koeficientov $c_2 = \dots c_n = 0$ a pre ANF dostávame

$$0 = c_{1,2}x_1x_2 \oplus \cdots \oplus c_{n-1,n}x_{n-1}x_n \oplus c_{1,2,3}x_1x_2x_3 \oplus \cdots \oplus c_{1,\dots,n}x_1\dots x_n.$$

Dosadíme do posledného vzťahu: $x_1 = x_2 = 1; x_3 = \cdots = x_n = 0$ a dostávame $c_{1,2} = 0$. Takto postupne určíme hodnoty všetkých koeficientov:

$$c_0 = c_1 = \cdots = c_n = c_{1,2} = \cdots = c_{1,\dots,n} = 0.$$

To ale znamená, že

$$a_0 = b_0, a_1 = b_1, \dots, a_{1,\dots,n} = b_{1,\dots,n},$$

a obe ANF Booleovskej funkcie $f(x_1,\dots,x_n)$ sa zhodujú.

Príklad 3.16. *Zostrojíme ANF pre Booleovskú funkciu* $f(x_1,x_2,x_3) = 10110100$ *z príkladu . Všeobecný tvar ANF Booleovskej funkcie troch premenných je*

$$\begin{aligned} &f(x_1,x_2,x_3) = \\ &\quad = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus a_3x_3 \oplus a_{1,2}x_1x_2 \oplus a_{1,3}x_1x_3 \oplus a_{2,3}x_2x_3 \oplus a_{1,2,3}x_1x_2x_3. \end{aligned}$$

Položíme $x_1 = x_2 = x_3 = 0$. *Všetky členy ANF, ktoré obsahovali aspoň jednu premennú, sa anulovali a ostal len absolútny člen,* a_0. *Keďže* $f(0,0,0) = 1$, $a_0 = 1$ *a*

$$\begin{aligned} &f(x_1,x_2,x_3) = \\ &\quad = 1 \oplus a_1x_1 \oplus a_2x_2 \oplus a_3x_3 \oplus a_{1,2}x_1x_2 \oplus a_{1,3}x_1x_3 \oplus a_{2,3}x_2x_3 \oplus a_{1,2,3}x_1x_2x_3. \end{aligned}$$

[19] resp. funkcie $f(x_1,\dots,x_n) \oplus f(x_1,\dots,x_n)$

	x_1	x_2	x_3	$f(x_1,x_2,x_3)$		koeficient
0.	0	0	0	1	$a_0 = 1$	$a_0 = 1$
1.	1	0	0	0	$1 \oplus a_1 = 0$	$a_1 = 1$
2.	0	1	0	1	$1 \oplus a_2 = 1$	$a_2 = 0$
3.	0	0	1	0	$1 \oplus a_3 = 0$	$a_3 = 1$
4.	1	1	0	0	$1 \oplus 1 \oplus a_{1,2} = 0$	$a_{1,2} = 0$
5.	1	0	1	1	$1 \oplus 1 \oplus a_{1,3} = 1$	$a_{1,3} = 1$
6.	0	1	1	1	$1 \oplus 1 \oplus 1 \oplus a_{2,3} = 1$	$a_{2,3} = 0$
7.	1	1	1	0	$1 \oplus 1 \oplus 1 \oplus 1 \oplus a_{1,2,3} = 0$	$a_{1,2,3} = 0$

Tabuľka 3.23: Konštrukcia ANF

Teraz určíme koeficient a_1. *Dosadíme do posledného vzťahu* $x_1 = 1, x_2 = x_3 = 0$. *Dostávame rovnosť* $f(1,0,0) = 1 \oplus a_1$, *pretože vypadli všetky členy ANF, ktoré obsahovali premenné* x_2, x_3. *Z pravdivostnej tabuľky funkcie zistíme, že* $f(1,0,0) = 0$, *to znamená, že* $a_1 = 1$. *Odvodenie ďalších koeficientov ANF Booeleovskej funkcie uvádzame kvôli stručnosti a prehľadnosti v tabuľke 3.23. Výsledná ANF Booleovskej funkcie* $f(x_1,x_2,x_3)$ *má tvar:*

$$1 \oplus x_1 \oplus x_3 \oplus x_1x_3.$$

Úloha 3.10. *Zostrojte ANF pre funkcie* $f_0, \ldots, f_{15}$*!*

Úloha 3.11. *Zadajte pomocou tabuľky pravdivostných hodnôt 5 funkcií troch premenných a zostrojte pre ne ANF!*

ANF pre Booleovské funkcie môžeme konštruovať aj tak, že transformujeme niektorú vhodnú formulu realizujúcu danú Booleovskú funkciu na ANF. Napríklad

$$x_1 \vee x_2 \equiv \neg(\neg x_1)\&(\neg x_2)) \equiv [1 \oplus (1 \oplus x_1)(1 \oplus x_2)] \equiv [1 \oplus (1 \oplus x_1 \oplus x_2 \oplus x_1x_2)] \equiv x_1 \oplus x_2 \oplus x_1x_2.$$

Doterajšie poznatky o úplnosti systémov Booleovských funkcií stačia na to, aby sme pre systém, ktorý je úplný vedeli jeho úplnosť dokázať. Aby sme vedeli exaktne zdôvodniť, že nejaký systém Booleovských funkcií nie je úplný a povedať prečo, potrebujeme vytvoriť aparát, ktorý nám umožní efektívne popísať všetky funkcie, ktoré môžeme dostať skladaním funkcií zo skúmaného systému Booleovských funkcií. Zavedieme preto dva dôležité pojmy: *uzáveru množiny* a *uzavretej množiny Booleovských funkcií*.

Definícia 3.5.2. *Nech je* $\mathcal{M}$ *ľubovoľná množina Booleovských funkcií;* $\mathcal{M} \subseteq \mathcal{P}_2$. *Množinu všetkých Booleovských funkcií, ktoré možno realizovať pomocou formuly nad* $\mathcal{M}$ *nazveme uzáverom množiny* $\mathcal{M}$ *a označíme ju symbolom* $[\mathcal{M}]$. *Množinu Booleovských funkcií* $\mathcal{M}$ *budeme nazývať (funkcionálne) uzavretou, ak* $[\mathcal{M}] = \mathcal{M}$.

Základné vlastnosti uzáveru množiny funkcií popisuje nasledujúca veta.

Veta 3.5.3. *Nech sú* $\mathcal{M}, \mathcal{M}_1, \mathcal{M}_2$ *ľubovoľné množiny Booleovských funkcií, potom*

1. $\mathcal{M} \subseteq [\mathcal{M}]$,
2. $[[\mathcal{M}]] = [\mathcal{M}]$,

3. *ak* $\mathcal{M}_1 \subseteq \mathcal{M}_2$, *tak potom* $[\mathcal{M}_1] \subseteq [\mathcal{M}_2]$,
4. $[\mathcal{M}_1] \cup [\mathcal{M}_2] \subseteq [\mathcal{M}_1 \cup \mathcal{M}_2]$.

Dôkaz. Ponechávame čitateľovi ako cvičenie. □

Úloha 3.12. *Dokážte vetu 3.5.3!*

Úloha 3.13. *Nájdite príklady množín Booleovských funkcií, pre ktoré vo vzťahoch uvedených vo vete 3.5.3 rovnosť nastáva (nenastáva)!*

Príklad 3.17. *Uvedieme niekoľko príkladov množín funkcií, ktoré sú/nie sú funkcionálne uzavreté.*

1. $\mathcal{P}_2$ *je uzavretá trieda Booleovských funkcií, lebo skladaním Booleovských dostaneme opäť Booleovskú funkciu.*
2. $\{\neg x\}$ *nie je uzavretá trieda Booleovských funkcií, lebo* $\neg\neg x = x \in [\{\neg x\}]$,
3. $\{1, x \oplus y\}$ *nie je uzavretá trieda Booleovských funkcií, lebo* $1 \oplus 1 = 0 \notin \{1, x \oplus y\}$
4. *množina* $[\mathcal{M}]$ *je uzavretá pre ľubovoľnú množinu Booleovských funkcií* $\mathcal{M}$.

Úloha 3.14. *Vytvorte aspoň 10 rozličných uzavretých množín Booleovských funkcií!*

Pomocou pojmu uzáver môžeme jednoducho definovať úplnosť množiny Booleovských funkcií $\mathcal{M}$: množina $\mathcal{M}$ tvorí úplný systém práve vtedy, ak $[\mathcal{M}] = \mathcal{P}_2$.

Veta 3.5.1 obsahovala kritérium, na základe ktorého bolo možné rozhodnúť, či je nejaká množina Booleovských funkcií úplná. Ak však úplná množina $\mathcal{F}$ obsahuje viacero Booleovských funkcií, toto kritérium nemusí byť pre praktické účely vhodné. Naviac, ak sa nám nepodarí vyjadriť nejakú funkciu z $\mathcal{F}$ pomocou formuly nad $\mathcal{G}$, nevieme povedať, či je to naša neschopnosť, alebo sa daná funkcia objektívne nedá vyjadriť pomocou formuly nad $\mathcal{G}$. V nasledujúcej časti preto odvodíme jednoduchšie kritérium úplnosti, ktoré pre danú množinu Booleovských funkcií dá jednoznačnú odpoveď, resp. lepšie povedané, ak sa jedná o množinu konkrétnych Booleovských funkcií, spomínanú kritérium úplnosti dá odpoveď áno, množina je úplná alebo nie, daná množina Booleovských funkcií netvorí úplný systém. Množina Booleovských funkcií $\mathcal{M}$ však nemusí byť zadaná len vymenovaním všetkých svojich prvkov, ale možno ju vyjadriť pomocou množinových operácií nad množinami Booleovských funkcií, resp. špecifikovaním vlastností, ktoré by Booleovské funkcie z danej množiny mali mať. V tomto prípade nemusí byť informácia o prvkoch množiny $\mathcal{M}$ (Booleovských funkciách) dostatočná a odpoveď môže znieť: ak v množine $\mathcal{M}$ existujú funkcie s takýmito vlastnosťami, tak potom (nie) je úplná.

3.6 Predúplné triedy. Veta o úplnosti

Pri zisťovaní úplnosti množiny Booleovských funkcií sa využíva 5 zvláštnych uzavretých množín Booleovských funkcií. Teraz tieto množiny charakterizujeme a potom vyslovíme a dokážeme vetu o úplnosti.

3.6.1 Triedy T_0 a T_1

Definícia 3.6.1. *Trieda*[20] *Booleovských funkcií*

$$T_0^n = \{f(x_1, \ldots, x_n) \in \mathcal{P}_2; f(0, \ldots, 0) = 0\}$$

sa nazýva triedou (n*-árnych) Booleovských funkcií zachovávajúcich* 0*. Trieda* n*-árnych Booleovských funkcií*

$$T_1^n = \{f(x_1, \ldots, x_n) \in \mathcal{P}_2; f(1, \ldots, 1) = 1\}$$

sa nazýva triedou (n*-árnych) Booleovských funkcií zachovávajúcich* 1*. Triedy Booleovských funkcií zachovávajúcich* 0*, resp.* 1 *potom definujeme nasledovne:*

$$T_0 = \bigcup_n T_0^n, \qquad T_1 = \bigcup_n T_1^n.$$

Tabuľka hodnôt n-árnej Booleovskej funkcie patriacej do triedy T_0 sa vyznačuje tým, že v prvom riadku má hodnotu 0, zatiaľ čo v ostatných $2^n - 1$ riadkoch može nadobúdať ľubovoľné hodnoty. Z toho výplýva, že, že trieda T_0 obsahuje 2^{2^n-1} n-árnych Booleovských funkcií. Podobne, n-árne Booleovské funkcie z triedy T_1 majú v poslednom riadku svojej pravdivostnej tabuľky hodnotu 1 a v ostatných $2^n - 1$ riadkoch možu nadobúdať ľubovoľné hodnoty. Potom zrejme T_1 obsahuje rovnako ako trieda T_0 2^{2^n-1} n-árnych Booleovských funkcií. Ukážeme, že trieda T_0 je uzavretá. Nech $f, g_1, \ldots, g_n \in T_0$ (bez ujmy na všeobecnosti môžeme predpokladať, že funkcie $f, g_1, \ldots, g_n$ sú n-árne a závisia od premenných $(x_1, \ldots, x_n)$. Nech je $F(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_1n(x_1, \ldots, x_n))$ zložená Booleovská funkcia. Potom

$$F(0, \ldots, 0) = f(g_1(0, \ldots, 0), \ldots, g_n(0, \ldots, 0)) = f(0, \ldots, 0) = 0,$$

a Booleovská funkcia $F(x_1, \ldots, x_n)$ patrí do T_0. To znamená, že trieda T_0 je uzavretá vzhľadom operáciu skladania Booleovských funkcií. Podobne by sme dokázali aj uvavretosť triedy T_1. Z elementárnych Booleovských funkcií patrí do triedy T_0 napríklad konjunkcia, disjunkcia, identická funkcia, súčet modulo 2; do triedy T_1 patrí konjunkcia, disjunkcia, indentická funkcia, ekvivalencia, implikácia. Negácia nepatrí ani do T_0 ani do T_1, implikácia nepatrí do T_0.

3.6.2 Trieda lineárnych funkcií, L

Definícia 3.6.2. *Booleovská funkcia* $f(x_1, \ldots, x_n)$ *sa nazýva* n*-árnou lineárnou Booleovskou funkciou, ak jej algebraická normálna forma obsahuje len lineárne členy; t.j.*

$$f(x_1, \ldots, x_n) = a_0 \oplus a_1x_1 \oplus \cdots \oplus a_nx_n$$

Triedou L^n *nazveme množinu všetkých* n*-árnych lineárnych Booleovských funkcií. Triedu* L *lineárnych Booleovských funkcií definujeme ako*

$$L = \bigcup_n L^n.$$

[20] v teórii Booleovských funkcií na označenie množiny funkcií často používa pojem trieda alebo systém Booleovských funkcií. Tejto konvencie sa budeme pridŕžať aj my.

Poznámka. Pri štúdiu kryptografických vlastností Booleovských funkcií sa Booleovské funkcie z triedy L nazývajú afinnými Booleovskými funkciami a pojem lineárna Booleovská funkcia sa rezervuje pre takú Booleovskú funkciu, ktorej AFN má tvar $f(x_1,\dots,x_n) = a_1x_1 \oplus \cdots \oplus a_nx_n$; t.j. koeficient a_0 v ANF je nulový. My sa budeme pridržiavať štandardného označenia.

Keďže ANF lineárnej Booleovskej funkcie má $n+1$ koeficientov, $|L^n| = 2^{n+1}$. Ukážeme ešte, že tieda lineárnych funkcií je uzavretá. Nech sú $f, g_1, \dots, g_n \in L$ n-árne Booleovské funkcie;

$$\begin{aligned} f(y_1,\dots,y_n) &= a_0 \oplus a_1y_1 \oplus \cdots \oplus a_ny_n, \\ g_1(x_1,\dots,x_n) &= b_{1,0} \oplus b_{1,1}x_1 \cdots \oplus b_{1,n}x_n, \\ g_2(x_1,\dots,x_n) &= b_{2,0} \oplus b_{2,1}x_1 \cdots \oplus b_{2,n}x_n, \\ &\dots \\ g_n(x_1,\dots,x_n) &= b_{n,0} \oplus b_{n,1}x_1 \cdots \oplus b_{n,n}x_n. \end{aligned}$$

Potom

$$\begin{aligned} F(x_1,\dots,x_n) &= f(g_1(x_1,\dots,x_n),\dots,g_n(x_1,\dots,x_n)) = \\ &= a_0 \oplus a_1(b_{1,0} \oplus b_{1,1}x_1 \cdots \oplus b_{1,n}x_n) \oplus \cdots \oplus a_n(b_{n,0} \oplus b_{n,1}x_1 \cdots \oplus b_{n,n}x_n) = \\ &= (a_0 \oplus a_1b_{1,0} \oplus \cdots \oplus a_nb_{n,0}) \oplus x_1(a_1b_{1,1} \oplus a_2b_{2,1} \oplus \cdots \oplus a_nb_{n,1}) \oplus \\ &\oplus\ x_2(a_1b_{1,2} \oplus a_2b_{2,2} \oplus \cdots \oplus a_nb_{n,2}) \oplus \cdots \oplus x_n(a_1b_{1,n} \oplus a_2b_{2,n} \oplus \cdots \oplus a_nb_{n,n}) = \\ &= c_0 \oplus c_1x_1 \oplus \cdots \oplus c_nx_n. \end{aligned}$$

To znamená, že zložená funkcia F je lineárna, resp. trieda L je uzavretá vzhľadom na operáciu skladania Booleovských funkcií.

Úloha 3.15. *Dokážte uzavretosť tried* T_0, T_1, L *bez predpokladov, že*

- *všetky čiastkové funkcie sú* n*-árne,*
- *čiastkové funkcie závisia od tých istých premenných.*

3.6.3 Trieda monotónnych funkcií, M

V matematickej analýze sme sa stretli s pojmami monotónne rastúcej reálnej funkcie; $f(x)$ bola monotónne rastúca, ak platilo $\forall x_0, x_1 \in \mathbb{R}[(x_0 < x_1) \Rightarrow (f(x_0) < f(x_1))]$. Zavedenie monotónnosti pre Booleovské funkcie naráža na problém—ako porovnávať binárne vektory? Ani lexikografické usporiadanie, ani usporiadanie založení na tom, že binárne vektory reprezentujú prirodzené čísla nebolo použiteľné. Preto na množine binárnych vektorov dĺžky n najprv definujeme reláciu čiastočného usporiadania a potom pomocou neho zavedieme pojem monotónnej Booleovskej funkcie.

Definícia 3.6.3. *Nech* $\alpha, \beta \in \{0,1\}^n$*;* $\alpha = (a_1, \ldots, a_n), \beta = (b_1, \ldots, b_n)$*. Budeme hovoriť, že vektor* α *predchádza vektor* β *práve vtedy, ak* $a_i \leq b_i \quad i = 1 \ldots, n$*. Tto skutočnosť budeme symbolicky zapisovať nasledovne:* $\alpha \preceq \beta$

Príklad 3.18. *Vektor* $(0,0,1)$ *predchádza vektor* $0,1,1$*. Medzi vektormi* $(0,1)$ *a* $(1,0)$ *neexistuje vzťah predchádzania, pretože* $a_1 < b_1$ *a* $a_2 > b_2$*. Takéto vektory sa nazývajú neporovnateľné. Keďže relácia* $\preceq$ *je definovaná na vektoroch rovnakej dĺžky, nedá sa aplikovať na vektory rozličnej dĺžky: napr.* $(0,0,1)$ *a* $(0,0)$*.*

Úloha 3.16. *Ilustrujte reláciu* $\preceq$ *na množine* $\{0,1\}^3$ *pomocou orientovaného grafu rádu 8. (Návod: vrcholu* v_i *priradíte vektor* $\sigma(3,i)$*,* $i = 0, \ldots, 7$*. Ak* $\sigma(3,i) \preceq \sigma(3,j)$*, vrcholy* v_i, v_j *spojte orientovanou hranou* (v_i, v_j)*!)*

Teraz zavedieme pojem *monotónnej Booleovskej funkcie*.

Definícia 3.6.4. *(*n*-árna)Booleovská funkcia* $f(x_1, \ldots, x_n)$ *sa nazýva monotónna, ak pre ľubovoľné dva vektory* $\alpha = (a_1, \ldots, a_n), \beta = (b_1, \ldots, b_n)$ *také, že* $\alpha \preceq \beta$ *platí* $f(a_1, \ldots, a_n) \leq f(b_1, \ldots, b_n)$*. Triedu všetkých* n*-árnych monotónnych Booleovských funkcií budeme označovať symbolom* M^n *a triedu všetkých monotónnych Booleovských funkcií budeme označovať symbolom* M*.*

Monotónnosť Booleovskej funkcie sa neprejavuje tak jednoducho v tabuľke pravdivostných hodnôt (ako v prípade funkcií zachovávajúcich hodnotu 0 alebo 1) ani v tvare ANF Booleovskej funkcie (ako v prípade afinných/lineárnych Booleovských funkcií.) Preto sa zatiaľ nepodarilo nájsť presné vyjadrenie pre mohutnosť M^n Pre veľké n platí tento asymptotický odhad: **??**

$$|M^n| \sim 2^{\binom{n}{n/2}} \exp\left[\binom{n}{(n/2)-1} \cdot \left(\frac{1}{2^{n/2}} + \frac{n^2}{2^{n+5}} + \frac{n}{2^{n+4}}\right)\right] \quad n \text{ je párne,}$$

resp. pre nepárne n

$$|M^n| \sim 2 \cdot 2^{\binom{n}{(n-1)/2}} \times$$
$$\times \quad \exp\left[\binom{n}{(n-3)/2} \cdot \left(\frac{1}{2^{(n+3)/2}} - \frac{n^2}{2^{n+6}} - \frac{n}{2^{n+3}}\right) + \binom{n}{(n-1)/2} \cdot \left(\frac{1}{2^{(n+1)/2}} + \frac{n^2}{2^{n+4}}\right)\right]$$

Zápis $a_n \sim b_n$ vyjadruje skutočnosť, že $\lim_{n\to\infty} \frac{a_n}{b_n} = 1$ a číta sa „a_n je asymptoticky rovné b_n.“

Dokážeme uzavretosť triedy M. Nech $f(y_1, \ldots, y_m), g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n) \in M$. Potom zložená funkcia

$$F(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$$

je monotónnou funkciou. Skutočne, nech sú $\alpha =, \beta = (b_1, \ldots, b_n)$ ľubovoľné dva binárne vektory také, že $\alpha \preceq \beta$. Keďže $g_1, \ldots d_m$ sú monotónne funkcie, platí pre ne

$$\begin{aligned} c_1 = g_1(a_1, \ldots, a_n) &\leq g_1(b_1, \ldots, b_n) = d_1 \\ c_2 = g_2(a_1, \ldots, a_n) &\leq g_2(b_1, \ldots, b_n) = d_2 \\ &\ldots \\ c_m = g_m(a_1, \ldots, a_n) &\leq g_m(b_1, \ldots, b_n) = d_m \end{aligned}$$

To však znamená, že $(c_1, \ldots, c_m) \preceq (d_1, \ldots, d_m)$. Ale aj funkcia $f(y_1, \ldots, y_m)$ je monotónna, a teda

$$f(c_1, \ldots, c_m) \leq f(d_1, \ldots, d_m).$$

Pre zloženú funkciu F postupne dostávame:

$$F(a_1, \ldots, a_n) = f(g_1(a_1, \ldots, a_n), \ldots, g_m(a_1, \ldots, a_n) =$$
$$= f(c_1, \ldots, c_m) \leq f(d_1, \ldots, d_m) = f(g_1(b_1, \ldots, b_n), \ldots, g_m(b_1, \ldots, b_n)) = F(b_1, \ldots, b_n).$$

Príklad 3.19. *Konjunkcia, disjunkcia, obidve konštanty a identická funkcia sú monotónne funkcie, negácia, implikácia, ekvivalencia, súčet modulo 2 nie sú monotónne funkcie.*

3.6.4 Trieda samoduálnych funkcií S

Samoduálna funkcia sa vyznačuje takou veľkou symetriou svojej tabuľky pravdivostných hodnôt, že na úplné zadanie samoduálnej Booleovskej funkcie stačí polovica jej tabuľky pravdivostných hodnôt. Na druhej strane, samoduálnosť je menej názorná ako ostatné vlastnosti Booleovských funkcií. Začneme preto jednoduchším pojmom duálnosti Booleovských funkcií.

Definícia 3.6.5. *Booleovská funkcia* $f(x_1, \ldots, x_n)$ *sa nazýva duálnou funkciou k Booleovskej funkcii* $g(x_1, \ldots, x_n)$*, ak*

$$f(x_1, \ldots, x_n) = \overline{g}(\overline{x}_1, \ldots, \overline{x}_n).$$

Ilustrujeme si pojem duálnosti funkcií na príkladoch.

Príklad 3.20. 1. *Funkcia* $f_1(x, y)$ *(konjunkcia) je duálna ku funkcii* $f_7(x, y)$ *(disjunkcii):*

$$\overline{\overline{x} \& \overline{y}} = (x \vee y)$$

a opačne disjunkcia je duálnou funkciou konjunkcie, lebo

$$\overline{\overline{x} \vee \overline{y}} = (x \& y)$$

2. *funkcia* $f_{12}(x, y)$ *(negácia) je duálna k sebe samej, lebo* $\neg\neg(\neg x) = \neg x$ *a funkcia* $f_3(x, y)$ *(identita) je duálna k sebe samej, lebo* $\neg(\neg x) = x$.

Zavedieme pojem samoduálnej funkcie:

Definícia 3.6.6. *Booleovská funkcia* $f(x_1, \ldots, x_n)$ *sa nazýva samoduálnou funkciou, ak*

$$f(x_1, \ldots, x_n) = \overline{f}(\overline{x}_1, \ldots, \overline{x}_n);$$

t.j. ak je duálna k sebe samej.

Trieda samoduálnych funkcií je neprázdna, pretože podľa predchádzajúceho príkladu medzi samoduálne funkcie patria napríklad identická funkcia x a negácia $\neg x$. Tabuľka pravdivostných hodnôt samoduálnej funkcie sa vyznačuje tým, že v riadkoch prislúchajúcich opačným vektorom vstupných hodnôt sú opačné hodnoty. Názorne si to ukážeme na nasledujúcom príklade.

x_1	x_2	x_3	$\overline{x}_1$	$\overline{x}_2$	$\overline{x}_3$	$f(x_1,x_2,x_3)$	$\overline{f}(\overline{x}_1,\overline{x}_2,\overline{x}_3)$
0	0	0	1	1	1	1	0
0	0	1	1	1	0	0	1
0	1	0	1	0	1	0	1
0	1	1	1	0	0	1	0

Tabuľka 3.24: Tabuľka samoduálnej funkcie

Príklad 3.21. *Definujme samoduálnu funkciu* $fx_1,x_2,x_3)$ *troch premenných. Aby sme dosiahli samoduálnosť funkcie* f *musíme zaistiť komplementárnosť hodnôt na opačných vektoroch. V tabuľke 3.24 sú kvôli názornosti uvedené opačné vektory v susedných stĺpcoch.*

Nech napríklad $f(0,0,0) = 1$. *Potom* $f(1,1,1) = 1$ *a* $\neg f(1,1,1) = 0$. *Zvolíme hodnoty funkcie* $fx_1,x_2,x_3)$ *na prvých 4 vektoroch, napríklad* $f(0,0,0) = 1, f(0,0,1) = 0, f(0,1,0) = 0, f(0,1,1) = 1$ *Tým sú jednoznačne definované hodnoty funkcie* $fx_1,x_2,x_3)$ *aj na opačných vektoroch—pozri tabuľku 3.24.*

Ako sme videli v predchádzajúcom príklade, na jednoznačné určenie samoduálnej Booleovskej funkcie stačí určiť jej hodnotu na jednom z každej dvojice opačných vektorov. To znamená, že n-árna samoduálna Booleovská funkcia je zadaná napr. prvou polovicou tabuľky, resp. binárným vektorom dĺžky 2^{n-1}. Z uvedeného faktu potom vyplýva, že n-árnych samoduálnych Booleovských funkcií je $2^{2^{n-1}} = \sqrt{2^{2^n}}$. Ukážeme, že aj trieda S samoduálnych funkcií je uzavretá na skladanie Booleovských funkcií. Nech sú $f(y_1,\ldots,y_m), g_1(x_1,\ldots,x_n),\ldots,g_m(x_1,\ldots,x_n) \in S$. Potom zložená funkcia

$$F(x_1,\ldots,x_n) = f(g_1(x_1,\ldots,x_n),\ldots,g_m(x_1,\ldots,x_n))$$

je samoduálnou funkciou. Negujeme najprv premenné zloženej funkcie F:

$$F(\overline{x}_1,\ldots,\overline{x}_n) = f(g_1(\overline{x}_1,\ldots,\overline{x}_n),\ldots,g_m(\overline{x}_1,\ldots,\overline{x}_n))$$

Keďže funkcie $g_1,\ldots,g_m$ sú samoduálne, platí pre ne

$$g_i(\overline{x}_1,\ldots,\overline{x}_n) = \overline{g}_i(x_1,\ldots,x_n).$$

Vonkajšia čiastková funkcia zloženej funkcie F, funkcia f je tiež samoduálna, preto

$$f(g_1(x_1,\ldots,x_n),\ldots,g_m(x_1,\ldots,x_n)) = \overline{f}(\overline{g}_1(x_1,\ldots,x_n),\ldots,\overline{g}_m(x_1,\ldots,x_n)),$$

resp.

$$f(\overline{g}_1,\ldots,\overline{g}_m) = \overline{f}(g_1,\ldots,g_m).$$

Z vyššie uvedených vzťahov vyplýva, že

$$F(\overline{x}_1,\ldots,\overline{x}_n) = \overline{f}(g_1(x_1,\ldots,x_n),\ldots,g_m(x_1,\ldots,x_n)) = \overline{F}(x_1,\ldots,x_n),$$

a teda trieda samoduálnych Booleovských funkcií je uzavretá.

Teraz môžeme sformulovať kritérium úplnosti systému Booleovských funkcií.

Veta 3.6.1. *(O funkcionálnej úplnosti) Množina Booleovských funkcií* D *tvorí úplný systém Booleovských funkcií práve vtedy, ak nie je podmnožinou žiadnej z tried* T_0, T_1, L, S, M.

Dôkaz. Nutnosť. Triedy T_0, T_1, L, S, M sú uzavreté a žiadna z nich nie je úplná (pre každú z nich vieme nájsť Booleovskú funkciu, ktorú neobsahuje). Ak by D bola úplná a zároveň bola podmnožinou niektorej z tried T_0, T_1, L, S, M, napríklad $D \subseteq M$, potom by podľa tvrdenia 3 vety 3.5.3 muselo platiť

$$D \subseteq M \Rightarrow [M] = M \supseteq [D] = \mathcal{P}_2.$$

Spor.

Ukážeme, že podmienka je aj postačujúca; t.j. ak D nie je podmnožinou žiadnej z tried T_0, T_1, L, S, M, tak dokážeme vytvoriť formuly nad D realizujúce funkcie $\neg x, x\&y$, ktoré tvoria úplný systém Booleovských funkcií.

Keďže D nie je podmnožinou žiadnej z tried T_0, T_1, L, S, M, môžeme predpokladať, že obsahuje funkcie $f_0 \notin T_0, f_1 \notin T_1, f_S \notin S, f_M \notin M$. Funkcie f_0, f_1, f_M, f_L, f_S nemusia byť nutne rôzne a f_0, f_1 sa nezhodujú s funkciami f_0, f_1 z tabuľky 3.1. Bez ujmy na všeobecnosti môžeme predpokladať, že všetky uvedené funkcie sú n-árne.

1. Najprv zoberieme funkciu f_0. Keďže $f_0 \notin T_0$, $f_0(0, \ldots, 0) = 1$. Stotožníme všetky premenné funkcie f_0 a vytvoríme novú funkciu jednej premennej: $\phi(x) = f_0(x, \ldots, x)$. Je zrejmé, že $\phi(0) = 1$. Pozrieme sa na opačný koniec tabuľky pravdivostných hodnôt Booleovskej funkcie f_0, aby sme zistili, akú hodnotu nadobúda funkcia $\phi(x)$ pre $x = 1$. Sú dve možnosti:

   (a) $f_0(1, \ldots, 1) = 1$. V tomto prípade funkcia $\phi(x) \equiv 1$ nemá podstatné premenné a predstavuje konštantu 1.

   (b) V druhom prípade $f_0(1, \ldots, 1) = 0$, a teda $\phi(1) = 0$, resp. $\phi(x) = \neg x$.

2. Teraz využijeme funkciu $f_1 \notin T_1$. Tak ako v predchádzajúcom prípade stotožníme všetky premenné funkcie f_1 a vytvoríme novú funkciu jednej premennej: $\psi(x) = f_1(x, \ldots, x)$. Platí $\psi(1) = f_1(1, \ldots, 1) = 0$. Zaujma nás hodnota $\psi(0) = f_1(0, \ldots, 0)$. Podobne ako v prípade funkcie $\phi(x)$, môžu aj tu nastať dve možnosti

   (a) $\psi(0) = f_1(0, \ldots, 0) = 0$. V tomto prípade $\psi(x) \equiv 0$ predstavuje konštantu 0.

   (b) Ak $\psi(0) = f_1(0, \ldots, 0) = 1$, $\psi(x) = \neg x$.

   V optimálnom prípade sme z funkcií f_0, f_1 vytvorili obe konštanty a negáciu, v horšom prípade buď obe konštanty alebo samotnú negáciu (Obr.3.16). Ukážeme, že v prípade, keď sme skonštruovali „len“ obe konštanty, vytvoríme pomocou nemonotónnej funkcie negáciu.

3. Keďže $f_M \notin M$, existujú také dva vektory hodnôt $\alpha = (a_1, \ldots, a_n), \beta = (b_1, \ldots, b_n)$ také, že $\alpha \preceq \beta$ a

   $$f(a_1, \ldots, a_n) = 1, \quad f(b_1, \ldots, b_n) = 0.$$

   Keďže $\alpha \preceq \beta$, potom existujú také hodnoty $i_1, \ldots, i_k$, že $a_{i_j} = 0, b_{i_j} = 1, \quad j = 1, \ldots, k$ a $a_i = b_i$ pre $i \notin \{i_1, \ldots, i_k\}$. Bez ujmy na všeobecnosti môžeme predpokladať, že sa vektory α, β odlišujú na prvých k miestach a na zostávajúcich $n - k$ miestach majú rovnaké hodnoty, t.j.:

   $$\alpha = (0, \ldots, 0, a_{k+1}, \ldots, a_n), \qquad \beta = (1, \ldots, 1, a_{k+1}, \ldots, a_n).$$

Definujeme teraz funkciu $\vartheta(x) = f_M(x, \ldots, x, a_{k+1}, \ldots, a_n)$. Pre funkciu $\vartheta(x)$ platí

$$\begin{aligned} \vartheta(0) &= f_M(0, \ldots, 0, a_{k+1}, \ldots, a_n) = 1 \\ \vartheta(1) &= f_M(1, \ldots, 1, a_{k+1}, \ldots, a_n) = 0. \end{aligned}$$

Potrebné konštanty sme vytvorili z funkcií f_0, f_1. Funkcia jednej premennej $\vartheta(x)$ predstavuje negáciu.

Ak sa nám z funkcií f_0, f_1 podarilo vytvoriť len negáciu, obe konštanty získame pomocou negácie a funkcie f_S, ktorá nie je samoduálna.

4. Keďže $f_S \notin S$, existujú také dva navzájom opačné vektory hodnôt $\alpha = (a_1, \ldots, a_n)$ $\overline{\alpha} = (\overline{a}_1, \ldots, \overline{a}_n)$, na ktorých funkcia f_S nadobúda rovnakú hodnotu;

$$f_S(a_1, \ldots, a_n) = f_S(\overline{a}_1, \ldots, \overline{a}_n).$$

(Pripomenieme význam označenia x^σ, ktoré sme zaviedli na začiatku tejto kapitoly: $x^\sigma = \overline{x}$ ak $\sigma = 0$ a $x^\sigma = x$ ak $\sigma = 1$.) Podobne ako v predchádzajúcich prípadoch využijeme funkciu f_S na vytvorenie Booleovskej funkcie jednej premennej. Položíme

$$\omega(x) = f_S(x^{a_1}, \ldots, x^{a_n}).$$

Takúto funkciu dokážeme zostrojiť pomocou negáciu, ktorú sme už vytvorili. Z definície výrazu x^σ vyplýva, že $1^\sigma = \sigma$ a $0^\sigma = \neg\sigma$. Pomocou týchto vzťahov vyjadríme hodnoty funkcie $\omega(x)$ (jednej premennej):

$$\omega(1) = f_S(1^{a_1}, \ldots, 1^{a_n}) = f_S(a_1, \ldots, a_n) = f_S(\overline{a}_1, \ldots, \overline{a}_n) = f_S(0^{a_1}, \ldots, 0^{a_n}) = \omega(0).$$

Funkcia $\omega(x)$ teda predstavuje konštantu. Z funkcie $\omega(x)$ nedokážeme síce zostrojiť predpísanú konštantu, ale to nepredstavuje žiaden problém, pretože druhú konštantu poľahky vytvoríme pomocou negácie a funkcie $\omega(x)$.

5. Výsledkom našich doterajších snažení sú funkcie $0, 1, \neg x$. Z tchto troch funkcií a nelineárnej funkcie $f_L \notin L$ vytvoríme konjunkciu. Z toho, že funkcia f_L nie je lineárna, vyplýva že v jej ANF sa vyskytuje konjunkcia aspoň dvoch premenných. Bez ujmy na všeobecnosti môžeme predpokladať, že ide o premenné x_1, x_2. Využijeme komutatívnosť a asociatívnosť sčítania modulo 2, distributívny zákon pre konjunkciu a súčet modulo 2 a upravíme ANF Booleovskej funkcie f_L na nasledujúci tvar:

$$\begin{aligned} f_L(x_1, \ldots, x_n) &= x_1x_2 \& f_{(1,2)}(x_3, \ldots, x_n) \oplus x_1 \& f_{(1)}(x_3, \ldots, x_n) \oplus \\ &\oplus \; x_2 \& f_{(2)}(x_3, \ldots, x_n) \oplus f_{(\emptyset)}(x_3, \ldots, x_n). \end{aligned} \tag{3.17}$$

Preusporiadali sme všetky členy ANF Boleovskej funkcie $f_L(x_1, \ldots, x_n)$ a rozdelili ich do 4 skupín (zátvoriek) tak, že prvá skupina pozostáva zo všetkých tých členov ANF, ktoré obsahujú konjunkciu x_1x_2, druhú tvoria tie členy ANF, ktoré obsahujú premennú x_1 ale nie x_2, tretiu—tie členy ANF, ktoré obsahujú premennú x_2 ale nie x_1 a napokon, do poslednej sme zaradili tie členy ANF, ktoré neobsahujú ani x_1, ani x_2. Zo všetkých členov prvej skupiny sme na základe distributívneho zákona vyňali pred zátvorku konjunkciu x_1x_2 a výraz v zátvorke vyjadrili pomocou Booleovskej funkcie $f_{(1,2)}(x_3, \ldots, x_n)$. Rovanako sme upravili ostatné tri skupiny členov ANF.

Všimnite si, že po vyňatí x_1x_2, x_1, x_2 z prvej, druhej, resp. tretej skupiny členov pred zátvorku, zostávajúce výrazy v zátvorkách už neobsahovali premenné x_1, x_2. Posledná, štvrtá skupina pozostávala z členov ANF, ktoré nezáviseli od x_1, x_2. To znamená, že výrazy v zátvorkách predstavujú funkcie premenných $x_3, \ldots, x_n$.

Nakoľko ANF Booleovskej funkcie f_L obsahuje konjunkciu x_1x_2, musí existovať aspoň jeden taký súbor hodnôt $(a_3, \ldots, a_n)$ premenných $x_3, \ldots, x_n$, že $f_{(1,2)}(a_3, \ldots, a_n) = 1$. Vytvoríme novú Booleovskú funkciu dvoch premenných $\Phi(x_1, x_2)$ dosadením konštánt $(a_3, \ldots, a_n)$ do funkcie f_L:

$$\Phi(x_1, x_2) = f_L(x_1, x_2, a_3, \ldots, a_n).$$

Zo zápisu funkcie f_L 3.17 vyplýva, že

$$\begin{aligned} \Phi(x_1, x_2) &= x_1x_2 \& f_{(1,2)}(a_3, \ldots, a_n) \oplus x_1 \& f_{(1)}(a_3, \ldots, a_n) \oplus x_2 \& f_{(2)}(a_3, \ldots, a_n) \oplus \\ &\oplus f_{(\emptyset)}(a_3, \ldots, a_n) = d_0x_1x_2 \oplus d_1x_1 \oplus d_2x_2 \oplus d_3. \end{aligned} \tag{3.18}$$

Koeficienty d_0, d_1, d_2, d_3 sú kontanty—hodnoty funkcií $f_{(1,2)}, f_{(1)}, f_{(2)}, f_{(\emptyset)}$ na vektore $(a_3, \ldots, a_n)$. Je zrejmé, že $d_0 = 1$. Ak by boli ostatné koeficienty d_1, d_2, d_3 nulové, funkcia $\Phi(x_1, x_2)$ by už predstavovala potrebnú konjunkciu. Ale konjunkciu z $\Phi(x_1, x_2)$ ľahko vytvoríme aj v prípade, keď je aspoň jeden z koeficientov d_1, d_2, d_3 nenulový. Na základe hodnôt d_1, d_2 transformujeme premenné funkcie $\Phi(x_1, x_2)$ a výslednú funkciu ešte v prípade potreby negujeme. Dostávame opäť funkciu dvoch premenných $\Theta(x_1, x_2)$:

$$\Theta(x_1, x_2) = \Phi(x_1 \oplus d_2, x_2 \oplus d_1) \oplus d_1d_2 \oplus d_3. \tag{3.19}$$

Pripomíname, že $x \oplus 1 = \neg x$ a $x \oplus 0 = x$. Dokážeme, že $\Theta(x_1, x_2) = x_1 \& x_2$. Vyjadríme $\Theta(x_1, x_2)$ pomocou vzťahov 3.18 a 3.19.

$$\begin{aligned} \Theta(x_1, x_2) &= (x_1 \oplus d_2)(x_2 \oplus d_1) \oplus d_1(x_1 \oplus d_2) \oplus d_2(x_2 \oplus d_1) \oplus d_3 \oplus d_1d_2 \oplus d_3 = \\ &= x_1x_2 \oplus x_1d_1 \oplus x_2d_2 \oplus d_1d_2 \oplus x_1d_1 \oplus d_1d_2 \oplus x_2d_2 \oplus d_1d_2 \oplus d_3 \oplus d_1d_2 \oplus d_3 = \\ &= x_1 \& x_2. \end{aligned}$$

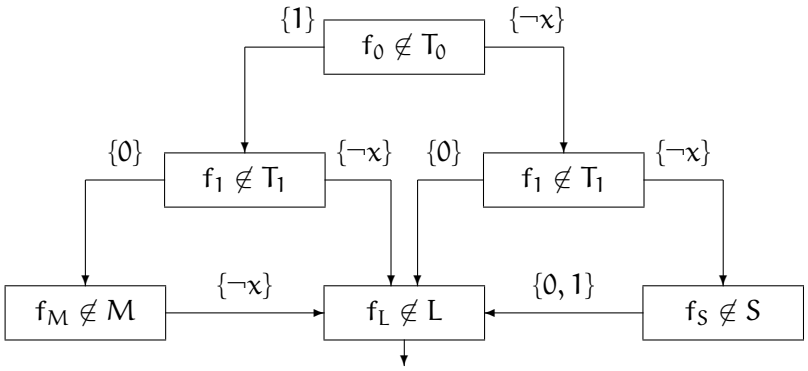
Všetky úpravy funkcie f_L, ktoré sme robili, sa dali realizovať pomocou dosadzovania konštánt a negácie premenných, resp. negácie funkcie. To znamená, že pomocou konštánt $0, 1$, negácie $\neg x$ a nelineárnej funkcie možno vytvoriť konjunkciu.

Pomocou pätice funkcií f_0, f_1, f_M, f_L, f_S sme vytvorili funkcie $\neg x, x_1 \& x_2$, ktoré tvoria úplný systém Booleovských funkcií. To znamená, že tak funkcie f_0, f_1, f_M, f_L, f_S, ako aj trieda/množina D tvoria úplný systém Booleovských funkcií. Prehľadná schéma dôkazu tejto vety je zobrazená na obrázku 3.16. □

Príklad 3.22. *Dôkaz vety 3.6.1 je pomerne dlhý a zložitý. Kvôli lepšiemu pochopeniu ho teraz ilustrujeme na dvoch konkrétnych príkladoch.*

1. *Ukážeme najprv, že množina Booleovských funkcií $\{x \Rightarrow y, 0\}$ tvorí úplný systém.*

   (a) *Funkcia $x \Rightarrow y$ nepatrí do triedy T_0, lebo $0 \Rightarrow 0 = 1$. Keďže $1 \Rightarrow 1 = 1$, funkcia $x \Rightarrow x$ predstavuje konštantu 1,*



Obr. 3.16: Schéma dôkazu vety 3.6.1

(b) Funkcia 0 *nepatrí do triedy* T_1.

(c) Implikácia nie je monotónnou funkciou, lebo $0 \Rightarrow 0 = 1$*, ale* $1 \Rightarrow 0 = 0$*. Skonštruujeme pomocou nej a konštanty* 0 *negáciu:* $x \Rightarrow 0 \equiv \neg x$.

(d) Funkcia $x \Rightarrow y$ *nie je lineárna. Zostrojíme jej ANF a z nej vytvoríme konjunkciu. Všeobecný tvar ANF Booleovskej funkcie dvoch premenných je:*

$$f(x, y) = a_0 \oplus a_1 x \oplus a_2 y \oplus a_3 xy.$$

Určíme hodnoty jednotlivých koefcientov funkcie $f(x, y) = x \Rightarrow y$*. Jednotlivé kroky odvodenia sú uvedené v nasledujúcej tabuľke*

hodnota	*rovnica*	*koeficient*
$f(0,0) = 1$	$a_0 = 1$	$a_0 = 1$
$f(1,0) = 0$	$1 \oplus a_0 = 0$	$a_1 = 1$
$f(0,1) = 1$	$1 \oplus a_2 = 1$	$a_2 = 0$
$f(1,1) = 1$	$1 \oplus 1 \oplus a_3 = 1$	$a_3 = 1$

ANF implikácie $x \Rightarrow y$ *má teda tvar:*

$$x \Rightarrow y = 1 \oplus x \oplus xy \qquad (= \Phi(x, y)).$$

Zostrojili sme funkciu $\Phi(x, y)$*, v ktorej koeficienty* d_i *nadobúdajú nasledujúce hodnoty:* $d_0 = d_1 = d_3 = 1$ *a* $d_2 = 0$*. Vytvoríme funkciu* $\Theta(x, y)$*:*

$$\Theta(x, y) = \Phi(x, y \oplus 1) \oplus 1 = \neg\Phi(x, \neg y) = [(x \Rightarrow (y \Rightarrow 0)) \Rightarrow 0].$$

2. Ukážeme, ako sa upravuje zložitejšia ANF.

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= 1 \oplus x_1 \oplus x_2 \oplus x_1x_2 \oplus x_2x_3 \oplus x_3x_4 \oplus x_1x_3x_4 \oplus x_1x_2x_3 \oplus x_1x_2x_3x_4 = \\ &= x_1x_2(1 \oplus x_3 \oplus x_3x_4) \oplus x_1(1 \oplus x_3x_4) \oplus x_2(x_1 \oplus x_3) \oplus (1 \oplus x_3x_4). \\ f(x_1, x_2, 0, 1) &= \Phi(x_1, x_2) = x_1x_2 \oplus x_1 \oplus x_2 \oplus 1, \\ f(\overline{x}_1, \overline{x}_2, 0, 1) &= \Theta(x_1, x_2) = x_1x_2. \end{aligned}$$

funkcia	T_0	T_1	L	S	M
0	+	−	+	−	+
1	−	+	+	−	+
x	+	+	+	+	+
$\neg x$	−	−	+	+	−
$x\&y$	+	+	−	−	+
$x \vee y$	+	+	−	−	+
$x \Rightarrow y$	−	+	−	−	−
$x \equiv y$	−	+	+	−	−
$x \oplus y$	+	−	+	−	−
xNANDy	−	−	−	−	−
xNORy	−	−	−	−	−

Tabuľka 3.25: Príslušnosť elementárnych Booleovských funkcií do tried T_0, T_1, L, S, M.

Všimnite si, že rovnica

$$(1 \oplus x_3 \oplus x_3x_4) = 1$$

má tri riešenia:

(a) $x_3 = x_4 = 0$,

(b) $x_3 = 0, x_4 = 1$,

(c) $x_3 = x_4 = 1$.

Úloha 3.17. *Každá uzavretá trieda* $A \subsetneq \mathcal{P}_2$ *je obsiahnutá v aspoň jednej z tried* T_0, T_1, L, S, M.

Úloha 3.18. *Zostrojte Vennov diagram pre množiny* n*-árnych Booleovských funkcií* T_0^n, T_1^n, L^n, S^n *a určte mohutnosti všetkých 16 množín, ktoré sú pomocou neho definované!*

Pri skúmaní úplnosti nejakej množiny Booleovských funkcií, resp. pri konštrukcii úplného systému Booleovských funkcií môže byť užitočná nasledujúca tabuľka: Aby bola nejaká množina Booleovských funkcií úplná, musí mať v každom zo stĺpcov T_0, T_1, L, S, M aspoň jeden znak −. Preto sú napríklad množiny $\{\neg x, x \vee y\}, \{x \Rightarrow y, x \oplus y\}$ úplné, ale $\{x\&y, x \vee y\}$ nie je úplná. Rozhodnúť o tom, či nejaká zložitejšia Booleovská funkcia patrí/nepatrí do tried L, S, M na základe definícií tried môže byť pomerne náročné. Pre praktické použitie však často vystačíme so slabšími ale podstatne jednoduchšími kritériami. Na ich zavedenie potrebujeme jeden jednoduchý pojem.

Definícia 3.6.7. n*-árna Booleovská funkcia* f *je balancovaná, ak má množina jej jednotkových vektorov (vektorov pravdivostných hodnôt, na ktorých* f *nadobúda hodnotu* 1*) mohutnosť* $2^{2^{n-1}}$.

- Ak je Booleovská funkcia samoduálna, tak je balancovaná.
- Ak je Booleovská funkcia lineárna, tak je buď konštantná, alebo balancovaná.

- Ak je n-árna Booleovská funkcia $f(x_1, \ldots, x_n)$ monotónna a $f(0, \ldots, 0) = 1$, alebo $f(1, \ldots, 1) = 0$, tak $f(x_1, \ldots, x_n)$ je konštantná.

Podľa týchto kritérií rýchle zaradíme napríklad implikáciu: nie je to ani konštantná, ani balancovaná funkcia, preto nemôže patriť do tried L, S, na vektore $(0, 0)$ nadobúda hodnotu 1, a teda nie je monotónna.

Triedy Booleovských funkcií T_0, T_1, L, S, M sú výnimočné, predstavujú jediné tzv. predúplné triedy Booleovských funkcií v $\mathcal{P}_2$. Preskúmame ich vlastnosti podrobnejšie.

Definícia 3.6.8. *Množina Booleovských funkcií $A \subseteq \mathcal{P}_2$ sa nazýva predúplnou triedou (Booleovských funkcií), ak $[A] \neq \mathcal{P}_2$, ale pre ľubovoľnú Booleovskú funkciu f, ktorá nepatrí do A platí $[A \cup \{f\}] = \mathcal{P}_2$.*

Poznámka. Predúplnosť triedy A znamená:

1. A je uzavretá, pretože ináč by sme mohli zobrať Booleovskú funkciu $f \in [A] - A$, pre ktorú by potom platilo: $[A \cup \{f\}] = [A] \neq \mathcal{P}_2$.
2. A je neúplná trieda, lebo $[A] \neq \mathcal{P}_2$, ale
3. triede A chýba k úplnosti tak málo, že stačí zobrať ľubovoľnú Booleovskú funkciu, ktorá do triedy A nepatrí, pridať ju k A, aby sme dostali úplný systém.

Zdôrazňujeme ešte raz, že poslednú uvedenú vlastnosť musí mať ľubovoľná a nie špeciálne vybraná funkcia z A^c. V opačnom prípade by totiž ľubovoľná uzavretá neúplná množina Booleovských funkcií tvorila predúplný systém, ktorý by sa dal „zúplniť“ pridaním napríklad Shefferovej alebo Pierceovej funkcie (NAND alebo NOR).

Nasledujúca veta je fakticky dôsledkom vety 3.6.1. Kvôli závažnosti jej obsahu ju formulujeme ako samostatnú vetu.

Veta 3.6.2. *V triede $\mathcal{P}_2$ existuje práve päť predúplných tried;* T_0, T_1, L, S, M.

Dôkaz Všetky triedy T_0, T_1, L, S, M sú neúplné a uzavreté. Stačí, aby sme o každej z nich ukázali, že nie je podmnožinou inej (predúplnej) triedy. Tento dôkaz ponechávame čitateľovi ako cvičenie. My pri dôkaze budeme vychádzať priamo z definície predúplnej triedy, vety 3.6.1 a tabuľky 3.25.

Predúplnosť triedy T_0. Trieda T_0 obsahuje okrem iných Booleovských funkcií aj funkcie $x, x\&y, x \oplus y$. Vyberieme ľubovoľnú funkciu $f \notin T_0$. Potom na základe tejto funkcie buď vytvoríme negáciu, ktorá spolu s konjunkciou tvorí úplný systém, alebo zostrojíme konštantu 1, dosadíme ju do funkcie $x \oplus y$ a dostávame negáciu $x \oplus 1$, ktorá potom spolu s konjunkciou tvorí úplný systém.

Predúplnosť triedy T_1. Vyberieme ľubovoľnú funkciu $f \notin T_1$. Ak z tejto funkcie vytvoríme negáciu, tak už máme úplný systém, lebo konjunkcia patrí do triedy T_1. Ak pomocou f vytvoríme „len“ konštantu 0, tak použijeme implikáciu z triedy T_1 a vytvoríme negáciu v tvare $x \Rightarrow 0$.

Predúplnosť triedy M Obe koštanty, konjunkcia a disjunkcia sú monotónne funkcie. Z nemonotónnej Booleovskej funkcie $f \notin M$ dosadzovaním konštánt zostrojíme negáciu, ktorá spolu s (monotónnou) konjunkciou tvorí úplný systém.

Predúplnosť triedy L Trieda lineárnych funkcií obsahuje obe konštanty aj negáciu. Z nelineárnej Booleovskej funkcie $f \notin L$ vytvoríme konjunkciu.

Predúplnosť triedy S Trieda samoduálnych funkcií obsahuje negáciu. Pomocou negácie a nesamoduálnej funkcie $f \notin S$ vytvoríme obe konštanty. Trieda n-árnych samoduálnych funkcií obsahuje $2^{2^{n-1}}$ a trieda n-árnych lineárnych funkcií obsahuje len 2^{n+1} Booleovských funkcií. To znamená, že v triede S existuje samoduálna a zároveň nelineárna funkcia aspoň troch premenných, z ktorej vytvoríme konjunkciu.

Predpokladajme, že v triede $\mathcal{P}_2$ existuje ďalšia predúplná trieda, označme ju X. Trieda X je uzavretá a nesmie byť obsiahnutá v žiadnej z predúplných tried T_0, T_1, L, S, M. To však znamená, že X je úplná. Spor. □

Úloha 3.19. *Zistite či sú nasledujúce množiny Booleovských funkcií úplné:*

1. $x\&y, 0, 1, x \oplus y \oplus z$
2. $x\&y, x \Rightarrow y, 1$
3. $x \oplus y, x \equiv y, x \Rightarrow y$
4. $x \Rightarrow (y\&\neg z)$
5. $x \oplus (y \vee z), x \Rightarrow z$
6. $\neg x \vee \neg y$
7. $\neg x\&\neg y$
8. $x \Rightarrow \neg y$
9. $x \vee \neg y$

Úloha 3.20. *Dokážte, že neexistuje samoduálna nelineárna funkcia dvoch premenných!*

Úloha 3.21. *Nájdite všetky samoduálne nelineárne Booleovské funkcie troch premenných! Návod: vytvorte ANF Booleovskej funkcie 3 premenných* $f(x, y, z)$ *a riešte rovnosť* $f(x, y, z) = \overline{f}(\overline{x}, \overline{y}, \overline{z})$ *vzhľadom na koeficienty* $a_0, \ldots, a_7$.

Hoci množiny Booleovských funkcií môžu byť veľmi rozsiahle, ale ako sa ukázalo v predchádzajúcich vetách, úplnosť množiny Booleovských funkcií môže zaistiť už jej malá podmnožina. V ideálnom prípade bude obsahovať jednu Booleovskú funkciu—NAND, NOR alebo podobnú Booleovskú funkciu tvoriacu úplný systém. V najhoršom prípade by to nemalo byť viac, ako vetou 3.6.1 garantovaných 5 funkcií. Nasledujúca veta ukazuje, že sa horný odhad na počet funkcií tvoriacich minimálny úplný podsystém dá ešte trocha stlačiť.

Veta 3.6.3. *Z každej úplnej množiny* D *Booleovských funkcií možno vybrať úplnú podmnožinu ubsahujúcu najviac 4 Booleovské funkcie.*

Dôkaz. Ponechávame čitateľovi ako cvičenie. □

Zatiaľ sme uvažovali o úplnosti vzhľadom na triedu všetkých Booleovských funkcií, $\mathcal{P}_2$. Pojem úplnosti je však možné zovšeobecniť aj na ľubovoľnú inú uzavretú triedu.

Definícia 3.6.9. *Množina Booleovských funkcií* $\{f_1, \ldots, f_k, \ldots\}$ *uzavretej triedy* A *sa nazýva úplnou v triede* A*, ak sa jej uzáver rovná* A.

Definícia 3.6.10. *Množina Booleovských funkcií* $\{f_1, \ldots, f_k, \ldots\}$ *uzavretej triedy* A *sa nazýva bázou triedy* A*, ak je úplná v triede* A *a žiadna jej vlastná podmnožina nie je úplná v triede* A.

Príklad 3.23. *Bázy tried Booleovských funkcií.*

1. $\{x\&y, \neg x\}$ *tvorí bázu* $\mathcal{P}_2$*,*
2. $\{0, 1, x \vee y, x\&y\}$ *tvorí bázu* M*,*
3. $\{1, x \oplus y\}$ *tvorí bázu* L.

Na záver tejto kapitoly uvedieme dva výsledky, ktoré pre uzavreté triedu Booleovských funkcií dokázal americký matematik Emil Post.

Veta 3.6.4. *Každá uzavretá trieda z* $\mathcal{P}_2$ *má konečnú bázu.*

Veta 3.6.5. *Mohutnosť množiny uzavretých tried v* $\mathcal{P}_2$ *je spočítateľná.*

Zoznam tabuliek