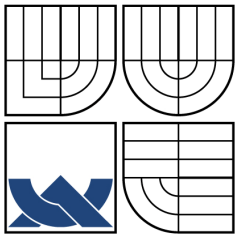
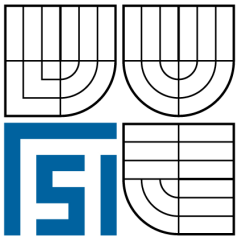


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE INSTITUTE

MINIMALIZACE LOGICKÝCH FUNKCÍ

MINIMISATION OF LOGICAL FUNCTIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — MIROSLAV HORKÝ
AUTHOR

VEDOUCÍ PRÁCE — doc. RNDr. Ing. MILOŠ ŠEDA, Ph.D.
SUPERVISOR

BRNO 2009

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2008/09

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

student(ka): Horký Miroslav

který/která studuje v **bakalářském studijním programu**

obor: **Strojní inženýrství (2301R016)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Minimalizace logických funkcí

v anglickém jazyce:

Minimisation of Logical Functions

Stručná charakteristika problematiky úkolu:

K minimalizaci logických funkcí se používá řada metod, např. uplatnění pravidel Booleovy algebry, Karnaughovy mapy a Quine-McCluskeyho algoritmus, které lze použít k návrhu obvodů s malým počtem součástek.

Cíle bakalářské práce:

Popište jednotlivé metody minimalizace logických funkcí a uveďte jejich výhody a nevýhody.

Seznam odborné literatury:

[1] Balátě, J.: Automatické řízení. BEN – technická literatura, Praha, 2003. ISBN 80-7300-020-0.
[2] Čulík, K., Skalická, M., Váňová, I.: Logika I. VUT FE, Brno, 1968.
[3] Švarc, I., Šeda, M., Vítečková, M.: Automatické řízení. CERM, Brno, 2007. ISBN 978-80-214-3491-2.

Vedoucí bakalářské práce: doc. RNDr. Ing. Miloš Šeda, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2008/09.

V Brně, dne 27. 11. 2008

L.S.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA STROJNÍHO INŽENÝRSTVÍ DĚKANÁT

Šeda

doc. RNDr. Ing. Miloš Šeda, Ph.D.
Ředitel ústavu

doc. RNDr. Miroslav Doupovec, CSc.
Děkan fakulty

Abstrakt

Pro minimalizaci logických funkcí se často využívají Booleova algebra a Karnaughovy mapy. Aplikace Karnaughových map je však založena na vizuálním rozpoznání sousedních buněk pro funkce s max. 6 proměnnými, a proto metoda není vhodná pro automatizované zpracování na počítačích. Přímá aplikace zákonu Booleovy algebry není omezena v tomto směru, ale neexistuje algoritmus, který by definoval posloupnost jejich použití, a tak rovněž není vhodná pro výpočet na počítači. Uvedené nevýhody odstraňuje metoda, kterou navrhli E. J. McCluskey a W. Orman Quine.

Abstract

For minimisation of logical functions, laws of the Boolean algebra and the Karnaugh maps are mostly used. However, use of Karnaugh's maps is based on visual recognition of adjacent cells for functions with no more than 6 variables and, therefore, the method is not suitable for automated processing on computers. A direct application of the Boolean algebra laws is not restricted in this way, but there is no general algorithm defining the sequence of their application and thus this approach is not suitable for computer implementation either. The well-known method usable on computers is the algorithm proposed by E. J. McCluskey and W. Orman Quine.

Klíčová slova:

logická funkce, Karnaughova mapa, Booleova algebra, Quine-McCluskey metoda.

Keywords:

logical function, Karnaugh's map, Boolean algebra, Quine-McCluskey Method.

Poděkování:

Zde bych chtěl poděkovat svému vedoucímu panu doc. RNDr. Miloši Šedovi, Ph.D. za odborné vedení, cenné rady, připomínky a za věnovaný čas při tvorbě bakalářské práce. Také bych chtěl poděkovat mým rodičům, kteří mě podporovali po celou dobu mého studia.

Prohlášení:

Prohlašuji, že jsem bakalářskou práci Minimalizace logických funkcí vypracoval samostatně s použitím literatury a podkladů, uvedených v seznamu použité literatury.

V Brně dne 14. května 2009 ……..

Miroslav Horký

Obsah:

1. Úvod

V dnešní praxi se setkáváme s neustálým zvyšováním nároků na spolehlivost a hlavně na snížení nákladů na výrobu a provoz jak samotného produktu, tak různých pracovních procesů. Těchto vlastností můžeme jednoduše dosáhnout pomocí minimalizace, často označované jako zjednodušování. Jedná se o postup, při kterém se snažíme dosáhnout nalezení nejkratších a nejjednodušších pracovních úkonů k dosažení „cíle“.

Minimalizace logických funkcí je postup, kterým se snažíme o nalezení nejjednoduššího vyjádření logické funkce, čímž rozumíme logickou funkci, která obsahuje nejmenší počet členů a každý člen obsahuje co nejmenší počet proměnných. Z matematického hlediska můžeme zapsat určitou logickou funkci několika možnými způsoby, které se mohou lišit například svým tvarem nebo délkou, ale jejich funkční závislost je vždy totožná, tj. pro stejnou kombinaci vstupů nabývá stejné výstupní hodnoty. Při navrhování nebo konstruování logických obvodů je velmi důležité, aby daná logická funkce byla v co nejjednodušším tvaru, abychom při samotné konstrukci logického obvodu použili co nejmenší počet logických členů, z čehož plyne ekonomičnost a vyšší spolehlivost daného obvodu.

V dalším textu této práce se budeme zabývat nejdůležitějšími metodami minimalizace logických funkcí a uvedeme příklady, na kterých budou vidět výhody a nevýhody studovaných metod.

2. Logické funkce a logické proměnné

Logická proměnná je veličina, která nabývá pouze dvou možných stavů označovaných logická 0 (nepravda, false) a logická 1 (pravda, true) a nemůže se spojitě měnit. V literatuře se můžeme setkat i s názvem dvouhodnotová proměnná nebo dvouhodnotová veličina.

Logická funkce *n* logických proměnných x_1, x_2 ... x_n stejně jako všechny její proměnné může nabývat pouze dvou hodnot logická 0 nebo logická 1. Logickou funkci můžeme také považovat v některých případech za proměnnou, protože může být argumentem jiné logické funkce. Logickou funkci *y* logických proměnných x_1, x_2 ... x_n můžeme vyjádřit vztahem

$$y = f (x_1, x_2 ... x_n), \qquad (2.1)$$

kde *n* může být 1, 2 nebo i číslo větší než 2 a pak mluvíme o logické funkci jedné, dvou či více proměnných.

2.1 Logické funkce jedné proměnné

Logickou funkci *y* jedné proměnné *x* zapisujeme vztahem $y = f(x)$. Existují 4 navzájem různé funkce logické funkce jedné proměnné, které jsou uvedeny v tabulce 2.1 s označením y_0, y_1, y_2 a y_3.

1. nulová funkce (falsum)	2. identická funkce (aserce)	3. negace (NOT)	4. jednotková funkce (verum)
x | y_0 0 | 0 1 | 0	x | y_1 0 | 0 1 | 1	x | y_2 0 | 1 1 | 0	x | y_3 0 | 1 1 | 1
$y_0 = 0$	$y_1 = x$	$y_2 = \overline{x}$	$y_3 = 1$

Tab. 2.1 Logické funkce jedné proměnné – název, pravdivostní tabulka, zápis

První funkce má pro jakékoliv *x* vždy hodnotu y_0 rovnu nule. Hodnota y_1 druhé funkce se vždy shoduje s hodnotou *x*. Třetí funkce negace má vždy opačnou hodnotu y_2 k hodnotě *x*. Nejčastěji používaný zápis negace je $y_2 = \overline{x}$, ale můžeme se setkat i se zápisem y_2 = NOT *x*. Poslední funkce má hodnotu y_3 vždy rovnu 1.

Praktický význam z těchto čtyř funkcí má negace, která je jednou z nejdůležitějších logických funkcí. Podrobněji je popsána v kapitole 3.1.

2.2 Logické funkce dvou proměnných

Pro dvě vstupní proměnné x_1 a x_2 můžeme vyjádřit celkem 16 možných logických funkcí y_0 až y_{15}, viz tabulka 2.2. Těchto 16 funkcí odpovídá všem možným kombinacím vstupních proměnných. Z těchto funkcí jsou velmi důležité konjunkce a disjunkce, které se používají v Booleově algebře a podrobně jsou popsány v kapitole 3.1. Ekvivalence nabývá hodnotu 1, pokud proměnné x_1, x_2 nabývají stejných hodnot, tedy obě hodnotu 0 nebo obě hodnotu 1. Pokud jsou proměnné různé, ekvivalence nabývá hodnotu 0. Implikace $x_1 \rightarrow x_2$ nabývá hodnotu 1, pokud je $x_1 \leq x_2$, předpokládáme přitom, že logická 0 je menší než logická 1, podobně jak v číselné algebře. Implikaci využijeme při objasnění Quine-McCluskeyho algoritmu. Neekvivalence (případně nonekvivalence) nabývá hodnoty 1, pokud proměnné x_1, x_2 nabývají různých hodnot. Nesmíme zapomenout na funkce negace logického součtu a součinu.

1. nulová funkce (falsum)	2. logický součin (konjunkce, AND)	3. přímá inhibice	4. identická funkce (aserce)
x_1 x_2 | y_0 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 0	x_1 x_2 | y_1 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1	x_1 x_2 | y_2 0 0 | 0 0 1 | 0 1 0 | 1 1 1 | 0	x_1 x_2 | y_3 0 0 | 0 0 1 | 0 1 0 | 1 1 1 | 1
$y_0 = 0$	$y_1 = x_1 x_2$	$y_2 = x_1 \bar{x}_2$	$y_3 = x_1$
5. zpětná inhibice	**6. identická funkce (aserce)**	**7. neekvivalence (dilema, XOR)**	**8. logický součet (disjunkce, OR)**
x_1 x_2 | y_4 0 0 | 0 0 1 | 1 1 0 | 0 1 1 | 0	x_1 x_2 | y_5 0 0 | 0 0 1 | 1 1 0 | 0 1 1 | 1	x_1 x_2 | y_6 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0	x_1 x_2 | y_7 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
$y_4 = \bar{x}_1 x_2$	$y_5 = x_2$	$y_6 = \overline{x_1 \equiv x_2}$	$y_7 = x_1 + x_2$
9. negovaný log. součet (Piercova fce, NOR)	**10. ekvivalence (XNOR)**	**11. negace**	**12. zpětná implikace**
x_1 x_2 | y_8 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0	x_1 x_2 | y_9 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1	x_1 x_2 | y_{10} 0 0 | 1 0 1 | 0 1 0 | 1 1 1 | 0	x_1 x_2 | y_{11} 0 0 | 1 0 1 | 0 1 0 | 1 1 1 | 1
$y_8 = \overline{x_1 + x_2}$	$y_9 = x_1 \equiv x_2$	$y_{10} = \bar{x}_2$	$y_{11} = x_1 \leftarrow x_2$
13. negace	**14. přímá implikace**	**15. negovaný log. součin (Shefferova fce, NAND)**	**16. jednotková funkce (verum)**
x_1 x_2 | y_{12} 0 0 | 1 0 1 | 1 1 0 | 0 1 1 | 0	x_1 x_2 | y_{13} 0 0 | 1 0 1 | 1 1 0 | 0 1 1 | 1	x_1 x_2 | y_{14} 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0	x_1 x_2 | y_{15} 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 1
$y_{12} = \bar{x}_1$	$y_{13} = x_1 \rightarrow x_2$	$y_{14} = \overline{x_1 x_2}$	$y_{15} = 1$

Tab. 2.2 Logické funkce dvou proměnných – název, pravdivostní tabulka, zápis

2.3 Logické funkce více než dvou proměnných

Pro *n* proměnných je možno vyjádřit $(2^2)^n$ logických funkcí. Například pro tři proměnné existuje 256 funkcí. Vyjádření takové funkce by bylo dost nepřehledné, proto ji můžeme vyjádřit pomocí známých funkcí dvou proměnných, protože $f_{(x_1,x_2,x_3)} = x_1\left(f_{(1,x_2,x_3)}\right) + \bar{x}_1\left(f_{(0,x_2,x_3)}\right)$. Pro ověření stačí položit $x_1 = 1$ nebo $x_1 = 0$. Z toho je zřejmé, že funkce $f_{(1,x_2,x_3)}$ a $f_{(0,x_2,x_3)}$ jsou funkce dvou proměnných. Stejnou úvahu použijeme i pro více než tři proměnné. Logickou funkci *n* proměnných je možno zapsat pomocí logických funkcí dvou proměnných [5].

3. Booleova algebra

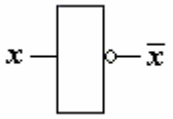
Jedná se o logickou algebru, která je založena na dvouhodnotových veličinách. Jejím zakladatelem je irský matematik G. Boole (1815 – 1864). Základem této algebry jsou veličiny, logické proměnné, které mohou nabývat pouze dvou možných hodnot a to logická 1 a logická 0, viz kapitola 2. Booleova algebra využívá tři základní funkce, kterými jsou negace, konjunkce a disjunkce. Je to algebra vztahů, nikoliv čísel.

3.1 Základní operátory Booleovy algebry

Negace

Tato logická operace, která se také velmi často označuje jako *inverze*. Je aplikována pouze na jednu proměnnou. Jak je z názvu patrné, mění hodnotu proměnné na opačnou. Zapisuje se přidáním „ pruhu “ nad proměnnou *x*, tedy $\bar{x}$. Výsledkem je hodnota $y = 1$, pokud $x = 0$ a naopak $y = 0$, pokud $x = 1$, jak je vidět v pravdivostní tabulce na obr. 3.1. Zápis této funkce $y = \bar{x}$ čteme *y* je negací proměnné *x* nebo *y* se rovná non *x*.

x	*y*
0	1
1	0



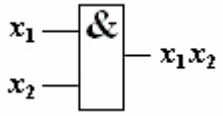
$y = \bar{x}$

Obr. 3.1 Negace – pravdivostní tabulka, schématická značka, zápis

Logický součin

Tato logická operace, která se také velmi často označuje jako *průnik* nebo *konjunkce*, je aplikovaná na dvě proměnné. Jak z názvu vyplývá, vytváří součin nebo také AND těchto dvou proměnných. Logický součin je charakterizován tím, že funkční hodnota *y* nabývá hodnotu 1 jen, když obě nezávislé proměnné x_1, x_2 mají rovněž hodnotu 1. Pokud je alespoň jedna hodnota nulová, tak je také funkční hodnota $y = 0$, jak je vidět z pravdivostní tabulky na obr. 3.2. Konjunkci značíme symbolem $\wedge$ mezi proměnnými. V praxi se spíše setkáváme se zápisem $y = x_1 . x_2$, který lze (podobně jako v aritmetice) dále zjednodušit na zápis $y = x_1 x_2$.

x_1	x_2	*y*
0	0	0
0	1	0
1	0	0
1	1	1



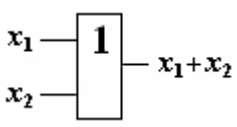
$y = x_1 x_2$

Obr. 3.2 Konjunkce – pravdivostní tabulka, schématická značka, zápis

Logický součet

Tato logická operace, která se také velmi často označuje jako *sjednocení* nebo *disjunkce*, je rovněž aplikovaná na dvě proměnné. Jak je z názvu zřejmé, vytváří součet nebo také OR těchto dvou proměnných. Logický součet je charakterizován tím, že funkční hodnota *y* nabývá hodnotu 1, jestliže alespoň jedna z proměnných x_1, x_2 nabývá hodnoty 1, jak je vidět v pravdivostní tabulce na obr. 3.3. Disjunkci značíme symbolem $\vee$ mezi proměnnými. V praxi se spíše setkáváme se zápisem $y = x_1 + x_2$.

x_1	x_2	*y*
0	0	0
0	1	1
1	0	1
1	1	1



$y = x_1 + x_2$

Obr. 3.3 Disjunkce – pravdivostní tabulka, schématická značka, zápis

3.2 Zákony Booleovy algebry [3]

Zákony Booleovy algebry nám slouží při práci s algebraickými výrazy, jejich úpravě, zjednodušování a rovněž pro minimalizaci logických funkcí, kterou se budeme zabývat podrobně v kapitole 4.1.

Zákon vyloučeného třetího

$$x + \bar{x} = 1 \tag{3.1}$$

Zákon logického rozporu

$$x \times \bar{x} = 0 \tag{3.2}$$

Zákon dvojité negace

$$\bar{\bar{x}} = x \tag{3.3}$$

Zákony opakování

$$x + x = x \qquad x \times x = x \tag{3.4}$$

Zákony komutativní

$$x_1 + x_2 = x_2 + x_1 \qquad x_1 \times x_2 = x_2 \times x_1 \tag{3.5}$$

Zákony asociativní

$$x_1 + (x_2 + x_3) = x_1 + x_2 + x_3 \qquad x_1(x_2 x_3) = x_1 x_2 x_3 \tag{3.6}$$

Zákony distributivní

$$x_1(x_2 + x_3) = x_1 x_2 + x_1 x_3 \qquad x_1 + (x_2 x_3) = (x_1 + x_2)(x_1 + x_3) \tag{3.7}$$

Zákony absorpční

$$\begin{aligned} x_1 + x_1 \times x_2 &= x_1 & \qquad x_1 \times (x_1 + x_2) &= x_1 \\ x_1 + \bar{x}_1 \times x_2 &= x_1 + x_2 & \qquad x_1 \times (\bar{x}_1 + x_2) &= x_1 \times x_2 \end{aligned} \tag{3.8}$$

Zákony neutrálnosti 0 *a* 1

$$0 + x = x \qquad x \times 1 = x \tag{3.9}$$

Zákony agresivnosti 0 *a* 1

$$1 + x = 1 \qquad x \times 0 = 0 \tag{3.10}$$

Zákony de Morganovy

$$\overline{x_1 + x_2} = \bar{x}_1 \times \bar{x}_2 \qquad \overline{x_1 \times x_2} = \bar{x}_1 + \bar{x}_2 \tag{3.11}$$

U většiny zákonů je platnost zřejmá, např. u zákonu dvojité negace (3.3) je zřejmé, že pokud proměnnou *x* znegujeme dvakrát, dostaneme tutéž hodnotu původní proměnné *x*. Platnost některých zákonů není zcela jasná, proto si pro ně můžeme sestavit pravdivostní tabulku se všemi možnými kombinacemi hodnot proměnných a vyhodnotíme si v ni výrazy na pravé a levé straně zkoumaného zákona. Pokud jsou ve všech řádcích pravdivostní tabulky hodnoty stejné, je tímto platnost ověřena. Pro názornost je to vidět v tabulce 3.1.

x_1	x_2	$x_1 + \bar{x}_1 \times x_2$	$x_1 + x_2$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

Tab. 3.1 Ověření 3. absorpčního zákonu

3.3 Způsoby vyjádření Booleových funkcí

Nejčastěji používané způsoby vyjádření Booleových funkcí jsou pravdivostní tabulka, algebraický výraz, Karnaughova mapa a blokové schéma.

Pravdivostní tabulka

Pravdivostní tabulka je tabulka, do které zapisujeme logickou (Booleovu) funkci. Tuto tabulku tvoří $r + n$ sloupců a 2^n řádků. Číslo n odpovídá počtu nezávisle proměnných x_1, x_2 ... x_n, pro které je funkce definována a číslo r odpovídá počtu sloupců výsledných funkcí (obyčejně bývá pouze jedna funkce – závislá proměnná y). Číslo 2^n udává počet všech možných kombinací nezávisle proměnných, kde číslo n je počet nezávisle proměnných. Pořadí kombinací nezávisle proměnných se píše v pravdivostní tabulce po sobě v binární (dvojkové) soustavě. Pokud je tedy zadána logická funkce, která má tři nezávisle proměnné a jednu výslednou funkci, pak bude mít pravdivostní tabulka čtyři sloupce ($r + n = 1 + 3 = 4$) a osm řádků ($2^n + 2^3 = 8$) [5].

x	y
0	1
1	0

x_1	x_2	y
0	0	1
0	1	0
1	0	1
1	1	1

x_1	x_2	x_3	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Tab. 3.2 Pravdivostní tabulka jedné,dvou a tří nezávisle proměnných

Pravdivostní tabulka 3.3 zobrazuje funkční hodnotu y (světlo – svítí, nesvítí), která je závislá na třech nezávislých proměnných x_1, x_2 a x_3 (vypínač – zapnuto, vypnuto). S touto tabulkou budeme pracovat v dalších možnostech vyjádření logické funkce.

x_1	x_2	x_3	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Tab. 3.3 Pravdivostní tabulka

Algebraický výraz

Logickou funkci můžeme zapsat ve dvou možných tvarech, které nazýváme *základní součtový* a *základní součinový tvar*. V literatuře se často používají názvy *úplná normální disjunktivní forma* – ÚNDF a *úplná normální konjunktivní forma* – ÚNKF.

ÚNDF je to součet součinů, které obsahují všechny proměnné v přímém nebo negovaném tvaru. Funkce nabývá hodnotu 1, jestliže některý ze součinů, který ji tvoří, nabývá hodnotu 1. ÚNDF tedy vyjadřuje funkci jako součet případů, kdy má hodnotu 1 [5].

ÚNKF je to součin součtů, které obsahují všechny proměnné v přímém nebo negovaném tvaru. Funkce má hodnotu 1, jestliže všechny dílčí součty mají hodnotu jedna. ÚNKF vyjadřuje funkci jako součin případů, kdy má hodnotu 0 [5].

Pokud tedy přecházíme od pravdivostní tabulky k algebraickému výrazu musíme vědět v jakém tvaru chceme mít funkci zapsanou. Pokud ji chceme v součtovém tvaru (ÚNDF), postupujeme v pravdivostní tabulce po řádcích a v případech, kdy funkční hodnota nabývá hodnoty 1, opíšeme odpovídající konjunkci podmínek. Tedy pro třetí řádek tabulky 3.3 odpovídá $x_1 = 0$, $x_2 = 1$, $x_3 = 0$. Když je hodnota proměnné rovna 0, můžeme ji ekvivalentně vyjádřit její negací. Když je proměnná rovna 1, zapíšeme ji přímo. Třetímu řádku tabulky 3.3 tedy odpovídá logický výraz $\bar{x}_1 x_2 \bar{x}_3$. Z tohoto vyplývá, že každému řádku, kde nabývá funkční hodnota y hodnotu 1, odpovídá právě jeden logický součin (konjunkce) vstupních proměnných vytvořený výše uvedeným způsobem. Výsledná logická funkce je tvořena součtem všech těchto nalezených výrazů. Aplikujeme-li tento postup na pravdivostní tabulku 3.3, dostáváme algebraické vyjádření ve tvaru:

$$y = \bar{x}_1 x_2 \bar{x}_3 + \bar{x}_1 x_2 x_3 + x_1 \bar{x}_2 x_3 + x_1 x_2 x_3$$

Naopak, chceme-li vyjádřit logickou funkci v součinovém tvaru (ÚNKF), postupujeme v pravdivostní tabulce opět po řádcích a v případech, kdy funkční hodnota nabývá hodnoty 0, opíšeme odpovídající disjunkci podmínek. Prvnímu řádku tabulky 3.3 odpovídá $x_1 = 0$, $x_2 = 0$, $x_3 = 0$, kde jsou všechny proměnné rovny 0, zapíšeme je přímo x_1, x_2 a x_3. Když je proměnná rovna 1, vyjádříme ji ekvivalentně její negací. Pro první řádek tedy platí logický výraz $x_1 + x_2 + x_3$. Výsledná logická funkce je tvořena součinem všech těchto nalezených výrazů. Aplikujeme-li tento postup na pravdivostní tabulku 3.3, dostáváme algebraické vyjádření ve tvaru:

$$y = (x_1 + x_2 + x_3)(x_1 + x_2 + \bar{x}_3)(\bar{x}_1 + x_2 + x_3)(\bar{x}_1 + \bar{x}_2 + x_3)$$

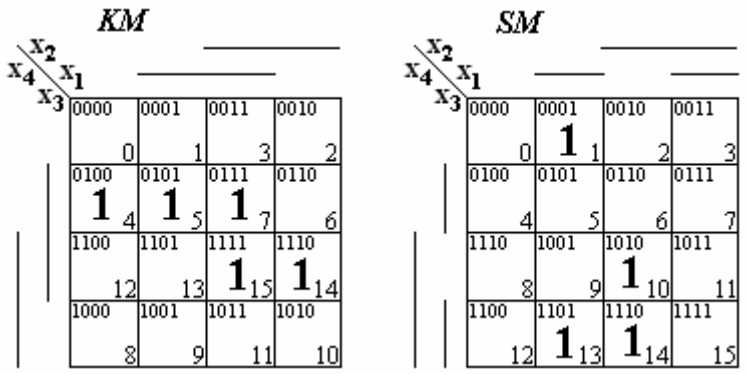
V praxi se používá především součtový tvar, tedy ÚNDF.

Karnaughova mapa

Zobrazení pomocí map slouží k vyjádření logických funkcí, a především k jejich minimalizaci, kterou se budeme zabývat podrobně v kapitole 4.2. Použití těchto map je vhodné, pokud počet nezávislých proměnných nepřekračuje počet šest.

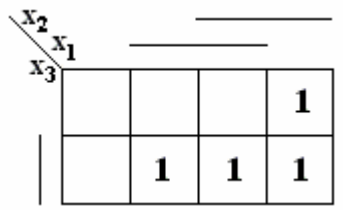
Mapa obsahuje tolik políček, kolik je všech možných kombinací nezávisle proměnných. To odpovídá 2^n polím, kde číslo n odpovídá počtu nezávisle proměnných x_1, x_2 ... x_n. Je to v podstatě přetransformovaná pravdivostní tabulka, kde každému řádku odpovídá jedno pole mapy. V řádku nebo ve sloupci, ve kterém je příslušná proměnná rovna hodnotě 1, označujeme to vedle mapy svislou nebo vodorovnou čarou, které připíšeme příslušnou proměnnou. Kde nabývá v pravdivostní tabulce funkční hodnota hodnotu 1, zapíšeme do příslušného pole tabulky 1, pokud je funkční hodnota rovna 0, necháváme příslušné pole prázdné. Zapisování funkce do mapy je jednoduché, v podstatě spočívá v přepsání funkčních hodnot do příslušných polí. Při zápisu do mapy můžeme vycházet jak z pravdivostní tabulky, tak z algebraického výrazu, který musíme mít v součtovém tvaru, tedy ÚNDF.

Podle způsobu „ kódování “ řádků a sloupců rozlišujeme dvě mapy Karnaughovu nebo Svobodovu mapu. Karnaughova mapa využívá ke kódování řádků a sloupců Greyův kód, tudíž sousední políčka se od sebe liší pouze v jedné hodnotě. Tato mapa je výhodnější a v praxi se používá především při minimalizaci. Svobodova mapa využívá ke kódování řádků a sloupců binární kód, stavové indexy rostou zleva doprava po sloupcích a shora dolů po řádcích. Rozdíl těchto dvou map je patrný na následujícím obrázku 3.4.



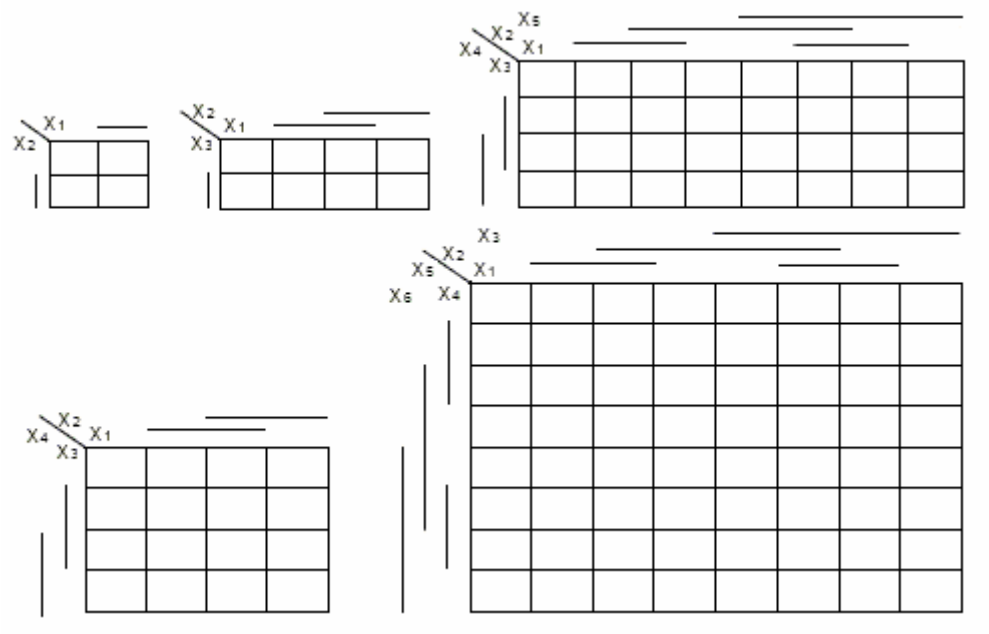
Obr. 3.4 Karnaughova mapa, Svobodova mapa

Podle postupu, pro vyplňování mapy popsaného výše si vytvoříme Karnaughovu mapu pro pravdivostní tabulku 3.3, která je zobrazena na obrázku 3.5.



Obr. 3.5 Karnaughova mapa pravdivostní tabulky 3.3

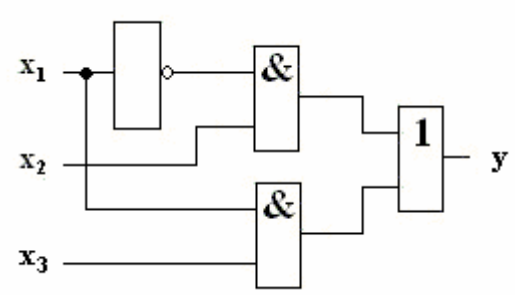
Pro názornost si uvedeme Karnaughovu mapu všech možných variant, obrázek 3.6.



Obr. 3.6 Karnaughova mapa 2 až 6 proměnných [3]

Blokové schéma

Blokové schéma tvoříme pomocí základních operátorů Booleovy algebry a to negace, logického součinu a logického součtu. Jejich schématické značky jsou uvedeny v kapitole 3.1. Toto schéma se kreslí především až po provedené minimalizaci, abychom použili co nejmenší počet logických členů. Blokové schéma pro pravdivostní tabulku 3.3, tedy pro algebraický výraz $y = \bar{x}_1 x_2 \bar{x}_3 + \bar{x}_1 x_2 x_3 + x_1 \bar{x}_2 x_3 + x_1 x_2 x_3$, by bylo příliš složité, proto logickou funkci nejdříve minimalizujeme. Po provedené minimalizaci v našem příkladu dostaneme logickou funkci $y = \bar{x}_1 x_2 + x_1 x_3$. Této logické funkci odpovídá blokové schéma na obrázku 3.5.



Obr. 3.7 Blokové schéma pro pravdivostní tabulku 3.3

4. Minimalizace logických funkcí

Minimalizace, zjednodušování je postup, kterým se snažíme dosáhnout nejjednodušší vyjádření logické funkce, což je funkce s minimálním počtem členů a každý člen obsahuje co nejmenší počet proměnných.

Pro minimalizaci logických funkcí slouží různé metody, kterými jsou algebraická minimalizace, využití Karnaughovy mapy a Quine-McCluskeyho algoritmus.

4.1 Algebraická minimalizace

Algebraická minimalizace logických funkcí je zjednodušování logického výrazu, při které využíváme všech pravidel a zákonů Booleovy algebry (kapitola 3.2). Pravidla a zákony aplikujeme na danou logickou funkci tak dlouho, dokud nedostaneme nejkratší výraz.

Při algebraické minimalizaci nejvíce využíváme zákonu vyloučeného třetího (3.1), zákonu logického rozporu (3.2), zákonu dvojité negace (3.3) a zákonu opakování (3.4). Důležité jsou také de Morganovy zákony (3.11), které se požívají velmi často, pokud se ve výrazu nachází negace přes více proměnných. Nesmíme zapomenout na ostatní zákony, které jsou rovněž velmi důležité. Pro představu, jak se tato minimalizace provádí, si uvedeme dva příklady.

Příklad 4.1 : Minimalizujte logickou funkci $y = \bar{x}_1\bar{x}_2x_3 + \overline{x_1x_2}x_3 + x_1\bar{x}_2x_3$

Řešení:

$$y = \bar{x}_2x_3\left(\bar{x}_1 + x_1\right) + \left(\bar{x}_1 + \bar{x}_2\right)x_3 = \bar{x}_2x_3 + \bar{x}_1x_3 + \bar{x}_2x_3 = \bar{x}_1x_3 + \bar{x}_2x_3 = x_3\left(\bar{x}_1 + \bar{x}_2\right) = \underline{\underline{\overline{x_1x_2}x_3}}$$

V prvním kroku vytkneme z 1. a 3. členu $\bar{x}_2x_3$ a 2. člen upravíme podle de Morganova zákona (3.11). Výraz v první závorce je podle zákona vyloučeného třetího (3.1) roven jedné. Druhou závorku roznásobíme. V druhém kroku využijeme zákonu opakování (3.4). V posledním kroku opět využijeme de Morganův zákon (3.11) a dostáváme konečný tvar minimalizované funkce.

Příklad 4.2 : Minimalizujte logickou funkci $y = \bar{x}_1x_2x_3 + x_1\bar{x}_2\bar{x}_3 + x_1\bar{x}_2x_3 + x_1x_2\bar{x}_3 + x_1x_2x_3$ [3]

Řešení:

$$y = \bar{x}_1x_2x_3 + x_1\bar{x}_2\left(\bar{x}_3 + x_3\right) + x_1x_2\left(\bar{x}_3 + x_3\right) = \bar{x}_1x_2x_3 + x_1\bar{x}_2 + x_1x_2 = \bar{x}_1x_2x_3 + x_1\left(\bar{x}_2 + x_2\right) =$$

$$\bar{x}_1x_2x_3 + x_1 = \underline{\underline{x_1 + x_2x_3}}$$

V prvním kroku vytkneme z 2. a 3. členu $x_1\bar{x}_2$ a z 4. a 5. členu x_1x_2. Výraz v první a druhé závorce je podle zákona vyloučeného třetího (3.1) roven jedné. V druhém kroku vytkneme z 2. a 3. členu x_1 a výraz ve vzniklé závorce je opět podle zákona vyloučeného třetího (3.1) roven jedné. V posledním kroku využijeme absorpční zákon (3.8) a dostáváme konečný tvar minimalizované funkce.

4.2 Minimalizace pomocí Karnaughovy mapy

Minimalizace logických funkcí pomocí Karnaughovy mapy (v literatuře se můžeme setkat i s označením grafická minimalizace) je umožněna tím, že sousední políčka se od sebe liší pouze v jedné hodnotě. Minimální logickou funkci určíme tak, že v Karnaughově mapě vytváříme tzv. podmapy [5]. Podmapou rozumíme sjednocení sousedních políček do dvojic, čtveřic, osmic, šestnáctic atd., ve kterých nabývá logická funkce funkční hodnotu 1. Při sloučení dvou sousedních políček vypadne jedna proměnná, pokud sloučíme čtyři sousední políčka, vypadnou dvě proměnné atd. V odpovídající logické funkci chybí vždy ta proměnná, která

v dané dvojici, čtveřici atd. mění svojí hodnotu. Snažíme se vytvářet co největší podmapy, aby vypadlo co nejvíce proměnných. Výběr podmap provádíme podle těchto šesti pravidel:

- vybranými podmapami musí být zakroužkovány všechny jednotkové stavy logické funkce, nesmí být žádna vynechána
- dvojice, čtveřice atd. vytváříme jak svisle, tak vodorovně. Také v podmapách spojujeme sousední pole, které spolu sousedí i přes okraje mapy. Rohy mapy jsou také sousední pole
- podmapy se mohou prolínat, tedy jedna jednička může být současně součástí dvojice, čtveřice atd.
- podmapy pravidelného tvaru (čtverec, obdélník) vytváříme co největší, tedy přednost mají osmice před čtveřicemi, čtveřice před dvojicemi atd., aby se ze stavů vyloučilo co nejvíce proměnných
- nevytváříme zbytečné podmapy, tedy nespojujeme ty stavy, které už máme pokryty jinou mapou
- podle pravidla, že nesmíme žádnou jedničku vynechat, se snažíme, aby počet podmap byl co nejmenší

Minimalizace pomocí Karnaughovy mapy spočívá v tom, že pro danou Karnaughovu mapu, pravdivostní tabulku hledáme algebraický výraz, který je již minimalizovaný. Logická funkce, kterou chceme minimalizovat, může být zadaná i jako algebraický výraz. Pro představu, jak se tato minimalizace provádí, si uvedeme dva příklady.

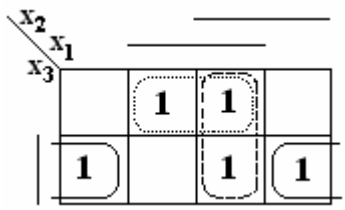
Příklad 4.3 : Minimalizujte logickou funkci, která je dána pravdivostní tabulkou 4.1

x_1	x_2	x_3	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Tab. 4.1 Pravdivostní tabulka příkladu 4.3

Řešení:

Nakreslíme Karnaughovu mapu pro tři proměnné, zapíšeme jedničky do příslušných políček a zakroužkujeme tři dvojice obr. 4.1. Těmto třem dvojicím tedy odpovídají tři výrazy. První dvojice (tečkovaná čára) odpovídá výrazu $x_1\bar{x}_3$, druhá dvojice (čárkovaná čára) odpovídá x_1x_2 a třetí dvojice (plná čára) odpovídá výrazu $\bar{x}_1x_3$. Konečný výsledek je logický součet těchto výrazů tedy $y = \underline{\underline{x_1\bar{x}_3 + x_1x_2 + \bar{x}_1x_3}}$.

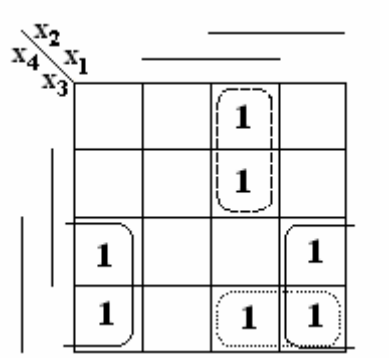


Obr. 4.1 Karnaughova mapa tří proměnných příkladu 4.3

Příklad 4.4 : Minimalizujte logickou funkci, která je zadána algebraickým výrazem
$y = \overline{x}_1\overline{x}_2\overline{x}_3x_4 + \overline{x}_1\overline{x}_2x_3x_4 + \overline{x}_1x_2\overline{x}_3x_4 + \overline{x}_1x_2x_3x_4 + x_1x_2\overline{x}_3\overline{x}_4 + x_1x_2\overline{x}_3x_4 + x_1x_2x_3\overline{x}_4$

Řešení:

Nakreslíme Karnaughovu mapu pro čtyři proměnné, zapíšeme jedničky do příslušných políček a zakroužkujeme čtveřici a dvě dvojice obr. 4.2. První dvojice (čárkovaná čára) odpovídá výrazu $x_1x_2\overline{x}_4$, druhá dvojice (tečkovaná čára) odpovídá výrazu $x_2\overline{x}_3x_4$ a čtveřice odpovídá výrazu $\overline{x}_1x_4$. Výsledek je logický součet těchto výrazů $y = \underline{\underline{x_1x_2\overline{x}_4 + x_2\overline{x}_3x_4 + \overline{x}_1x_4}}$.



Obr. 4.2 Karnaughova mapa čtyř proměnných příkladu 4.4

4.3 Quine-McCluskeyho algoritmus

Tuto metodu minimalizace logických funkcí navrhli filozof W. Orman Quine a profesor elektrotechniky a informatiky E. J. McCluskey. Podstatou této metody je využití Booleových zákonů a to zákona vyloučeného třetího (3.1), zákona opakování (3.4) a distributivního zákona (3.7). Pokud chceme využít právě zákona vyloučeného třetího a distributivního zákona, musí být daná funkce ve tvaru ÚNDF (součtový tvar) a dva výrazy z této funkce se musí lišit pouze v jedné proměnné. To znamená, že v jednom výrazu je daná proměnná negovaná a v druhém bez negace. Pokud toto platí, je možno oba výrazy zjednodušit na část, která je oběma společná. Obecně je možno na takovéto zjednodušení narazit tehdy, když počet negací v obou výrazech se liší o jednu, z čehož vyplývá základní postup Quine-McCluskeyho algoritmu:

- výrazy minimalizované logické funkce se rozdělí do skupin tak, že v jedné skupině se nachází pouze ty členy, ve kterých se nachází stejný počet proměnných s negací
- vyšetříme všechny možné dvojice, kdy jeden výraz patří do jedné skupiny a druhý je vždy se skupiny s počtem negací o jednu vyšší
- při odlišnosti výrazů pouze v jedné proměnné se využije zákonu vyloučeného třetího nebo distributivního zákonu a výrazy zjednodušíme, porovnávané výrazy vyloučíme (označíme je zatržením). Do další iterace se přenesou výrazy již zjednodušené na shodné proměnné
- opakující se výrazy v další iteraci podle zákonu opakování vypustíme
- pokud výrazy v nové iteraci je možno dále zjednodušovat, pokračujeme jako v předchozích krocích, pokud to již není možné, algoritmus končí
- výsledkem tohoto algoritmu jsou ty výrazy, které zůstaly nezaškrtnuty, tedy ty, které nebylo možno dále zjednodušit

Pro dobré pochopení tohoto algoritmu si uvedeme jednoduchý příklad, na kterém si ukážeme, jak postupujeme v tomto algoritmu krok po kroku.

Příklad 4.5: Minimalizujte logickou funkci [2],[3]

$$y = \bar{x}_1 x_2 x_3 x_4 + x_1 \bar{x}_2 x_3 x_4 + x_1 x_2 \bar{x}_3 x_4 + \bar{x}_1 x_2 x_3 \bar{x}_4 + x_1 \bar{x}_2 x_3 \bar{x}_4$$
$$+ x_1 x_2 \bar{x}_3 \bar{x}_4 + \bar{x}_1 \bar{x}_2 x_3 \bar{x}_4 + x_1 \bar{x}_2 \bar{x}_3 x_4 + \bar{x}_1 \bar{x}_2 x_3 \bar{x}_4$$

Řešení:

V prvním kroku rozdělíme výrazy do skupin podle počtu negací, dostáváme tedy výchozí iteraci.

(0) $\bar{x}_1 x_2 x_3 x_4 ; x_1 \bar{x}_2 x_3 x_4 ; x_1 x_2 \bar{x}_3 x_4$
$\bar{x}_1 x_2 x_3 \bar{x}_4 ; x_1 \bar{x}_2 x_3 \bar{x}_4 ; x_1 x_2 \bar{x}_3 \bar{x}_4 ; \bar{x}_1 \bar{x}_2 x_3 \bar{x}_4 ; x_1 \bar{x}_2 \bar{x}_3 x_4$
$\bar{x}_1 \bar{x}_2 x_3 \bar{x}_4$

V tomto kroku budeme srovnávat všechny dvojice ze sousedních skupin, tedy 5 x 3 + 5 x 1 dvojic. Zakroužkujeme (umístíme do svislých závorek) všechny výrazy výchozí iterace (0), které je možno zjednodušit viz (0*), výsledné zjednodušené výrazy zapíšeme v nové iteraci (1), kde je opět rozdělíme do skupin podle počtu negací.

(0*) $\left|\bar{x}_1 x_2 x_3 x_4\right| ; \left|x_1 \bar{x}_2 x_3 x_4\right| ; \left|x_1 x_2 \bar{x}_3 x_4\right|$
$\left|\bar{x}_1 x_2 x_3 \bar{x}_4\right| ; \left|x_1 \bar{x}_2 x_3 \bar{x}_4\right| ; \left|x_1 x_2 \bar{x}_3 \bar{x}_4\right| ; \left|\bar{x}_1 \bar{x}_2 x_3 \bar{x}_4\right| ; \left|x_1 \bar{x}_2 \bar{x}_3 x_4\right|$
$\left|\bar{x}_1 \bar{x}_2 x_3 \bar{x}_4\right|$

(1) $\bar{x}_1 x_2 x_3 ; \bar{x}_1 x_3 x_4 ; x_1 \bar{x}_2 x_3 ; \bar{x}_2 x_3 x_4 ; x_1 \bar{x}_2 x_4 ; x_1 x_2 \bar{x}_3 ; x_1 \bar{x}_3 x_4$
$\bar{x}_1 x_3 \bar{x}_4 ; \bar{x}_2 x_3 \bar{x}_4 ; \bar{x}_1 \bar{x}_2 x_3$

Předchozí kroky opět aplikujeme na iteraci (1), budeme tedy vyšetřovat 3 x 7 dvojic. Z této iterace vypadnou zakroužkované výrazy v (1*) a ty, které bylo možno zjednodušit jsou obsaženy v iteraci (2).

(1*) $\left|\bar{x}_1 x_2 x_3\right| ; \left|\bar{x}_1 x_3 x_4\right| ; \left|x_1 \bar{x}_2 x_3\right| ; \left|\bar{x}_2 x_3 x_4\right| ; x_1 \bar{x}_2 x_4 ; x_1 x_2 \bar{x}_3 ; x_1 \bar{x}_3 x_4$
$\left|\bar{x}_1 x_3 \bar{x}_4\right| ; \left|\bar{x}_2 x_3 \bar{x}_4\right| ; \left|\bar{x}_1 \bar{x}_2 x_3\right|$

(2) $\bar{x}_1 x_3 ; \bar{x}_2 x_3$

Protože máme v iteraci (2) pouze jednu skupinu výrazů a nelze je tedy dále zjednodušovat, algoritmus končí. Výsledkem je logický součet výrazů, které nebylo možno dále zjednodušovat tedy: $y = \bar{x}_1 x_3 + \bar{x}_2 x_3 + x_1 \bar{x}_2 x_4 + x_1 x_2 \bar{x}_3 + x_1 \bar{x}_3 x_4$

Pokud bychom logickou funkci z příkladu 4.5 minimalizovali pomocí Karnaughovy mapy, získali bychom výslednou funkci, která by obsahovala menší počet výrazů než výsledek, který jsme dostali pomocí Quine-McCluskeyho algoritmu. Proto byla navržena S. R. Petrickem modifikace tohoto algoritmu, která odstraní redukci ve výsledku [3]. Než budeme pokračovat, musíme si ujasnit, co jsme dostali za funkci pomocí Quine-McCluskeyho algoritmu.

Výrazy v této funkci se nazývají minimální implikanty (*prime implicants*). Výraz g je implikantou výrazu f, pokud je každá proměnná výrazu g proměnnou výrazu f, a jestliže funkce implikace $g \rightarrow f$ nabývá všude hodnoty 1. Implikace nabývá hodnoty 1 pouze tehdy, když $g \leq f$ (přitom musíme předpokládat, že logická 0 je menší než logická 1 podobně jak v číselné algebře). Z této definice plyne, že g je implikantou f jen tehdy, když platí $g \leq f$. Minimální

implikantou logické funkce, jsou výrazy, které splňují podmínku, že pokud vypustíme kterýkoliv výskyt kterékoliv proměnné negované nebo bez negace, dostaneme součin, který není implikantou daného výrazu. Jinak řečeno, minimální implikanta se nedá již dále zjednodušovat. Funkce, kterou jsme dostali, se skládá z minimálních implikant, ale nevíme, zda se jedná o minimální disjunktivní formu. Minimální implikanty známe a zbývá tedy druhá část, pomocí které najdeme minimální disjunktivní formu.

K tomu nám slouží Petrickova modifikace. Z výsledku, který jsme dostali pomocí Quine-McCluskeyho algoritmu, sestavíme tzv. tabulku minimálních implikant, která má tolik sloupců, kolik je minimálních implikant a v záhlaví každého sloupce je jediná minimální implikanta. Tabulka obsahuje tolik řádků, kolik je výrazů v ÚNDF dané funkce a v záhlaví každého řádku je jediný výraz. V průsečíku řádku se sloupci vložíme symbol (*), jestliže minimální implikanta ve sloupci je součástí výrazu v řádku. Tabulku minimálních implikant pro příklad 4.5 je zobrazena v tabulce 4.2.

	$\bar{x}_1 x_3$	$\bar{x}_2 x_3$	$x_1 \bar{x}_2 x_4$	$x_1 x_2 \bar{x}_3$	$x_1 \bar{x}_3 x_4$
$\bar{x}_1 x_2 x_3 x_4$	*				
$x_1 \bar{x}_2 x_3 x_4$		*	*		
$x_1 x_2 \bar{x}_3 x_4$				*	*
$\bar{x}_1 x_2 x_3 \bar{x}_4$	*				
$x_1 \bar{x}_2 x_3 \bar{x}_4$		*			
$x_1 x_2 \bar{x}_3 \bar{x}_4$				*	
$\bar{x}_1 \bar{x}_2 x_3 \bar{x}_4$	*	*			
$x_1 \bar{x}_2 \bar{x}_3 x_4$			*		*
$\bar{x}_1 \bar{x}_2 x_3 \bar{x}_4$	*	*			

Tab. 4.2 Tabulka minimálních implikant

Minimální implikanty nemůžeme dále zjednodušit, ale některé mohou být redundantní a z tabulky je můžeme vypustit. Musí být splněno to, že znak * ve sloupcích musí být rozložen do všech řádků. Pokud by alespoň jeden řádek neobsahoval žádný znak *, pak by součet uvažovaných implikant neodpovídal dané funkci. Pokud se v daném řádku nachází pouze jedna *, v odpovídajícím sloupci nemůže být minimální implikanta vypuštěna. Tyto implikanty, které nemůžeme vypustit, označujeme jako nezbytné implikanty (*essential prime implicants*) a jsou vždy obsaženy v minimální disjunktivní formě. Z tabulky 4.2 je vidět, že nezbytné implikanty jsou $\bar{x}_1 x_3$, $\bar{x}_2 x_3$ a $x_1 x_2 \bar{x}_3$. Tabulku 4.2 můžeme zjednodušit tak, že vypustíme sloupce nezbytných implikant a řádky, které obsahují jejich hvězdičky, tímto vznikne nová tabulka 4.3.

	$x_1 \bar{x}_2 x_4$	$x_1 \bar{x}_3 x_4$
$x_1 \bar{x}_2 \bar{x}_3 x_4$	*	*

Tab. 4.3 Zjednodušená tabulka

Z tabulky 4.3 je zřejmé, že je třeba přibrat buď jednu nebo druhou implikantu k nezbytným implikantám a tímto dostaneme dvě minimální disjunktivní formy:

$$y_{1min} = \bar{x}_1 x_3 + \bar{x}_2 x_3 + x_1 \bar{x}_2 x_4 + x_1 x_2 \bar{x}_3$$
$$y_{2min} = \bar{x}_1 x_3 + \bar{x}_2 x_3 + x_1 \bar{x}_3 x_4 + x_1 x_2 \bar{x}_3$$

I když provedeme Petrickovu modifikaci Quine-McCluskeyho algoritmu, pořád tento postup výpočtu má svá omezení. Problém je pokrytí výchozí ÚNDF minimálními implikantami [3]. Pokud je v tabulce počet nezbytných implikantů velký, je patrné, že po vyškrtnutí odpovídajících sloupců a řádků se nám tabulka výrazně zmenší. Obecně může nastat situace, kdy žádný z výrazů není nezbytný.

Problém výběru co nejmenšího počtu minimálních implikant, které musíme k nezbytným implikantům přidat tak, aby všechny výrazy zadané ÚNDF byly pokryty, lze popsat problémem pokrytí [3]:

Nechť A je matice o *m* řádcích a *n* sloupcích odpovídající zjednodušené tabulce. Hodnoty prvku a_{ij} matice A jsou 1 nebo 0, pokud na pozici *i*-tého řádku a *j*-tého sloupce je znak * nebo ne. Označme symbolem v_j dvouhodnotovou proměnnou, která nabývá hodnoty 1, jestliže minimální implikant v j-tém sloupci zjednodušené tabulky je vybrán do řešení, v opačném případě nabývá hodnoty 0.

Cílem je minimalizovat výraz : $\sum_{j=1}^{n} v_j$ (4.1)

Za omezujících podmínek:

$$\sum_{j=1}^{n} a_{ij} v_j \geq 1, i = 1,2,...,m$$
$$v_j \in \{0,1\}, j = 1,2,...,n \qquad (4.2)$$

Omezující podmínky zaručují pokrytí řádku alespoň jedním sloupcem. Uvádí se, že pro ÚNDF s *n* výrazy je počet minimálních implikant $3^n/n$. Pro $n = 32$ můžeme dostat $6{,}5 \times 10^{15}$ minimálních implikant. Po redukci nezbytných implikant zůstane *k* minimálních implikant, pro které existuje 2^k-1 různých způsobů výběrů minimálních implikant pro pokrytí řádku. Některé z nich nemá smysl uvažovat, protože nepokrývají všechny řádky. Problém pokrytí patří mezi tzv. NP-těžké optimalizační problémy, jejichž časová složitost exponenciálně roste se zvyšujícím se počtem vstupů [3].

5. Porovnání minimalizačních metod, jejich výhody a nevýhody

Minimalizaci logických funkcí provádíme pomocí metod, které jsou popsány v kapitole 4. Metody se od sebe liší způsobem dosažení zjednodušené logické funkce. Každá z těchto metod má bezesporu své výhody, ale také i nevýhody.

Algebraická minimalizace využívající zákonů Booleovy algebry, které aplikujeme tak dlouho, dokud nedostaneme minimální tvar funkce, není metodou omezenou v počtu proměnných, ale neexistuje žádný postup, který by nám definoval, v jakém pořadí máme zákony použít, proto není možné sestrojení výpočtu na počítači. Pro výraz obsahující max. 3 proměnné je metoda velmi rychlá. S narůstajícím počtem proměnných je úprava značně pracná, nepřehledná, obtížná a nejsme si jisti, že zjednodušený výraz je v minimálním tvaru.

Minimalizace logických funkcí pomocí Karnaughovy mapy je umožněna sousedností políček lišících se pouze v jedné hodnotě. Oproti předchozí metodě je omezena počtem proměnných na max. 6, ale při zjednodušování je tato metoda velmi rychlá a efektivní, protože výsledná funkce je v minimálním tvaru. Jelikož je metoda založena na vizuálním rozpoznávání skupin sousedních buněk, je opět nevhodná pro realizaci výpočtu na počítači.

U Quine-McCluskeyho algoritmu se funkce rozdělí do skupin se stejným počtem negací a následovně se dále upravuje podle zmíněných pravidel v kapitole 4.3. Díky tomu je tato metoda oproti předchozím dvěma vhodná pro realizaci na počítači. Rovněž není omezena počtem proměnných, což je bezesporu výhodou. Ovšem i tento algoritmus má svá určitá omezení, kterým je problém pokrytí výrazů ÚNDF minimálními implikanty.

6. Závěr

Cílem této bakalářské práce bylo popsat jednotlivé metody minimalizace logických funkcí a uvést jejich výhody a nevýhody.

Po nastudování doporučených literárních pramenů a zdrojů z internetu byla vypracována bakalářská práce. Úvodní část práce se zabývá Booleovou algebrou, jejími zákony a také možnostmi vyjádření Booleových funkcí. Těžištěm práce je popis metod minimalizace logických funkcí. Jak již bylo uvedeno v kapitole 4, vedle přímé aplikace zákonů Booleovy algebry existují specializované metody, jejichž efektivita závisí na tvaru zadané logické funkce. Pokud budeme mít dostatek informací o logické funkci, nejlépe pokud ji známe, můžeme zvolit vhodnou metodu pro zjednodušení dané funkce.

Porovnáním těchto metod bylo zjištěno, že algebraická minimalizace a minimalizace pomocí Karnaughovy mapy jsou nevhodné pro realizaci výpočtu na počítači. Tento problém odstraňuje Quine-McCluskeyho algoritmus. Dá se říci, že v případě, kdy daná logická funkce obsahuje 3 nebo 4 proměnné, nemusíme se bát použít první dvě metody. Zvláště použití Karnaughovy mapy pro takové funkce je velmi rychlé, efektivní a zaručuje nalezení minimálního tvaru, i když nemáme mnoho zkušeností s minimalizací funkcí. Pokud má daná funkce více než 6 proměnných, je bezesporu vhodné hledat minimalizovaný tvar logické funkce pomocí softwarového řešení.

Seznam použité literatury:

[1] Balátě, J.: *Automatické řízení*. BEN – technická literatura, Praha, 2003.

[2] Čulík, K., Skalická, M., Váňová, I.: *Logika I*. VUT FE, Brno, 1968.

[3] Švarc, I., Šeda, M., Vítečková, M.: *Automatické řízení*. CERM, Brno, 2007.

[4] Hronek, J., Mazura, J.: *Struktura počítačů* [PDF dokument]. Olomouc, 2006, Dostupný z: <http://www.inf.upol.cz/study/dist/xstrp.html>

[5] Klimeš, C.: *POLPO-Prvky elektronických počítačů, logické obvody* [PDF dokument]. Ostrava, 2008, Dostupný z: <http://www.informatika-osu.czechian.net/skripta.html>

[6] Šeda, M.: Heuristic Set-Covering-Based Postprocessing for Improving the Quine-McCluskey Method, *International Journal of Computational Intelligence*, vol. 4, 2007, No.2, pp.139-143. ISSN 1304-2386

Seznam příloh:

1. CD s vypracovanou bakalářskou prací ve formátu pdf