

Aplikace matematiky

Svatopluk Slavíček; Bohuslav Balcar

Program minimizace plně neurčeně logické funkce na samočinném počítači

Aplikace matematiky, Vol. 11 (1966), No. 4, 278–282

Persistent URL: <http://dml.cz/dmlcz/103030>

Terms of use:

© Institute of Mathematics AS CR, 1966

Institute of Mathematics of the Czech Academy of Sciences provides access to digitized documents strictly for personal use. Each copy of any part of this document must contain these *Terms of use*.



This document has been digitized, optimized for electronic delivery and stamped with digital signature within the project *DML-CZ: The Czech Digital Mathematics Library* <http://dml.cz>

PROGRAM MINIMIZACE PLNĚ NEURČENÉ LOGICKÉ FUNKCE
NA SAMOČINNÉM POČÍTAČI

SVATOPLUK SLAVÍČEK, BOHUSLAV BALCAR

(Došlo dne 13. dubna 1964.)

Logickou funkcí nazýváme funkci konečného počtu dvouhodnotových proměnných (hodnoty se značí zpravidla 0 a 1), která sama může nabývat také jen dvě hodnoty. Vylučujeme dále triviální případy, kdy logická funkce je identicky rovna 0 nebo 1. Minimizací rozumíme úlohu, která spočívá ve vyjádření dané logické funkce disjunktivním normálním výrazem, skládajícím se z nejmenšího počtu symbolů. Při řešení úlohy minimizace mají důležitý význam pojmy implikant a prostý implikant. Implikantem logické funkce $f(x_1, \dots, x_n)$ je elementární konjunkce, sestávající z některých proměnných x_1, \dots, x_n nebo jejich negací, přičemž nabývá-li funkce f pro dané hodnoty x_1, \dots, x_n nulové hodnoty, je pro tyto hodnoty také elementární konjunkce rovna nule. Implikant nazýváme prostým, když žádná jeho vlastní část není již implikantem. Existuje triviální algoritmus vytvoření minimálního disjunktivního výrazu. Nechť n je počet proměnných dané funkce. Vytvoříme nejprve všechny možné konjunkce, které závisí nejvýše na n proměnných (jejich počet je $\sum_{k=0}^n \binom{n}{k} 2^k = 3^n$)

a potom z těchto konjunkcí sestavíme všechny možné disjunktivní normální výrazy dané funkce, které neobsahují dvě stejné konjunkce. V souboru takto vzniklých výrazů budou obsaženy všechny minimální disjunktivní výrazy dané funkce a tyto pak můžeme získat prostým výběrem. Triviální řešení této úlohy je spojeno s výběrem ohromného množství disjunktivních normálních výrazů a již při nevelkém počtu proměnných (4–5) je prakticky neproveditelné. Proto pro úspěšné řešení úlohy minimizace byly vypracovány metody, které zmenšují množství vybíraných disjunktivních normálních výrazů. W. O. QUINE dokázal, že libovolný minimální disjunktivní normální výraz logické funkce se skládá pouze z některých prostých implikantů zadané funkce. Disjunkce prostých implikantů ekvivalentní logické funkci, z nichž nelze ani jeden prostý implikant vypustit, aniž by se narušili podmínky ekvivalentnosti, se nazývá nezbytným výrazem logické funkce. Je zřejmé, že každý minimální výraz je zároveň nezbytný, avšak ne každý nezbytný výraz je minimální. Celý proces minimizace je proto možno rozdělit na dvě etapy:

1. nalezení všech prostých implikantů a sestavení všech nezbytných výrazů ekvivalentních zadané logické funkci,

2. přebrání všech nezbytných výrazů a nalezení alespoň jednoho minimálního výrazu.

Pro minimalizaci logických funkcí, které mají maximálně 5–7 proměnných, byly vytvořeny speciální algoritmy minimalizace. Tyto algoritmy však pro přílišnou pracnost a počet nutných operací nejsou prakticky použitelné při větším počtu proměnných bez pomoci počítačích strojů. Pro využití počítačích strojů se jeví jako nejvhodnější metoda M. A. GAVRILOVA [1], která vychází z praktických technických požadavků. Tato metoda je zvláště vhodná u tzv. plně neurčených logických funkcí, které jsou v technické praxi hojně rozšířeny. Jsou to funkce, jejichž hodnoty nejsou určeny pro všechny možné kombinace nezávisle proměnných.

Mějme jednotaktní releové zařízení s n vstupy a jedním výstupem. Na vstup přivedme současně signály x_1, x_2, \dots, x_n , jejichž hodnota je rovna 1 nebo 0. Předpokládejme, že pro k kombinací vstupních signálů, které označíme F , dostáváme na výstupu signál $z = 1$ a pro m kombinací vstupních signálů, které označíme G , dostáváme na výstupu signál $z = 0$. Zbylé kombinace, jejichž počet je $2^n - (k + m)$, jsou pro nás lhostejné ze dvou důvodů:

1. buď část z nich se na vstupu releového zařízení nemůže objevit v rámci pracovního režimu předcházejícího obvodu,
2. nebo objeví-li se na vstupu, pak pro činnost následujícího zařízení je lhostejné, bude-li na výstupu $z = 0$ nebo $z = 1$.

O logické funkci $f(x_1, \dots, x_n)$ (plně určené) říkáme, že pokrývá množinu F , když tato funkce nabývá pro každou kombinaci z F hodnotu rovnou jedné. Gavrilovův algoritmus „minimalizace“ vede k sestrojení nezbytného výrazu nějaké logické funkce, která má tu vlastnost, že pokrývá množinu F a nepokrývá žádný prvek množiny G . Předpokládejme, že množiny F a G máme uspořádány do matic, tj. $F = (a_{ij})_k^n$ a $G = (b_{ij})_m^n$, kde a_{ij}, b_{ij} jsou nuly nebo jedničky. Zkoumejme, zda a_{11} je totožné s některým prvkem b_{i1} pro $1 \leq i \leq m$. Jestliže tomu tak je, zkoumejme totožnost a_{12} s prvky b_{i2} respektive a_{13} a b_{i3}, \dots, a_{1n} a b_{in} .

Dále musíme brát všechny kombinace $(a_{1j_1}, a_{1j_2}), 1 \leq j_1 < j_2 \leq n$ ze dvou prvků a srovnat s příslušnými kombinacemi (b_{ij_1}, b_{ij_2}) každého řádku matice G . Takto stále prodlužujeme délku vybrané kombinace až nalezneme kombinaci $(a_{1j_1}, \dots, a_{1j_r})$, která je různá od $(b_{ij_1}, \dots, b_{ij_r})$ pro každé $1 \leq i \leq m$. Sestavme konjunkci $g_1 = \tilde{x}_{j_1} \wedge \dots \wedge \tilde{x}_{j_r}$, kde

$$\tilde{x}_{j_q} = x_{j_q}, \text{ když } a_{1j_q} = 1$$

a

$$\tilde{x}_{j_q} = \neg x_{j_q}, \text{ když } a_{1j_q} = 0.$$

Konjunkce g_1 zřejmě pokrývá první řádek matice F . Vyloučíme všechny řádky F , které jsou pokryty g_1 . Obdržíme matici F_1 s menším počtem řádků než má F a celý proces dále opakujeme. Pokračujeme tak dlouho, až všechny řádky F jsou vyčerpány. Získáme tímto konjunkce g_1, \dots, g_r , přičemž $r \leq k$. Položíme-li

$$f(x_1, \dots, x_n) \stackrel{\text{Df}}{=} g_1 \vee \dots \vee g_r,$$

obdržíme logickou funkci pokrývající F a nepokrývající žádný prvek G , přičemž výraz na pravé straně je jejím nezbytným výrazem.

Abychom našli skutečně minimální výraz plně neurčené logické funkce, bylo by třeba nalézt všechny prosté implikanty, z nich sestavit nezbytné výrazy a mezi nimi vybrat minimální ve smyslu nejmenšího počtu symbolů. KAZAKOV dokázal [2], že počet prostých implikantů logické funkce různého typu nepřevyšuje počet prostých implikantů symetrických funkcí při stejném počtu proměnných. V jeho práci je rovněž ukázáno jak prudce narůstá množství prostých implikantů při zvětšování počtu proměnných. Výběr tak velkého množství prostých implikantů a jejich uspořádání do nezbytných výrazů nemůže být provedeno ani samočinným počítačem a to jak z důvodu velkého množství operací tak i požadavku na nadměrně velkou paměť samočinného počítače.

V současnosti neexistuje a jak se zdá ani existovat nemůže prakticky použitelná metoda, která by umožňovala sestavovat minimální výrazy libovolných logických funkcí o větším počtu proměnných, jejichž minimálnost by byla zaručena. Naším algoritmem tedy nezískáme minimální výraz plně neurčené logické funkce, nýbrž z hlediska technické praxe vyhovující nezbytný výraz dané funkce. Tím značně zmenšíme množství strojových operací i požadavky na kapacitu paměti samočinného počítače. Algoritmus minimalizace může mít v ALGOLU-60 následující tvar:

begin integer k, m, n :

read (k, m, n);

begin integer g, i, e, x, y, z ; **Boolean array** $F[1 : k, 0 : n]$,

$G[1 : m, 1 : n]$, $E[1 : n, 0 : n]$, $H[1 : n]$, $M[1 : n]$;

integer array $B[1 : n]$; **Boolean** J ;

read (F, G);

comment: pole F určuje ty členy logické funkce, které mají hodnotu **true**, pole G pak určuje členy logické funkce, jež mají hodnotu **false**. Nultý sloupec pole F je pomocný. Při zavedení musí mít všechna jeho písmena hodnotu **true**. Pole E, H, B jsou pomocná;

origin: **for** $i := 1$ **step** 1 **until** n **do** $B[i] := 0$;

for $i := 1$ **step** 1 **until** n **do**

for $e := 1$ **step** 1 **until** n **do** $E[i, e] := \text{false}$;

$g := 1$;

term: $i := B[g] + 1$; $B[g] := i$; $E[g, i - 1] := \text{false}$;

if $i \leq (n + 1 - g)$ **then**

begin $E[g, i] := \text{true}$; **if** $g = 1$ **then go to sum** **else**

begin $B[g - 1] := B[g]$; $g := g - 1$; **go to term**

end;

end else

begin $g := g + 1$; **if** $g \leq n$ **then go to term** **else go to K**

end;

sum: for $i := 1$ step 1 until n do $H[i] := \text{false}$;

for $i := 1$ step 1 until n do

for $e := 1$ step 1 until n do

$H[i] := H[i] \vee E[e, i]$;

comment: vytváří se extrakční pole H . V prvním cyklu dostáváme do prvního sloupce pole H hodnotu **true** a do ostatních hodnotu **false**. V druhém cyklu je hodnota **true** posunuta o jedno místo vpravo atd. až vyčerpáme všechny kombinace z jedné hodnoty **true** a $(n - 1)$ hodnot **false**. Dále získáváme všechny kombinace ze dvou hodnot **true** a $(n - 2)$ hodnot **false**, ze tří hodnot **true** a $(n - 3)$ hodnot **false** atd. až po jedinou kombinaci z n hodnot **true**. Extrakční pole H se získává logickým součtem jednotlivých sloupců pole E ;
 $x := 1$;

A: if $F[x, 0]$ then go to L else

begin $x := x + 1$; if $x \leq k$ then go to A

else go to K

end;

L: for $e := 1$ step 1 until n do $M[e] := \text{false}$;

for $y := 1$ step 1 until m do

begin for $z := 1$ step 1 until n do

$M[z] := \neg (F[x, z] \equiv G[y, z]) \wedge H[z]$; $J := \text{false}$;

for $e := 1$ step 1 until n do $J := J \vee M[e]$;

if $\neg J$ then go to term

end;

comment: postupně je prověřováno, zda symbol řádku pole F , který má stejnou polohu jako hodnota **true** v extrakčním poli H se liší od písmene ve stejném sloupci pole G ve všech jeho řádcích. Jakmile se objeví aspoň jedna totožnost odpovídajících si písmen přechází se k návěští **term** a vytváří se další kombinace extrakčního pole H ;

print ($H[1 : m]$, x);

comment: tiskne se extrakční pole H a hodnota x , která udává řádek pole F . Do prostého implikantu g se berou ty symboly řádku x , na jejichž místech má pole H hodnotu **true**;

$F[x, 0] := \text{false}$; $i := x$;

for $e := 1$ step 1 until n do

$M[e] := \text{false}$;

for $x := x + 1$ while $k - x \geq 0$ do

begin for $z := 1$ step 1 until n do

$M[z] := ((F[x, z] \equiv F[i, z]) \wedge H[z])$; $J := \text{true}$;

for $e := 1$ step 1 until n do

$J := J \wedge (M[e] \equiv H[e])$; if J then $F[x, 0] := \text{false}$

end;

go to origin;

comment: je-li nalezen prostý implikant určitého řádku pole F , zkoumá se, zda ještě některé jiné řádky pole F jsou pokryty tímž prostým implikantem. V kladném případě se mění ve všech řádcích, které mají společný prostý implikant, hodnota **true** na **false**;

end;

K:

end

Program je ukončen, jsou-li ve všech místech nultého sloupce pole F hodnoty **false**, nebo když jsou vyčerpány všechny kombinace hodnot **false** a **true** v extrakčním poli H . Druhý případ svědčí o tom, že v poli F i G je tentýž řádek, což není přípustné. Jako výsledek dostáváme disjunctci prostých implikantů $g_1 \vee g_2 \vee \dots \vee g_r$, která pokrývá všechny členy F a nepokrývá ani jeden člen G .

Popsaný algoritmus byl použit pro sestavení programu na samočinný počítač URAL-2. Při výpočtu byla využívána jen operační paměť. Maximální počet řádků F a G byl 3552, počet proměnných byl menší než 19.

Literatura

- [1] *Гаврилов М. А.*: Минимизация булевых функций, характеризующих релейные цепи. Автоматика и телемеханика XX (1959), № 9.
- [2] *Казаков В. Д.*: Нахождение максимального числа простых импликантов произвольной логической функции n переменных. Автоматическое управление, ДАН СССР.

Резюме

ПРОГРАММА МИНИМИЗАЦИИ НЕДООПРЕДЕЛЕННОЙ ФУНКЦИИ АЛГЕБРЫ ЛОГИКИ НА ВЫЧИСЛИТЕЛЬНОЙ МАШИНЕ

СВАТОПЛУК СЛАВИЧЕК, БОГУСЛАВ БАЛЦАР
(SVATOPLUK SLAVIČEK, BOHUSLAV BALCAR)

В статье рассматривается вопрос минимизации функций алгебры логики, имеющих форму нормальных дизъюнктивных выражений. Определяется на АЛГОЛЕ-60 алгоритм нахождения простых импликантов недоопределенной функции алгебры логики данного типа.

Summary

COMPUTER MINIMIZATION OF INCOMPLETELY SPECIFIED LOGICAL FUNCTIONS

SVATOPLUK SLAVIČEK, BOHUSLAV BALCAR

This paper is concerned with the minimization of logical functions in normal disjunctive form. An ALGOL-60 algorithm is presented for finding the prime impli-cants of incompletely specified logical functions of the given form.

Adresa autorů: Inž. Svatopluk Slaviček, Vojenská akademie Antonína Zápotockého, Brno. — Bohuslav Balcar, Matematicko-fyzikální fakulta KU, Sokolovská 83, Praha 8 — Karlín.