➢ Matematická logika, Booleova algebra

Petr:

Konjunkce, disjunkce, implikace, ekvivalence,… a k tomu ještě podivné značky. K čemu to je vůbec dobré? Na průmyslovce jsme pracovali s logickými funkcemi a Booleovou algebrou. Bylo to v pohodě, používali jsme logický součin a logický součet a bylo jasné k čemu to je. Je to totéž, nebo není?

JaJ:

Petře,
nejste v tom sám, spousta lidí v tom nemá úplně jasno. Je to pravděpodobně způsobeno tím, že jste se na střední škole jen okrajově dotkli poměrně složitého matematického oboru, který jste nedotáhli k aplikacím. Máte ale výhodu, že se dokážete zeptat. Pojďme si o tom trochu popovídat.

Používání formální matematické logiky a její symboliky má v matematice (na jisté úrovni) zcela jistě své důležité místo. A jestli je práce s logickými funkcemi a Booleovou algebrou o tomtéž, nebo není? Tak tedy je, vlastně skoro. Rozhodně ze stejného nebo podobného soudku. A tím je věda o logice jako taková.

Logika jako formální věda o způsobu myšlení vyvozující jednoznačné závěry z jasně definovaných předpokladů vznikla původně jako součást filozofie. Za zakladatele logiky je obecně považován Aristoteles (384–322 př. n. l.). Vyvíjela se a postupem času našla své uplatnění i v mnoha jiných oborech. V současné době má své nezastupitelné místo i v matematice, v právních vědách, v genetice, v biologii, v meteorologii, aj. Zásadně podmínila vznik a vývoj všech informačních technologií. Vývoj probíhal v různých oborech víceméně odděleně, což je důvodem mírného formálního nesouladu ve značení i v některých přístupech.

V matematice i technických vědách se hlavně jedná o výrokovou logiku pracující s binárními informacemi, tzv. výroky, pro které má smysl tvrdit, že jsou pravdivé (obvykle značíme symbolem 1, [‚true']), nebo nepravdivé (značíme symbolem 0, [‚false']) a nastává vždy právě jedna z těchto možností. Pro práci s takovýmto typem informací, pro popis složených výroků a

vyhodnocení jejich pravdivosti jsou zavedeny a používají se výrokové operace a jejich označení (výrokové spojky):

a	¬a
1	0
0	1

negace

a	b	c = a∧b
1	1	1
1	0	0
0	1	0
0	0	0

konjunkce

a	b	c = a∨b
1	1	1
1	0	1
0	1	1
0	0	0

disjunkce

a	b	c = a⇒b
1	1	1
1	0	0
0	1	1
0	0	1

implikace

a	b	c = a⇔b
1	1	1
1	0	0
0	1	0
0	0	1

ekvivalence

Uvedená symbolika doplněná kvantifikátory: ∀ (pro všechny), ∃ (existuje alespoň jedno), ∃! (existuje právě jedno) nám dává možnost vyjádřit i složité logické vazby a souvislosti. Tvoří základ jakéhosi symbolického matematického nadnárodního metajazyka, který je srozumitelný komukoliv ve světě, tedy tomu, kdo se zabývá poněkud hlouběji matematikou. Velmi doporučuji si osvojit tento způsob komunikace, zcela jistě se s ním na vysoké škole budete setkávat a pomůže vám ve čtení a pochopení matematických textů.

Příklady:

$\forall\, x \in R\,;\; x^2 \geq 0$ — *„pro každé reálné číslo x platí, že jeho kvadrát je nezáporný“ (pravdivý výrok),*

$\exists\, n \in N\,;\; n^2 \in (0,1)$ — *„existuje alespoň jedno přirozené číslo n, pro které platí, že jeho kvadrát patří do intervalu“ (nepravdivý výrok),*

$\exists!\, k \in Z\,;\; k + 1 = 0$ — *„existuje právě jedno celé číslo k, pro které platí $k + 1 = 0$“ (pravdivý výrok).*

Je možné tímto způsobem vyjadřovat i velmi složité složené výroky a vyhodnocovat jejich pravdivost.

Velmi hezky je tato problematika zpracovaná např. na portálu ISIBALO jako soubor videí:

https://www.youtube.com/watch?v=ncNLKpBJFGA
https://www.youtube.com/watch?v=kvwlYJ2OjRI
https://www.youtube.com/watch?v=FqrnTXS-UTY
atd.

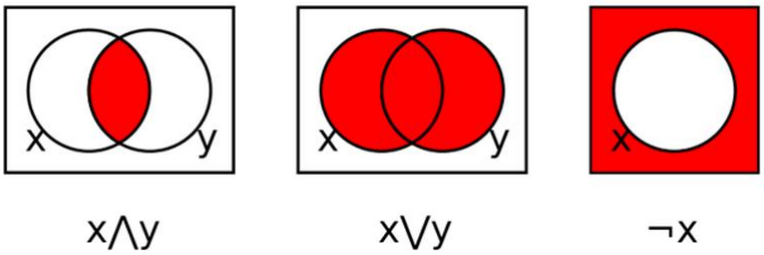
Nebo:

https://www2.karlin.mff.cuni.cz/~portal/logika/?page=title

Poznámka:

Na rozdíl od běžné matematiky, na kterou jste zvyklí, ve které je současná symbolika operací až na vzácné výjimky ustálená, v matematické logice tomu tak úplně není. Můžete se zde setkat i s jiným značením operací než je uvedeno výše. Poměrně často např. s označením negace apostrofem $\neg a = a'$ *(E. Schröder [1841-1902], Ch. S. Peirce [1839-1914]) nebo pruhem nad proměnnou* $\neg a = \bar{a}$ *(D. Hilbert [1862-1943]). Dnes již zřídka, ale přeci jen, také s:* $\neg a = \sim a$ *(G. Peano [1858-1932], B. Russell [1872-1970]) nebo:* $\neg a = Na$ *(J. Łukasiewicz [1862-1943]) nebo dokonce s:* $\neg a = -a$; $\neg a = !\, a$; $\neg a = NOT\ a$. *Podobně i u jiných operací, konjunkce:* $a \wedge b = a \cdot b = ab = a\& b = a \times b = Kab = a\ AND\ b$; *disjunkce:* $a \vee b = a + b = Aab = a\ OR\ b$.

Základní pravidla výrokové logiky mají jistou podobnost se základy teorie množin, pro reprezentaci množinových operací lze použít Vennův diagram. Je to jen jiná interpretace téhož základního principu.



x⋀y x⋁y ¬x

Matematická logika se stala v současné době netriviální součástí matematiky poskytující formální aparát teorie důkazu, zabývá se pojmy jako úplnost, splnitelnost, rozhodnutelnost, konzistence atd. a vyvíjí se vlastním směrem.

V průběhu času prodělala práce se složenými výroky značný vývoj, byly objeveny a dokázány některé důležité zákony a identity. Usnadnily úpravu komplikovaných logických vět, zjednodušily techniku ověřování jejich formální pravdivosti, např.:

De Morganovy zákony:

$\neg(a \vee b \vee c \vee \cdots) \Leftrightarrow \neg a \wedge \neg b \wedge \neg c \wedge \cdots$
$\neg(a \wedge b \wedge c \wedge \cdots) \Leftrightarrow \neg a \vee \neg b \vee \neg c \vee \cdots$

Augustus De Morgan [1806 – 1871] britský matematik

$(a \Rightarrow b) \Leftrightarrow \neg a \vee b$
$(a \Leftrightarrow b) \Leftrightarrow (a \wedge b) \vee (\neg a \wedge \neg b)$

náhrada implikace a ekvivalence pomocí konjunkce, disjunkce, negace

$a \vee a \Leftrightarrow a \quad ; \quad a \wedge a \Leftrightarrow a$
$a \vee \neg a \Leftrightarrow 1 \quad ; \quad a \wedge \neg a \Leftrightarrow 0$
$a \vee 0 \Leftrightarrow a \quad ; \quad a \wedge 0 \Leftrightarrow 0$
$a \vee 1 \Leftrightarrow 1 \quad ; \quad a \wedge 1 \Leftrightarrow a$

zákony agresivit a neutrálnosti

komutativní, asociativní a distributivní zákony

$a \vee b \Leftrightarrow b \vee a \quad ; \quad a \wedge b \Leftrightarrow b \wedge a$
$(a \vee b) \vee c \Leftrightarrow a \vee (b \vee c) \quad ; \quad (a \wedge b) \wedge c \Leftrightarrow a \wedge (b \wedge c)$
$(a \vee b) \wedge c \Leftrightarrow (a \wedge c) \vee (b \wedge c) \quad ; \quad (a \wedge b) \vee c \Leftrightarrow (a \vee c) \wedge (b \vee c)$

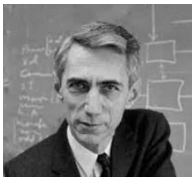
Výše uvedené je pouze ukázka základních pravidel a souvislostí, aparát matematické logiky je mnohem širší a obsáhlejší. Zmiňme alespoň Fregův zákon a jeho kalkul (F. L. Gottlob Frege [1848-1925]), zákon Dunse Scota (J. Duns Scotus [1266-1308]) nebo Hilbertův predikátový kalkul (David Hilbert [1862-1943]), používaný v oblasti predikátové logiky k popisu a k analýze matematických teorií a vět.

Booleova algebra

Booleova algebra, nazvaná podle irského matematika George Boolea [1815 –1864], je užitečná v mnoha matematických disciplínách a našla své široké uplatnění i v technických aplikacích. Z počátku se jednalo jen o snahu po zjednodušení zápisu matematických vět a jejich důkazů. O metodu jak zpřehlednit práci s logickými výroky a zobecnit vlastnosti množinových a logických operací.
Jedná se o precizně axiomaticky definovanou matematickou strukturu definovanou nad obecně neprázdnou množinou dvěma binárními a jednou unární operací, které splňují jisté vlastnosti. Povšimněme si, že je původně definovaná šířeji než ji většinou chápeme dnes v aplikaci ve výrokové logice nad dvouprvkovou množinou pravda/nepravda. V této aplikaci jsou binární operace tvořeny konjunkcí a disjunkcí, unární operace negací.

V Booleově algebře nefigurují operace implikace a ekvivalence, které se dají vyjádřit pomocí konjunkce a disjunkce (viz výše uvedené vztahy). V definici jsou uvedené pouze tři operace, které tvoří tzv. úplný soubor operací, kterými se dá vyjádřit jakýkoliv složený logický výrok.

Použití Booleovy algebry v technických disciplínách tvoří základ současného vývoje číslicových počítačů a informačních technologií. George Boole je proto někdy (možná trochu neprávem) považován za zakladatele informatiky, ačkoli v jeho době nebylo o počítačích ještě ani uvažováno.



V roce 1937, tedy 73 let po smrti G. Booleho, si uvědomil tehdy mladý jednadvacetiletý americký budoucí matematik a elektrotechnický inženýr Claude Elwood Shannon [1916 – 2001], že lze Booleovu algebru použít k popisu a ke konstrukci složitých reléových sítí. Tuto myšlenku v témže roce publikoval a obhájil ve své diplomové práci na univerzitě v Michiganu. Pokud by bylo na mně, pokládal bych za zakladatele informatiky raději C. D. Shannona (hlavně za jeho

další práce a obrovský přínos tomuto oboru). I když, pravda, teoretickým základem byla skutečně exaktní matematická Booleova algebra.

V technickém prostředí se postupně spíše vžilo značení logických spojek, které původně navrhoval D. Hilbert: místo disjunkce $a \vee b$ logický součet $a + b$, místo konjunkce $a \wedge b$ logický součin $a \cdot b$ nebo jen ab , místo ce $\neg a$ značit $\bar{a}$ a rovnost (ekvivalenci) rovnítkem $a = b$. S touto symbolikou jste pravděpodobně pracovali také i vy na průmyslovce.

Pro úpravy a jednodušší vyjádření složených výroků vznikla a rozvinula se technika práce s pravdivostní tabulkou a s Karnaughovou mapou **[***M. Karnaugh, ∗ 1924].*

Poznámka:

Můžeme se setkat i s méně používanou variantou vyjádření logické funkce tzv. Svobodovou mapou (A. Svoboda [1907 – 1980], český konstruktér prvních československých počítačů SAPO a EPOS1, v roce 1964 emigroval do USA, zemřel při pozorování výbuchu sopky Mount St. Helens roku 1980). Svobodova mapa je indexována jiným způsobem než Karnaughova mapa, sousední pole se neliší pouze v jedné proměnné (jako je to u Karnaqughovy mapy). Není proto vhodná pro jednoduchou intuitivní minimalizaci vyjádření logické funkce. Je to i důvod jejího menšího rozšíření.

Poznámka:

Z hlediska exaktní matematické teorie algebry je výroková logika definovaná nad dvouprvkovou množinou pravda/nepravda (vč. výše uvedené varianty používané v technice) tzv. modelem obecnější Booleovy algebry definované nad obecně neprázdnou (tedy ne nutně dvouprvkovou) množinou. Zcela obdobně to platí i o aparátu běžně používaných množinových operací.

Některé důležité zákony a identity vyjádřené formalizmy Booleovy algebry:

$$\overline{(a + b + c + \cdots)} = \bar{a} \cdot \bar{b} \cdot \bar{c} \cdot \cdots$$
$$\overline{(a \cdot b \cdot c \cdot \cdots)} = \bar{a} + \bar{b} + \bar{c} + \cdots$$

De Morganovy zákony

$$\bar{1} = 0 \quad ; \quad \bar{0} = 1 \quad ; \quad \bar{\bar{a}} = a$$
$$a + a = a \quad ; \quad a \cdot a = a$$
$$a + \bar{a} = 1 \quad ; \quad a \cdot \bar{a} = 0$$
$$a + 0 = a \quad ; \quad a \cdot 0 = 0$$
$$a + 1 = 1 \quad ; \quad a \cdot 1 = a$$

zákony agresivity a neutrálnosti

$$a + b = b + a \quad ; \quad a \cdot b = b \cdot a$$
$$(a + b) + c = a + (b + c) \quad ; \quad (a \cdot b) \cdot c = a \cdot (b \cdot c)$$
$$(a + b) \cdot c = (a \cdot c) + (b \cdot c) \quad ; \quad (a \cdot b) + c = (a + c) \cdot (b + c)$$

komutativní, asociativní a distributivní zákony

$$a + (a \cdot b) = a \quad ; \quad a \cdot (a + b) = a$$
$$a + (\bar{a} \cdot b) = a + b \quad ; \quad a \cdot (\bar{a} + b) = a \cdot b$$

zákony absorpce

Pozor, v reálné algebře neplatí!
$(a \cdot b) + c \neq (a + c) \cdot (b + c) \quad ; \quad a, b, c \in R$

Obvyklá vyjádření logické funkce:

- *pravdivostní tabulkou,*
- *mapou,*
- *algebraickým výrazem.*

1. Negace *(inverze, NOT)*

a	$\bar{a}$
1	0
0	0

pravdivostní tabulka

$y = f(a) = \bar{a}$

Booleova algebra

technické vyjádření

$y \Leftrightarrow \neg a$

matematická výroková logika

Vennův diagram

množinové vyjádření

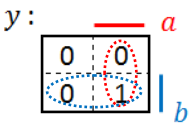
2. Logický součin *(konjunkce, průnik, AND, „toto a současně i ono“)*

a	b	$y = a \cdot b$
1	1	1
1	0	0
0	1	0
0	0	0

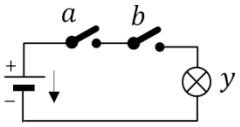
pravdivostní tabulka

$y = f(a, b) = a \cdot b$

Booleova algebra



Karnaughova mapa

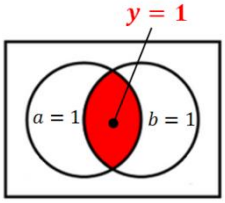


Shannonova interpretace

technické vyjádření

$y \Leftrightarrow a \wedge b$

matematická výroková logika



Vennův diagram

množinové vyjádření

3. Logický součet *(disjunkce, sjednocení, OR, „toto nebo ono“)*

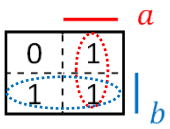
a	b	$y = a + b$
1	1	1
1	0	1
0	1	1
0	0	0

pravdivostní tabulka

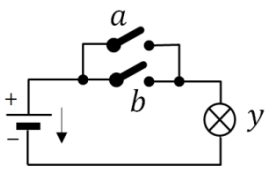
$$y = f(a, b) = a + b$$

Booleova algebra

y :



Karnaughova mapa

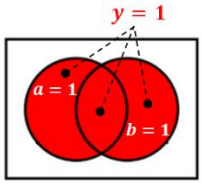


Shannonova interpretace

technické vyjádření

$y \Leftrightarrow a \vee b$

matematická výroková logika



Vennův diagram

množinové vyjádření

PŘÍKLADY

použití Booleovy algebry v některých z její modelů

I. ***Zjednodušení výrokové formule*** *složeného výroku*

Postup:

Upravíme výrokovou formuli tak, aby obsahovala pouze konjunkce, disjunkce a negace (úplný soubor booleovských funkcí). Potom ji převedeme do dvouprvkové Booleovy algebry s využitím jejího modelu výrokové logiky. Upravíme vzniklý booleovský výraz a výsledek převedeme zpět na formuli výrokové logiky.

Příklad 1:

Upravte výrokovou formuli: $(A \wedge B) \vee \big((\neg A \vee \neg B) \wedge (A \Rightarrow B)\big)$.

– *Úprava implikace pomocí:* $(A \Rightarrow B) \Leftrightarrow \neg A \vee B$.

$$(A \wedge B) \vee \Big((\neg A \vee \neg B) \wedge \underbrace{(A \Rightarrow B)}\Big) =$$
$$= (A \wedge B) \vee \big((\neg A \vee \neg B) \wedge (\neg A \vee B)\big) .$$

– *Přepis do Booleovy algebry a úprava:*

$$a \cdot b + \big((\bar{a} + \bar{b}) \cdot (\bar{a} + b)\big) = ab + \underbrace{\bar{a}\bar{a}}_{\bar{a}} + \bar{a}b + \bar{a}\bar{b} + \underbrace{\bar{b}b}_{0} =$$
$$= ab + \underbrace{\bar{a} + \bar{a}b + \bar{a}\bar{b}}_{} = ab + \bar{a}\Big(\underbrace{1 + b + \bar{b}}_{1}\Big) = ab + \bar{a} .$$

– *Zpětné převedení:*

$$ab + \bar{a} \sim (A \wedge B) \vee \neg A ,$$
$$(A \wedge B) \vee \big((\neg A \vee \neg B) \wedge (A \Rightarrow B)\big) \Leftrightarrow (A \wedge B) \vee \neg A .$$

II. Důkaz výrokové ekvivalence

Příklad 2:

Dokažte platnost distributivního zákona:

$$(A \wedge B) \vee C \Leftrightarrow (A \vee C) \wedge (B \vee C) .$$

Poznámka:
Jedná se o ekvivalenci, jejíž analogie v reálné algebře neplatí:

$$(a \cdot b) + c \neq (a + c) \cdot (b + c) \;;\; a, b, c \in R .$$

Levá strana ekvivalence:

$$X = (A \wedge B) \vee C \sim ab + c = x ,$$

pravá strana ekvivalence:

$$Y = (A \vee C) \wedge (B \vee C) \sim (a + c) \cdot (b + c) = ab + ac + bc + \underbrace{cc}_{c} =$$
$$= ab + \underbrace{ac + bc + c}_{} = ab + c\Big(\underbrace{a + b + 1}_{1}\Big) = ab + c = y ,$$

$X \Leftrightarrow Y$,

resp.:

$$X \Leftrightarrow Y = X \Leftrightarrow X = (X \wedge X) \vee (\neg X \wedge \neg X) = X \vee \neg X = 1 .$$

Dospěli jsme k tzv. tautologii, prokázali jsme, že zkoumaný složený výrok je vždy pravdivý bez ohledu na pravdivostní hodnoty jeho parametrů. Tím je pravdivost výroku zadané ekvivalence dokázána.

Opakem tautologie je tzv. kontradikce, kterou bychom prokázali naopak spor tvrzení.

III. Množinové operace

Příklad 3:

Dokažte, že platí množinová rovnost:

$$\big((A \cap B) \cap C\big) \cup \big((A \cap B) \cap C'\big) = A \cap B \ .$$

Přepis do Booleovy algebry a úprava:

- *levá strana:*

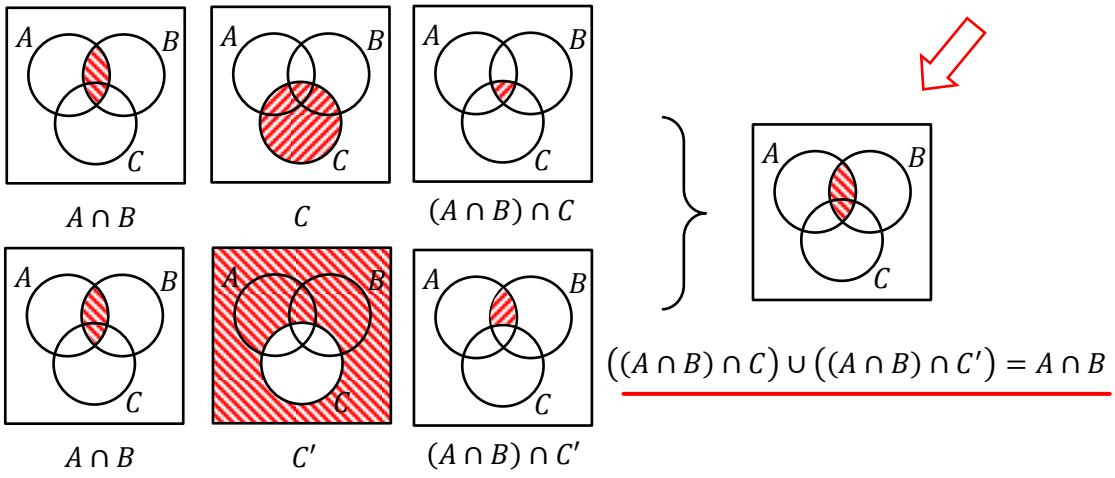
$$\big((a \cdot b) \cdot c\big) + \big((a \cdot b) \cdot \bar{c}\big) = abc + ab\bar{c} = ab\Big(\underbrace{c + \bar{c}}_{1}\Big) = a \cdot b \ ,$$

- *pravá strana:*

$$a \cdot b \ ,$$

levá strana se rovná pravé straně, rovnost tedy platí.

Vennův diagram:



Příklad 4:

Zjednodušte množinový zápis: $\big(A \cap (B \cup C)\big)' \cap (A \cap D)' \cap (C \cap D')$.

Přepis do Booleovy algebry a úprava:

$$\underbrace{\overline{a \cdot (b + c)}} \cdot \underbrace{\overline{a \cdot d}} \cdot (c \cdot \bar{d}) = \Big(\bar{a} + \underbrace{\overline{b + c}}\Big) \cdot (\bar{a} + \bar{d}) \cdot c \cdot \bar{d} =$$

$$= (\bar{a} + \bar{b}\bar{c}) \cdot \Big(\bar{a}c\bar{d} + c\underbrace{\bar{d}\bar{d}}_{\bar{d}}\Big) = (\bar{a} + \bar{b}\bar{c}) \cdot \Big(\underbrace{\bar{a}c\bar{d} + c\bar{d}}\Big) =$$

$$= (\bar{a} + \bar{b}\bar{c}) \cdot \Big(\underbrace{\bar{a} + 1}_{1}\Big) \cdot c\bar{d} = (\bar{a} + \bar{b}\bar{c}) \cdot c\bar{d} = \bar{a}c\bar{d} + \bar{b}\underbrace{\bar{c}c}_{0}\bar{d} = \bar{a}c\bar{d} \ .$$

- *Zpětné převedení:*

$$\big(A \cap (B \cup C)\big)' \cap (A \cap D)' \cap (C \cap D') = A' \cap C \cap D' \ .$$

IV. Obecná situační logika

Příklad 5:

K soudu byli předvedeni tři podezřelí z krádeže A, B a C. Při výslechu byly zjištěny tyto skutečnosti:

1. *krádež nespáchal A nebo ji nespáchal B,*
2. *když krádež nespáchal B, nespáchal ji ani A,*
3. *C byl na místě činu právě tehdy, když tam nebyl A.*

Kdo je pachatelem krádeže?

Nejprve převedeme situaci formálně do výrokové logiky zavedením parametrů a přiřazením jejich pravdivostních hodnot:

A je: vinen … A=1, nevinen … A=0, *B je: vinen … B=1, nevinen … B=0,* *C je: vinen … C=1, nevinen … C=0.*

Současně musí platit:

„krádež nespáchal A nebo ji nespáchal B“ $\neg A \wedge \neg B$,

„když krádež nespáchal B, nespáchal ji ani A“ $\neg B \Rightarrow \neg A$,

„C byl na místě činu právě tehdy, když tam nebyl A“ $\neg A \Rightarrow C$,

$$(\neg A \wedge \neg B) \wedge \underbrace{(\neg B \Rightarrow \neg A)} \wedge \underbrace{(\neg A \Rightarrow C)} =$$
$$= (\neg A \wedge \neg B) \wedge (\neg\neg B \vee \neg A) \wedge (\neg\neg A \vee C) \ .$$

Přepis do Booleovy algebry a úprava:

$$(\bar{a}\bar{b}) \cdot \left(\underbrace{\bar{\bar{b}}}_{b} + \bar{a}\right) \cdot \left(\underbrace{\bar{\bar{a}}}_{a} + c\right) = \bar{a}\bar{b} \cdot (b + \bar{a}) \cdot (a + c) =$$
$$= \left(\bar{a}\,\underbrace{\bar{b}b}_{0} + \underbrace{\bar{a}\bar{a}}_{\bar{a}}\,\bar{b}\right) \cdot (a + c) = \bar{a}\bar{b} \cdot (a + c) = \underbrace{\bar{a}a}_{0}\,\bar{b} + \bar{a}\bar{b}c = \bar{a}\bar{b}c \ ,$$
$$\neg A \wedge \neg B \wedge C = 1 \quad \rightarrow \quad A = B = 0 \ ; \ C = 1 \ .$$

Pachatelem krádeže je podezřelý C.

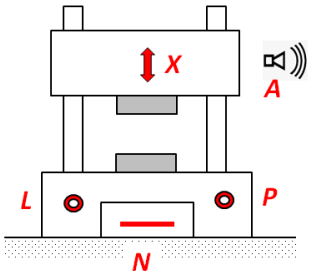
V. Návrh kombinačního obvodu

Poznámka:

Kombinačním obvodem rozumíme reálný obvod, který realizuje logickou funkci, jejíž funkční (výstupní) hodnota jednoznačně odpovídá okamžité kombinaci parametrů (vstupních hodnot). Jinou skupinou jsou tzv. sekvenční obvody, jejichž výstupní hodnoty jsou určeny nejen okamžitou kombinací vstupních hodnot, ale i sekvencí jejich minulých kombinací. Jejich konstrukce vyžaduje použití paměťových prvků.

Příklad 6:

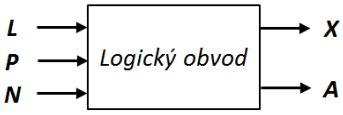
Navrhněte kombinační logický obvod sloužící k zajištění bezpečnosti práce na tvářecím lisu.



Lis se spouští do pracovního pohybu X nožním pedálem N. Kvůli bezpečnosti lis pracuje pouze tehdy, když je stisknutý nožní pedál a současně jsou stisknutá tlačítka levé L a pravé P ruky. Pokud tomu tak není, lis nepracuje a ozve se výstražný alarm A.

Navrhněte a realizujte logický obvod zajišťující bezpečný provoz.

Logické proměnné, jejich význam a označení:



Vstupní proměnné:

L … tlačítko levé ruky: stisknuté L=1,
uvolněné L=0,

P … tlačítko pravé ruky: stisknuté P=1,
uvolněné P=0,

N … nožní pedál: stisknutý N=1,
uvolněný N=0.

Výstupní proměnné:

X … pohyb lisu: pracuje X=1,
stop X=0,

A … alarm: výstraha A=1,
klid A=0.

Vyjádření logické funkce:

L	P	N	X	A
1	1	1	1	0
1	1	0	0	0
1	0	1	0	1
1	0	0	0	0
0	1	1	0	1
0	1	0	0	0
0	0	1	0	1
0	0	0	0	0

pravdivostní tabulka

X:

0	0	0	0
0	0	1	0

(L / P, N)

Karnaughova mapa

$$X = LPN$$

A:

0	0	0	0
1	1	0	1

(L / P, N)

$$A = \bar{L}\bar{P}N + L\bar{P}N + \bar{L}PN$$

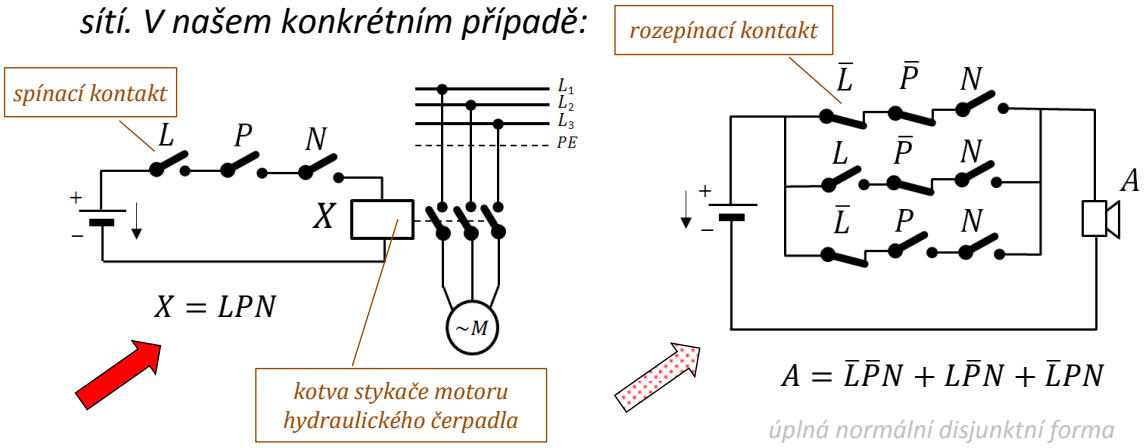
Booleova algebra

Vyjádření logické funkce v algebraickém tvaru je důležité pro následnou technickou realizaci logického obvodu. Mohli jsme ho vytvořit buď přímo z pravdivostní tabulky nebo z Karnaughovy

mapy tak, že každé políčko obsahující 1 (pravdivostní hodnotu ‚true'/pravda) vyjádříme součinem jednotlivých vstupních proměnných odpovídajících této kombinaci (tzv. minterm). Kolik jedniček, tolik součinů všech vstupních proměnných (v negaci nebo nenegovaných, podle polohy mintermu), všechny v logickém součtu. Dostali jsme tak jednoznačné, ale nejsložitější možné vyjádření logické funkce (tzv. úplná disjunktivní/součtová normální forma).

Realizace pomocí kontaktního pole

Jak jsme zmínili již v předcházejícím textu, v roce 1937 si uvědomil tehdy mladičký C. D. Shannon, že lze algebraického zápisu logické funkce využít k její technické realizaci pomocí přepínačů a reléových sítí. V našem konkrétním případě:

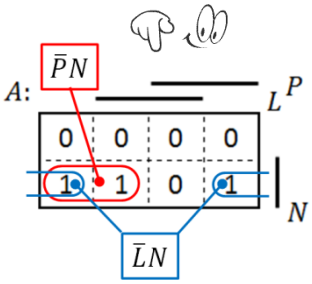


Složitost realizace je přímo úměrná složitosti vyjádření odpovídající logické funkce. Minimalizací vyjádření logické funkce snadno ovlivníme komplikovanost (a často i cenu) realizace. Funkci ovládání motoru lisu X minimalizovat kvůli její jednoduchosti již dále nejde, funkci aktivace alarmu A však ano. Minimalizujeme buď algebraickými prostředky, nebo v mapě.

$$A = \bar{L}\bar{P}N + L\bar{P}N + \bar{L}PN = \underbrace{\bar{L}\bar{P}N + \bar{L}\bar{P}N}_{\bar{L}\bar{P}N} + L\bar{P}N + \bar{L}PN =$$

$$= \underbrace{\bar{L}\bar{P}N + L\bar{P}N}_{} + \underbrace{\bar{L}PN + \bar{L}\bar{P}N}_{} = \bar{P}N\underbrace{(\bar{L} + L)}_{1} + \bar{L}N\underbrace{(\bar{P} + P)}_{1} =$$

$$= \bar{P}N + \bar{L}N = \underbrace{(\bar{P} + \bar{L})N}_{} .$$



Výhodou Karnaughovy mapy je indexování její polí tak, že sousední pole se vzájemně liší pouze v jedné proměnné. To nám umožňuje jednoduchou intuitivní minimalizaci vyjádření logické funkce přímo v mapě.

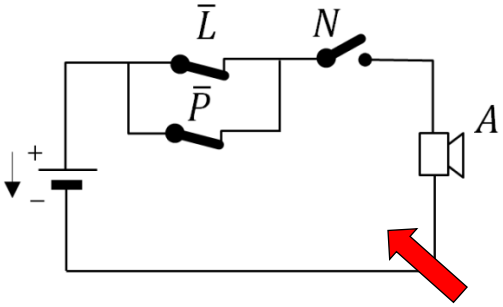
Term $\bar{P}N$ $(P = 0, N = 1)$ je reprezentován tělesem v mapě označený červeně, term $\bar{L}M$ $(L = 0, N = 1)$ je reprezentován modrým tělesem. Protože vytváříme součtovou (konjunktivní) formu vyjádření, tělesa

se mohou překrývat (termy jsou v součtu). Výsledné vyjádření po následném vytknutí N:

$$A = \bar{P}N + \bar{L}N = (\bar{P} + \bar{L})N \ .$$

Minimalizací oběma způsoby jsme dostali stejné vyjádření. Minimalizace v mapě je jen jiný, možná trochu jednodušší a snadnější, způsob minimalizace vyjádření logické funkce. Odpovídající zapojení kontaktního pole:

Je zřejmé, že se jedná o výrazně jednodušší kombinaci zapojení.



minimální realizace

Možná vás napadne, že zapojení je natolik logické, že bychom ho byli schopni navrhnout i bez složité teorie. Je to však způsobeno jednoduchostí zvoleného příkladu, ve složitějších případech je jasná strategie přístupu výhodou.

Poznámka:

Shannonova myšlenka odstartovala překotný vývoj technických aplikací, strojů na zpracování informací a informačních technologií. První československý počítač SAPO byl v 50. létech minulého století realizován pomocí reléových sítí, brzy však byly výpočetní stroje nahrazeny elektronkovými, později tranzistorovými a polovodičovými systémy vysoké integrace. Technologický vývoj způsobil i to, že realizace kombinačních logických obvodů pomocí reléových sítí se staly již zastaralými a prakticky se nepoužívají. Pro pochopení podstaty problému a názornou technickou realizaci je to i dnes, podle mého soudu, ilustrativní a užitečné.

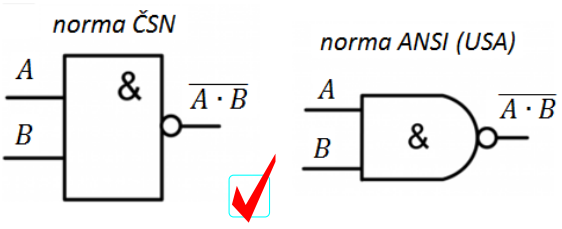
Realizace pomocí TTL integrovaných obvodů

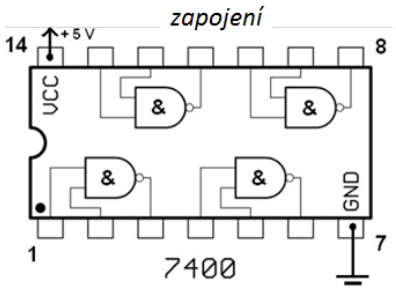
Technologický vývoj směřoval k součástkové základně na bázi bipolárních křemíkových tranzistorů TTL integrovaných obvodů (Tranzistor-Tranzistor-Logic). Tato technologie brzy vytlačila reléové i elektronkové systémy a rychle se rozšířila díky své jednoduchosti, rychlosti, spolehlivosti a ceně (cena jednotlivých hradel je dnes srovnatelná s cenou několika rohlíků).

Booleova algebra využívá k vyjádření logické funkce trojici základních funkcí: negaci, logický součin (konjunkci) a logický součet (disjunkci), tzv. úplný soubor logických funkcí. Ukázalo se však, že úplným souborem může být pouze i jediná funkce, např. NAND ($\overline{A \cdot B}$, Shefferova funkce, H. M. Scheffer [1882 - 1914]), NOR ($\overline{A + B}$, Peirceova funkce, Ch. S. Peirce [1839 - 1914]), XNOR ($AB + \bar{A}\bar{B}$, ekvivalence), aj. Touto jedinou funkcí lze vyjádřit negaci, logický součin i součet.

*Asi nejrozšířenější v technické praxi se stala funkce NAND: $\bar{A} = \overline{A \cdot A}$, $A + B = \overline{\bar{A} \cdot \bar{B}}$, $A \cdot B = \overline{\bar{A} + \bar{B}}$. Lze tak pomocí součástkové základny jednoho typu s výhodou realizovat libovolnou logickou funkci. Technickým standardem se stala řada 74** firmy Texas Instruments*

s několika variantně i vícevstupovými hradly (vždy negace součinu všech vstupů) zalisovanými do plastového pouzdra. Napájení 5V, logická 1 ≈5V, logická 0 ≈0V. Schématické značení:

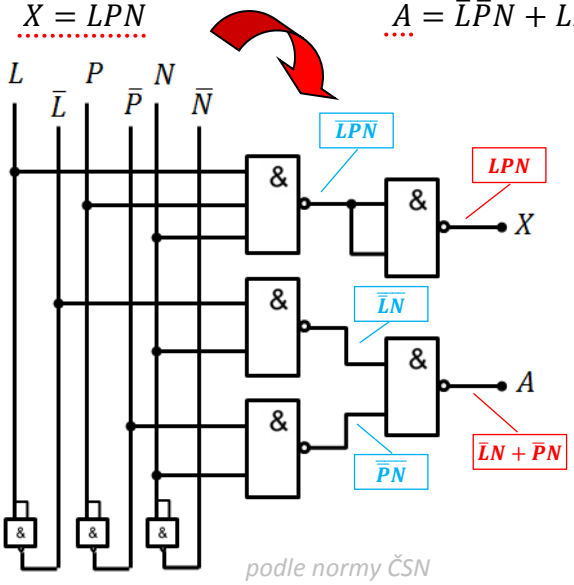




podle normy ANSI

Realizace pomocí prvků NAND:

$X = LPN$ $\qquad A = \bar{L}\bar{P}N + L\bar{P}N + \bar{L}PN = \bar{L}N + \bar{P}N$



podle normy ČSN

Zbývá jen propojit příslušná hradla, navrhnout desku tištěných spojů a osadit. Logické signály jsou samozřejmě jen napěťové nízkého výkonu. Pro ovládání výkonného motoru hydraulického čerpadla i signálního zařízení je nutné obvod doplnit ještě o výkonovou část.

Široká nabídka prvků TTL logiky nám umožňuje realizovat obvod i pomocí jiných typů hradel.

Závěrečná poznámka:

Velmi rychlý vývoj informačních technologií zatlačuje v současné době klasický návrh logických obvodů do pozadí. Zejména v průmyslovém prostředí jsou dnes již velmi často logické funkce realizované pomocí softwarových prostředků prostřednictvím průmyslových počítačů, PLC systémů a programovatelných logických polí. Minimalizace vyjádření logických funkcí se v těchto případech stává bezvýznamnou. Na složitosti logických výrazů v těchto případech nezáleží, většinou se zadávají pravdivostní tabulkou nebo instrukcemi speciálního programovacího jazyka. Sofistikované metody minimalizace (např. algoritmus Quine-McCluskey, 1956) se tak pomalu propadají do zapomnění.

Ukazuje se tak, jak v dynamice vývojové spirály může důsledek zatlačit do pozadí předpoklad, bez kterého by nemohl vůbec vzniknout. Nečekaně se tak v závěru našeho povídání objevil až filozofický aspekt našeho světa.