

55

F50

TK 44.356

KFKI-73-21

Hegedüs Cs.

MÁGNESES BUBORÉKTÁROLÓK VIZSGÁLATA
A LOGIKAI MŰVELETVÉGZÉS SZEMPONTJÁBÓL

Hungarian Academy of Sciences

CENTRAL
RESEARCH
INSTITUTE FOR
PHYSICS

BUDAPEST

2017

KFKI-73-21

MÁGNESES BUBORÉKTÁROLÓK VIZSGÁLATA A LOGIKAI MŰVELETVÉGZÉS
SZEMPONTJÁBÓL

Hegedűs Csaba

Központi Fizikai Kutató Intézet, Budapest
Számítástechnikai Főosztály

ABSTRACT

"Investigations of magnetic bubble operations from the point of view of representing switching functions."

This paper gives a brief survey on the new research field of magnetic bubble logic. As a necessary tool the elements of the theory of switching functions are cited.

Later some transfer operations or instructions among magnetic bubble locations are investigated from the point of view of representing switching functions. A care distinction is made between the logical completeness of switching functions and the computational completeness of a set of bubble instructions. Finally due to Friedman and Menon computationally complete systems of instructions are presented.

KIVONAT

A dolgozat áttekintést ad a címben megjelölt új kutatási terület kezdeti eredményeiről. Mivel a tárgyalásmód igényli a logikai függvények elméletének elemeit, ezért a bevezető rész egy ilyen áttekintést tartalmaz.

Ezután az irodalomban leírt rácsműveleteket vizsgálja abból a szempontból, hogy az egyes műveletek segítségével reprezentálhatók-e a logikai függvények. Világosan megfogalmazza az utasításrendszer és a logikai függvényrendszer funkcionális teljessége közötti különbséget. Végül Friedman és Menon dolgozata alapján funkcionálisan teljes utasításrendszereket tárgyal.

РЕЗЮМЕ

Математические изучения магнитных пузырьковых памяти ЭВМ с точки зрения логических операций.

В работе даётся краткий обзор о первых результатах достигнутых в области определенной в заглавии.

Во втором разделе рассматриваются необходимые элементы теории функции алгебры логики (Ф.а.л.). В дальнейшем исследуются возможности представления Ф.а.л. с помощью операций, сделанных между точками пузырьковой памяти.

Различие между функциональной полноты системы Ф.а.л. и полноты системы операции определяется в явном виде. В заключение по работе Фридмана и Менон-а обсуждаются функционально полные системы операции.

1. BEVEZETÉS

Jelen írás célja, hogy egyrészt bevezetőt adjon a címben megjelölt témához, másrészt áttekintse az első próbálkozásokat.

A mágneses buboréktárolóban az információ hordozói vékony kristálylapban elhelyezkedő, hengeres alakú mágneses tartományok, másként mágneses buborékok. Ezek a buborékok kellő erősségű mágneses térben stabilak, könnyen mozgathatók.

A mágneses buboréktárolókat elsősorban tömegtárolók céljára kívánják felhasználni a nagy tárolási sűrűségük $/10^5 - 10^6 \text{ bit/cm}^2/$ miatt.

A kutatások másik iránya viszont azt célozza, felhasználhatók-e a buboréktárolók aktiv memóriaként, azaz lehetséges-e a tárolt információval szükséges műveleteket magában a memóriában elvégezni. Ezt a kérdést a buborékok könnyű mozgathatósága és egymás közötti taszító kölcsönhatása veti fel.

A fejlődés jelenlegi szakaszában a buborékok számára rács készíthető a kristálylapon, és a buborékok mozgatása rácspontról rácspontra történhet. Lehetséges a buborékok keltése és megsemmisítése is.

Ahhoz, hogy a tárolt információval minden szükséges műveletet el tudjunk végezni, meglehetősen hatékony műveleteket kell megvalósítani az egyes rácspontokban ülő buborékok között.

Minthogy a buborékok apró mágnesek, a taszító hatás a rácspontok közötti buborékműveletek alapja. Jelenleg már több különböző művelet megvalósítható. A későbbiekben e műveletek elemzése a célunk.

Előbb azonban néhány szót ejtünk a logikai függvényekről, amelyek a buborékműveletek vizsgálatakor hasznos segítőtársaink lesznek.

2. A LOGIKAI FÜGGVÉNYEK /LF/

Röviden összefoglaljuk a későbbiek megértéséhez szükséges ismereteket.

A logikai értékek reprezentációjául a 0 és 1 számokat választjuk. A 0 legyen a "hamis", az 1 pedig az "igaz" logikai érték megfelelője. Egy logikai változó ennek megfelelően a 0 és 1 értékek valamelyikét veheti fel. Az n-változós logikai függvény n logikai változóhoz rendel egy logikai értéket. A logikai függvényre a későbbiek során az LF rövidítést fogjuk használni. Az LF-eknek a szakirodalomban sokféle elnevezése van, mondanak és írnak még kapcsoló, kombinációs, Boole- vagy igazságfüggvényeket is.

Az LF-ek megadása történhet táblázattal. Erre az 1. és 2. táblázatban adunk példát.

-	0	1
	1	0

negáció

$$f(x) = \bar{x}$$

+	0	1
0	0	1
1	1	1

VAGY

$$f(x,y) = x+y$$

.	0	1
0	0	0
1	0	1

ÉS

$$f(x,y) = x \cdot y$$

⊕	0	1
0	0	1
1	1	0

"Kizáró Vagy" /KIV/

$$\begin{aligned} f(x,y) &= x \oplus y = \\ &= \bar{x} \cdot y + x \cdot \bar{y} \end{aligned}$$

/	0	1
0	1	1
1	1	0

"Nem ÉS" /NÉS/

$$\begin{aligned} f(x,y) &= x/y = \\ &= \overline{x \cdot y} \end{aligned}$$

1. táblázat

A függvények értékei a jobb alsó sarokban vannak. A bal oldali oszlop y, a felső sor pedig x lehetséges értékeit tartalmazza.

A kétváltozós függvények tabellázása az 1. táblázatban négyzethálóban történt. Itt a következőkre kell felhívni a figyelmet: mindegyik függvényt egyben műveletként is értelmezhetjük. Az első példa egyváltozós /unáris/ logikai művelet, a negáció vagy tagadás. A többiek pedig két logi-

kai változó között értelmeznek műveletet. Ezek közül a logikai szorzás /konjunkció, ÉS-művelet/ és a logikai összeadás /diszjunkció, VAGY-művelet/ az ismertebbek.

A "Kizáró Vagy" /KIV/ művelet a közöséges VAGY művelettől abban különbözik, hogy csak akkor "igaz" a függvény értéke, ha a két változó logikai értéke ellentétes. A "Nem ÉS" /NÉS/ művelet, amelyet gyakran Sheffer-vonásnak is neveznek, az ÉS művelet eredményének tagadásával jön létre.

Az értéktáblázatok alapján könnyű ellenőrizni, hogy az ÉS és VAGY műveletek asszociatívok és kommutatívok, tehát több változó esetén a zárójelzés elhagyható és a sorrend közömbös. Ugyanez viszont a NÉS /Scheffer-vonás/ műveletre nem áll, tehát itt a zárójelzés szükséges.

Hasonlóképpen látható be az is, hogy az ÉS és VAGY műveletek a VAGY műveletre nézve a disztributív tulajdonsággal is rendelkeznek.

Indukcióval igazolható, hogy az n -tagu logikai összeg végeredménye 1 ha benne legalább egy logikai változó értéke 1, különben 0, és az n -tényezős logikai szorzat értéke 0, ha legalább egy tényezője 0, egyébként 1.

Ha logikai műveletek segítségével kívánjuk megadni a LF-t, akkor az egy- és kétváltozós függvények közül kiválasztunk néhányat és ezekkel, mint műveletekkel végezzük el a függvény megadását.

Most természetesen felvetődik a kérdés: a kiválasztott műveletekkel tetszőleges LF-t meg tudunk-e adni? Hogy elegendő számú művelet kiválasztható, arra példa a következő állítás: Minden n -változós LF az ÉS, VAGY és negáció műveletével előállítható.

Az állítást egy 3-változós LF-en szemléltetjük. Ennek a függvénynek az értéktáblázatát a 2. táblázat tartalmazza. A táblázatban baloldalt a változók lehetséges értékei állnak, jobboldalt pedig a nekik megfelelő függvényértékek. A bal oldalt úgy rendeztük el, hogy az előálló 3-jegyű kettes számrendszerbeli számok egyenlők legyenek a sorszám értékével.

Itt a változók összes lehetséges variációja fel van tüntetve, n változó esetén ezek száma 2^n , ami jelen esetben $n=3$ folytán egyenlő 8-cal. Közbevetőleg megjegyezzük, hogy az összes n változós LF-ek száma annak számával egyenlő, ahányféleképpen a függvényértékek oszlopa kitölthető. Eredményül $2 \cdot 2^n$ -ed rendű ismétléses variációink számát kapjuk, 2^{2^n} -et.

A következőképp okoskodunk: minden értékhármashoz hozzárendeljük a változók egy háromtényezős szorzatát, pl.

$$(0,0,1) \leftrightarrow \bar{x} \cdot \bar{y} \cdot z ,$$

ahol a változót negáljuk, ha az értékhármásban szereplő neki megfelelő érték 0. Ezáltal olyan szorzatot kaptunk, amelynek értéke csupán a hozzátartozó értékhármás mellett lesz 1, egyébként 0.

A logikai összeg korábban adott értelmezésénél fogva, ha azon szorzatok összegét vesszük, amelyekhez a függvény az "igaz" értéket rendeli; előállítottuk a függvényt:

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

Ezt a logikai függvények kanonikus előállításának nevezzük. A kanonikus

	x	y	z	f(x,y,z)
0.	0	0	0	0
1.	0	0	1	1
2.	0	1	0	1
3.	0	1	1	0
4.	1	0	0	1
5.	1	0	1	0
6.	1	1	0	1
7.	1	1	1	1

2. táblázat

alak jellemzője, hogy minden szorzatban minden változó /negálva vagy anélkül/ csupán egyszer szerepel. A logikai összegzést hangsúlyozva nevezik ezt az alakot diszjunktív normálformának is.

Mind az összegekben /ha az nem többtényezős szorzatok összege/, mind a szorzatokban felesleges, hogy ugyanaz a változó többször szerepeljen az

$$x+x = x, \quad x+\bar{x} = 1, \quad x \cdot x = x, \quad x \cdot \bar{x} = 0$$

összefüggések és az asszociatív és kommutatív tulajdonságok következtében.

Ugyanezen összefüggések alapján van mód a kanonikus alak redukálására, ha ez lehetséges. Redukáláskor a műveletek számát kívánjuk a lehetőség szerint csökkenteni. Bemutatjuk az egyszerűsítés egy lépését. Példánkban az utolsó két szorzatot a következőképpen hozhatjuk egyszerűbb alakra:

$$xy\bar{z} + xyz = xy(z+\bar{z}) = xy.$$

Kezdetben láttuk, műveletek bevezetésére több lehetőségünk van. A műszaki megvalósításkor felvetődik tehát a kérdés: mely műveleteket érdemes kiválasztani a sok közül?

Ennek megválaszolásában segít bennünket a következő észrevétel: a tagadás és a szorzás segítségével az összeadás megvalósítható:

$$\overline{\overline{x \cdot y}} = x + y ,$$

és hasonló összefüggés a szorzásra is fennáll:

$$\overline{(\overline{x+y})} = x \cdot y$$

Ebből következik, hogy nem szükséges egyszerre mindhárom műveletet realizálni. Kézenfekvő a következő definíció: Az egy és két változóra értelmezett műveleteknek egy halmazát funkcionálisan teljesnek nevezzük, ha segítségével a behelyettesítés véges sokszori alkalmazásával tetszőleges LF előállítható.

Valamely műveletrendszer funkcionálisan teljes, ha elemeinek segítségével egy már ismert funkcionálisan teljes műveletrendszer elemei megvalósíthatók.

Funkcionálisan teljes művelethalmazra további példák:

a/ {KIV,ÉS}: ugyanis $\overline{x} = x \cdot 0$, és már láttuk, hogy a negáció és a szorzás funkcionálisan teljes műveletrendszer.

b/ {NÉS}: minthogy $\overline{x} = x/1$ és $xy = (x/y)/(x/y)$.

A NÉS művelet tehát önmagában funkcionálisan teljes. Az ilyen műveleteket univerzális műveleteknek nevezzük.

A LF-ek áttekintését ezzel befejeztük. A LF-ekről bővebb ismertetés pl. a hivatkozott angol és magyar nyelvű irodalomban található. Ezekután rátérünk a buboréktárolók vizsgálatára.

3. RÁCSMŰVELETEK, KÓDOLÁS

Feltesszük, hogy a buborékok egy n pontból álló rácson helyezkedhetnek el. Egy rácspontnak két állapota lehetséges: vagy van ott buborék, vagy nincs. Nem nehéz belátni, hogy a teljes rácsnak így összesen 2^n különböző állapota létezik.

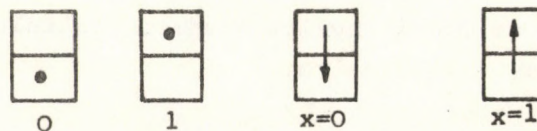
Egy rácsművelet alkalmazása során kívülről fizikailag egy vagy több rácspont állapotát előírt módon megváltoztatjuk. Legegyszerűbb rácsművelet a buborékok törlése mágneses térrel. Jelenleg a következő - csak néhány rácspontot érintő - műveleteket lehet már megvalósítani:

- 1/ Eltolás, (a,b) . Hatására az "a" rácspontból a "b" rácspontba kerül a buborék, feltéve, hogy volt buborék az "a" pontban és "b" pedig üres. Minden egyéb esetben az utasítás hatástalan. Ábráinkon ezt az utasítást az "a" pontból a "b" pontba mutató nyillal fogjuk jelölni.
- 2/ Buborék keltése, $(1,a)$. Ekkor a művelet alkalmazása után az "a" rácspontba buborék kerül.
- 3/ Buborék megsemmisítése, $(0,a)$. Alkalmazása után "a" üres lesz.
- 4/ Hasítás, $(a,b)_h$. Ha "b" üres és "a"-ban van buborék, akkor ennek elhasításával a művelet "b"-ben buborékot kelt, egyébként hatástalan. Ez a művelet tehát csak akkor másol, ha "b" üres.
- 5/ Romboló eltolás, $(a,b)_r$. Ez a közönséges eltolástól abban különbözik, hogy ha "a" és "b" is tartalmaz buborékot, akkor a "b"-ben levő buborék megsemmisül.
- 6/ Feltételes eltolás, $(a,b)_c$. Az eltolás csak akkor valósul meg, ha "c"-ben van buborék.

A kódolásnál két lehetőséget fogunk használni:

1-pozíciós kód. Itt egy rácspont két lehetséges állapota ábrázol 1 bit információt. Természetes a 0-s állapotot az üres, az 1-es állapotot pedig a buborékot tartalmazó esettel azonosítani. Az 1- pozíciós kód mellett a buborékkeltés és megsemmisítés az értékadó műveletek.

2-pozíciós kód. Ekkor két rácspont és egy buborék ábrázolnak 1 bit információt. A buborék mindig a két pont valamelyikében van. A továbbiakban mi a következő konvenciót használjuk:



1. ábra

Ha alul van a buborék, jelentse ez a 0-s, ha felül van a buborék, akkor pedig az 1-es állapotot. Az eltolás műveletét nyilakkal ábrázolva most szemléletes lesz a le- és felfelé mutató nyilak, mint értékbeállító utasítások használata. A lefelé mutató nyíl a nullázást, a felfelé mutató nyíl pedig az 1-es érték adását fogja jelenteni. /1. ábra/.

Program alatt az utasításoknak egy véges sorozatát értjük. Az utasításokat a korábban ismertetett rácsműveletek jelenthetik, melyeket a végrehajtás sorrendjében írunk egymás után.

4. RÁCSMŰVELETEK ÉS LOGIKAI FÜGGVÉNYEK

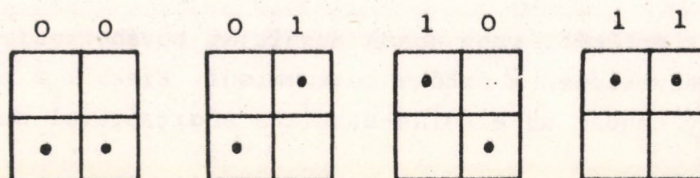
A felsorolt műveletek közül az eltolás valósítható meg legegyszerűbben. Éppen emiatt fontos annak az ismerete, hogy ez a művelet mire képes, segítségével pl. meg lehet-e valósítani az összes LF-t.

Az eltolás műveletének számos tulajdonságát Graham /1970/ tárta fel.

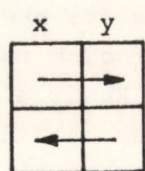
A továbbiakban itt csak az eltolás-utasításokból álló programok vizsgálata a célunk.

A 2. ábra néhány lehetőséget mutat 2-pozíciós kód mellett két logikai változó közötti logikai műveletek realizálására. x, y a művelet végzése előtt, x', y' pedig a művelet végzése után jelöli az x és y logikai változó értékeit. A rajzok alá írtuk azokat a logikai összefüggéseket, amelyeket a két nyilból álló program megvalósít.

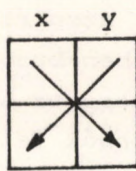
Ezeket az összefüggéseket a 2/a ábrán felsorolt lehetséges kiinduló állapotok segítségével ellenőrizhetjük.



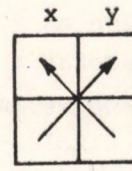
a/ Két szomszédos logikai változó lehetséges különböző állapotai



$$\begin{aligned} A: x' &= xy \\ y' &= x + y \end{aligned}$$



$$\begin{aligned} B: x' &= x\bar{y} \\ y' &= \bar{x}y \end{aligned}$$



$$\begin{aligned} C: x' &= x + \bar{y} \\ y' &= \bar{x} + y \end{aligned}$$

b/ Logikai műveletek realizálása 2 változó között

2. ábra

Azt gondolhatnánk, hogy az itt felsorolt A, B, C programokból tetszőleges LF felépíthető, mivel az A program megvalósítja az ÉS és VAGY műveletet, a negálás pedig a B vagy C program segítségével egy segédváltozó felhasználásával elérhető. Azonban, mint később látni fogjuk, a dolog nem ilyen egyszerű.

Ugyanis a LF-ek előállításakor hallgatólagosan feltételeztük, hogy bármely változó értékét vagy annak negáltját annyiszor érhetjük el, ahányszor szükséges. Ez a gyakorlatban azt jelenti, hogy nem elég, ha bináris információval műveleteket tudunk végezni. Bináris információ másolására is szükség van, hiszen a műveletek végzése során, mint azt látni fogjuk, a bináris információ megsemmisülhet.

Ezek után próbáljunk meg másolni! Azt kívánjuk tehát, hogy az x rekesz tartalmát csupán eltolás-utasításokkal vigyük át az y rekeszbe, meghozzá úgy, hogy x tartalma változatlan maradjon. Az A program esetében a következőképpen próbálkozunk: y -t nullázzuk, majd alkalmazzuk A-t.

Igy tényleg $y'=x$ lesz, de $x'=x \cdot 0=0$, tehát x értéke megsemmisül.

A B program lehetővé teszi x negáltjának az előállítását: $y=1$, $y'=\bar{x} \cdot 1$, de sajnos, x értéke most is felülíródik: $x'=x \cdot 0=0$.

A C programmal is hasonlóképpen járunk, ha megpróbáljuk x negáltját előállítani.

Tehát x értékét, vagy annak negáltját továbbítani tudjuk ugyan, de a művelet eredményeképpen x értéke megsemmisül. Ezekből a próbálkozásokból sejthetjük, hogy csupán az eltolás-utasítás segítségével nem tudunk másolni.

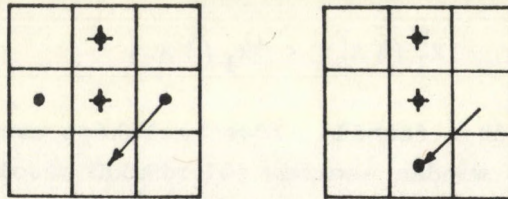
Ahelyett, hogy tovább próbálkoznánk, általánosan a "nem csökkenő átfedés" tételével megmutatjuk, hogy az eltolás-utasításokból nem építhető fel olyan program, amellyel minden esetben másolni lehet.

Jelölje az R rácson X_1 azon rácspontok halmazát, ahol buborék van. Az ilyen halmazokat a későbbiekben buborékalmaznak fogjuk nevezni. X_2 legyen ugyanezen rácson egy másik lehetséges buborékalmaz. Ekkor a nem csökkenő átfedés tétele: eltolás-utasításokból felépített program hatására tetszőleges X_1 és X_2 buborékalmazok közös elemeinek száma nem csökkenhet. Matematikai alakban:

$$|x_1^P \cap x_2^P| \geq |x_1 \cap x_2|$$

ahol x_1^P a P program hatására az x_1 -ből kapott halmazt jelenti, $|x|$ pedig az x halmaz elemeinek számát jelöli.

A tétel röviden belátható: a program mindkét esetben ugyanazoknak az eltolás-utasításoknak az egymásutánját tartalmazza. Egy utasítás hatására minden közös pont továbbra is közös pont marad. Két nem közös pontot az utasítások viszont fedésbe hozhatnak, mint ahogy azt a nyíl is mutatja a 3. ábrán. Így az átfedés tényleg csak nőhet, amivel a bizonyítást befejeztük.

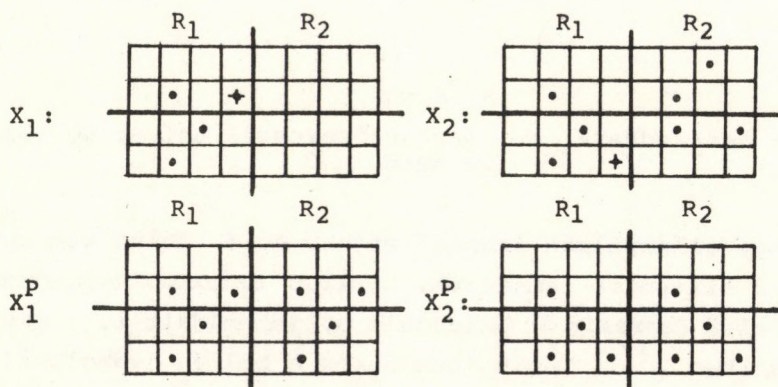


3. ábra

Két különböző buborékhalmoz ugyanazon a rácson. A közös pontok kereszttel jelölve.

Ezek után visszatérünk a másolás problémájához. Természetesen csak olyan másolás felől érdeklődünk, amely nem igényel buborékkeltést vagy -megsemmisítést.

Legyen olyan az általános másolóprogram, amelyre az R_1 rácson levő buborékhalmoz változatlan marad /4. ábra/, és R_2 -ben megjelenik ugyanaz a buborék-konfiguráció, mint ami



4. ábra

A másolás modellje. A P másolóprogram tetszőleges R_1 -beli halmaz képét $(X_1 \cap R_1)$ állítsa elő R_2 -ben, feltéve, hogy $|X_1 \cap R_1| = |X_1 \cap R_2|$, $i=1,2$.

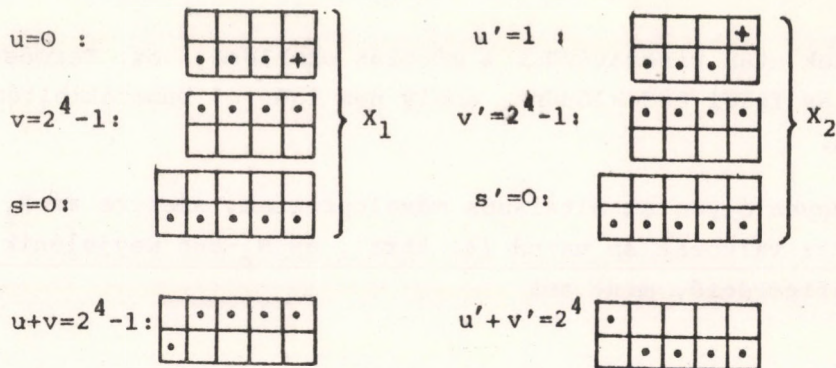
R_1 -ben van. Ez a program legyen univerzális, tehát minden konfigurációt tudjon másolni. De ha ilyen program léteznék, akkor az ellentmondana a nem csökkenő átfedés tételének.

Legyen ugyanis az egyesített $R_1 \cup R_2$ rácson az X_1 buborékhalmoz. X_2 különbözzék ettől egyetlen - az ábrán kereszttel jelölt-pontban. Ekkor a P másoló programnak olyan X_1^P és X_2^P halmazokat kellene eredményeznie, amelyeknek átfedése csökken,

$$|X_1^P \cap X_2^P| < |X_1 \cap X_2|$$

tehát az ellentmondás valóban fennáll. Következésképp csupán az eltolás segítségével nem készíthető minden esetben jól működő másolóprogram.

Hasonló módon látható be az is, hogy az eltolás segítségével nem lehet jó bináris összeadó programot sem készíteni.

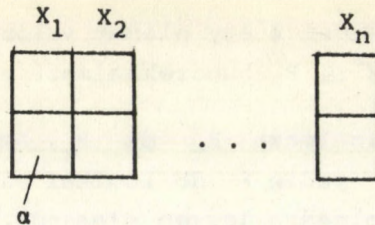


5. ábra

Példa a bináris összeadásra. Az összegezőrekeszek helyes végállapota alul látható

Ezt is egy ellenpéldán szemléltetjük. Az 5. ábrán két különböző összeadás kiinduló állapotát láthatjuk. Az alsó összegezőrekeszek ki vannak nullázva, az első összeadandó baloldalt 0, jobboldalt 1, a második összeadandó mindkét esetben 2^4-1 . Tehát induláskor a bal és jobboldali esetben fellépő buborékhalmozok egyetlen kereszttel jelölt pontban különböznek. Ha a két kezdőhalmazt X_1 és X_2 -vel jelöljük, akkor azt találjuk, a helyes összeadás megkívánja, hogy a P összeadóprogram által eredményezett X_1^P, X_2^P halmazok átfedése csökkenjen, ami szintén ellentmond a nem csökkenő átfedés tételének.

A LF-ek előállítását a következő formában vizsgáljuk:



6. ábra

Összesen $2n$ darab rácspont tartalmazza az n logikai változó értékeit 2 pozíciós kód szerint /6. ábra/. A csak eltolás-utasításokból álló program legyen olyan, hogy a változók tetszőleges kezdőállapotából kiindulva végül egy kiszemelt rácspontban (α) 1-pozíciós kód értelemben szolgáltatssa a függvény helyes értékét.

Graham és munkatársai így a két-, három- sőt négy-változós logikai függvények programjait is elkészítették. De Graham bebizonyította a következő állítást: Biztosan létezik olyan legalább 11-változós LF, amely a fenti módon nem reprezentálható. A bizonyítás úgy történt, hogy Graham n változó esetén meghatározta, hány különböző P programot lehet készíteni ilyen elrendezés mellett, és $n \geq 11$ esetén azt találta, hogy a lehetséges LF-ek száma ennél nagyobb.

Mindezek a vizsgálatok azt mutatják, hogy az eltolás-utasítás önmagában nem elegendő maradéktalanul működő buboréklogika készítésére.

5. AZ UTASÍTÁSRENDSZER TELJESSÉ TÉTELE

Előző megfontolásaink arra vezettek, hogy az eltolást ki kell egészítenünk egyéb utasításokkal. Kézenfekvő a következő definíció, amely a LF-ek köréből ismert funkcionális teljesség megfelelője: Egy utasításrendszer funkcionálisan teljes, ha az utasításrendszer elemeinek segítségével tetszőleges n -változós LF realizálható.

Ez a definíció a LF-ek ismeretében szemléletes. Egy utasításrendszerrel szemben mégsem ez a legszigorubb követelés.

Jelölje az m pontból álló R rács összes részalmazainak halmazát H . Mint tudjuk, H elemeinek / R részalmazainak/ száma 2^m . Legyen ϕ mindazon φ_j

leképezések halmaza, amelyek a H halmazt önmagába képezik le. ϕ elemeinek száma így $(2^m)^{2^m}$ lesz, vagyis egyenlő $2^m 2^m$ -ed rendű ismétléses variációinak számával.

Egy P program voltaképpen ϕ egy elemét valósítja meg, t.i. bármely $X \subseteq R$ buborék-halmazhoz egy $X^P \subseteq R$ buborék-halmazt rendel.

Legyen R -nek két részhalmaza R_t és R_k , amelynek akár közös elemei is lehetnek. R_t n db, R_k pedig i db logikai változó értékeinek valamilyen kódolás szerinti ábrázolására legyen elegendő. R_t -t nevezzük tárgyhalmaznak, R_k -t pedig képhalmaznak.

A funkcionális teljesség ekkor a következőképpen konkretizálható: induláskor R_t tartalmazza n logikai változó megadott értékét. Ezekről függetlenül az $R \setminus R_t$ halmazon is beállíthatunk valamilyen kiinduló állapotot. Megköveteljük, hogy az utasításrendszer elemeiből legyen felépíthető egy olyan program, amelynek hatása után $i=1$ -re R_k egy előírt LF helyes értékét tartalmazza.

Ezt a követelést tovább szigoríthatjuk, ha az R, R_t halmazokat változatlanul hagyjuk, de az elkészítendő LF-ek számát, i -t, növeljük oly módon, hogy R_k i ($\neq 1$) db tetszőleges LF helyes értékét tartalmazza a program működése után.

Ha R_t összes részhalmazainak halmazát H_t jelöli, R_k -ra pedig ugyanez H_k , akkor az előbbi követelés azt kívánja, hogy H_t valamennyi H_k -ba való leképezése legyen megvalósítható. Legyen Ψ mindazon ψ_j leképezések halmaza, amelyek H_t elemeit H_k -ba képezik le. Megkülönböztetésül azt az utasításrendszert, amelynek elemeivel a Ψ leképezések megvalósíthatók, a $H_t \xrightarrow{\Psi} H_k$ leképezésekre nézve teljesnek nevezzük.

Ezzel a szigorubb követeléssel egyrészt a felhasználható segédrekeszek számát csökkentjük, másrészt a logikai műveletvégzés hatékonyságát növeljük.

Jelenleg még nem ismeretes, hogy ilyen szigorubb megkötéseket mennyiben és hogyan érdemes kielégíteni. Természetesen a követelés akkor a legszigorubb, ha $R=R_t=R_k$, amikor is ϕ valamennyi eleme már megvalósítható lesz.

E bevezető után rátérünk a funkcionálisan teljes utasításrendszerek vizsgálatára /Friedman, Menon, 1971./.

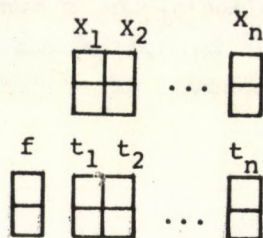
Ha az eltolást kiegészítjük a hasítás, keltés és megsemmisítés műveleteivel, akkor funkcionálisan teljes utasításrendszert kapunk, amihez a hozzátartozó tárgy- és képhalmazt meg fogjuk adni. Ezekkel a műveletekkel biztosan nem lehet a ϕ leképezések mindegyikét megvalósítani, hiszen nem megy a

komplementálás, vagyis nem tudunk olyan P programot összeállítani, amelyre $X^P = R-X$.

Ugyanis ekkor minden rácspont állapotát a kiegészítő állapotra kell változtatni. Az egy-rácspontos műveleteink erre nem alkalmasak. A két-rácspontos műveletek esetén is könnyű ellenpéldát találni, hiszen ha mindkét rácspontban buborék ül, a két-rácspontos utasításaink hatástalanok. Tehát már nem tudunk az 1 vagy 2 rácspontból álló rácásra sem általános komplementáló programot készíteni.

A komplementálás a ϕ elemei közül az egy-egy értelmű leképezések közé tartozik, mivel ez a leképezés megfordítható. Később igazolni fogjuk, hogy a 3. pontban felsorolt műveletek egyikével sem lehet egy-egy értelmű leképezéseket megvalósítani.

A funkcionális teljességet azáltal igazoljuk, hogy megadjuk, hogyan lehet egy n-változós LF programját elkészíteni. A 7. ábrán látható elrendezésből indulunk ki.



7. ábra

A felső rekeszekben $\{x_1, x_2, \dots, x_n\} = R_t$ legyenek a függvény változói. Az alattuk lévő $t_i / i=1, 2, \dots, n/$ rekeszek segédrekeszek és $f=R_k$, amely a LF értéket fogja tartalmazni.

Mivel komplementálni nem tudunk, továbbra is 2-pozíciós kódot kell használnunk, különben nem tudnánk előállítani egy változó negáltját.

Az LF legyen szorzatok összegeként adva. Ekkor egy logikai szorzat kiszámításának és a szorzatok összegezésének lépései a következők:

- A t_1 segédrekeszekben lévő buborékokat megsemmisítjük.
- A hasítás-utasítással a szorzatban szereplő minden változó értékét vagy negáltját átmásoljuk egymás mellé balról kezdődően az alsó rekeszekbe.
- A korábban ismertetett A programmal /2. ábra/ jobbról kezdve sorban össze-szorozzuk a változókat, úgy hogy az eredmény mindig a baloldali rekeszben legyen.

d/ A szorzat tartalmát "f"-hez adjuk, ugyancsak A segítségével.

Induláskor természetesen "f"-et nullázni kell. Ha valamennyi szorzatot összegeztük, akkor a függvény előállításával készen vagyunk.

Az eltolás és a hasítás műveletét helyettesíthetjük egyetlen művelettel, anélkül, hogy elrontanánk a funkcionális teljességet. Erre a romboló eltolás vagy a feltételes eltolás is alkalmas.

Egy pozíciós kód értelemben az "a" és "b" rácspontok egyben logikai változókat ábrázolnak. Az "a" és "b" rácspont között megvalósított romboló eltolás a következőt eredményezi a logikai műveletek nyelvén:

$$a' = a \cdot b, \quad b' = a \circ b = \bar{a}b + a\bar{b} .$$

Tehát az "a" rácspontra a szorzás, a "b" rácspontra pedig a KIV művelet eredménye kerül. Az LF-ek tárgyalásakor már beláttuk, hogy az ÉS és KIV művelet funkcionálisan teljes rendszert ad.

Ha az alsó index azt jelzi, hogy a betűváltozót milyen logikai értékkel indítjuk, akkor az $(a, c_1)_r$ utasítás eredménye: $a' = a \cdot 1 = a$ és $c_1' = \bar{a} \cdot 1 + a \cdot 0$. Tehát "a" értéke nem vészett el, és c_1 -be, "a" komplemente került. Ha ezt a processzust megismételjük, pl. c_1' és b_1 -gyel, akkor b_1 -ben megjelenik "a" negáltja, azaz "a" maga. Így egy segédváltozóval már tudunk másolni, sőt meg a komplementálás is.

Az összeadást az

$$a+b = a \oplus b \oplus (a \cdot b)$$

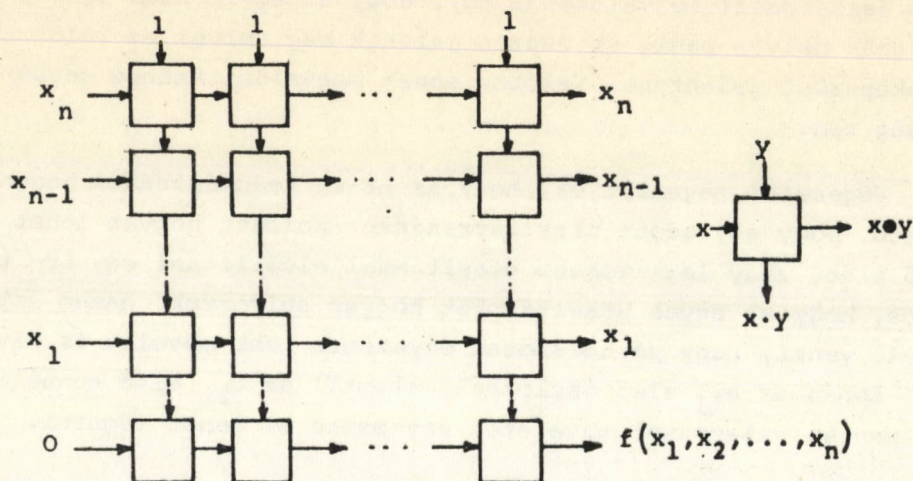
azonosság alapján végezhetjük. Innen látható, hogy az $(a, b)_r$ utasítás két egymást követő alkalmazása "b"-ben az $a+b$ értékét fogja eredményezni és nem nehéz belátni azt sem, hogy "a"-ba pedig 0 kerül.

A romboló eltolás lehetővé teszi, hogy az n-változós LF-ek előállítását 1-pozíciós kódban végezzük el a már ismerttetett eljárás alapján, de csupán a romboló eltolás segítségével is készíthető olyan elrendezés, amelylyel tetszőleges LF-t elő tudunk állítani.

Ezt a 8. ábra mutatja. A táblázatnak n változós LF esetén 2^n oszlopa van. Be lehet látni, hogy az utolsó sorban valamennyi teljes szorzat meg fog jelenni. A szaggatott nyíl azt jelöli, hogy az összeköttetés a kívánalomnak megfelelően megszakítható. Elég csak azokat az oszlopokat bekötni, amelyekben megjelenő teljes szorzat a függvény kanonikus előállításában szerepel.

A másik lehetőség az eltolás és a hasítás helyettesítésére a feltételes eltolás. Ez a művelet 1-pozíciós kód értelemben a következő logikai függvényeket valósítja meg:

$$x' = x \cdot y + x \cdot \bar{z}, \quad y' = x \cdot z + y, \quad z' = z$$



8. ábra

Ezt figyelembe véve $(a_1, b_0)x$ eredményeképpen a buborékkal indított a_1 rácspont x negáltját, " b_0 " pedig x értékét fogja tartalmazni.

Tehát ezzel a művelettel is tudunk segédváltozót feltételezve másolni, negálni, szorozni, összeadni, következésképpen a hasítás és az eltolás helyett ezt is használhatjuk.

Végül igazoljuk, hogy egyik utasítás sem valósít meg kölcsönösen egyértelmű leképezést.

Ha " e " az utasítások valamelyikét jelöli, legyen S olyan buborék-halmaz, $S \subseteq R$, amelyre $S^e \neq S$. Ekkor mindig létezik olyan $Z \subseteq R$ buborék-halmaz, amelyre $S \neq Z$, de $S^e = Z^e$. Az állítást bebizonyítottuk, ha minden utasítástípushoz megadjuk a konkrét előállítást:

1. $e = (a, b)$, $Z = (S - \{a\}) \cup \{b\}$
2. $e = (1, a)$ $Z = S \cup \{a\}$
3. $e = (0, a)$ $Z = S - \{a\}$
4. $e = (a, b)_h$ $Z = S \cup \{b\}$
5. $e = (a, b)_r$ $Z = (S - \{a\}) \cup \{b\}$
6. $e = (a, b)_c$ $Z = (S - \{a\}) \cup \{b\}$

Ugy is fogalmazhatjuk, hogy ezek a műveletek nem megfordíthatók, hiszen a kiinduló állapot az utasítás aktivizálása után nem minden esetben állítható vissza a végállapot ismeretében.

Ennek a ténynek a következménye, hogy nem lehet a fenti hat utasításból olyan programot összeállítani, amely segédrekeszek igénylése nélkül pl.

két szám összeadását megvalósítja úgy, hogy az egyik szám változatlan marad, a másik szám helyén pedig az összeg jelenik meg, mivel ez kölcsönösen egyértelmű leképezést jelentene. Valóban, ennek megvalósításához segédrekeszre is szükség van.

Végezetül megemlítjük, hogy az LF-ek redukálásához hasonlóan itt is felvethető, hogy egy adott utasításrendszer mellett hogyan lehet a legrövidebb idő alatt vagy legkevesebb utasítással előállítani egy LF-t. Itt nemcsak az fontos, hogy az egyes utasításokat hogyan sulyozzuk, hanem azt is figyelembe kell venni, hogy párhuzamosan egyszerre több művelet is elvégezhető. Pl. a 7. ábrán az a_{11} első oszlopbeli elemtől az a_{11} első sorbeli elemig huzott egyenes mentén valamennyi műveletet egyszerre el lehet végezni.

ÖSSZEFOGLALÁS

Jelenleg már több olyan utasítás valósítható meg a buborékmemóriában, amelyek együttesen lehetőséget nyújtanak tetszőleges logikai függvény reprezentálására, következésképpen lehetőség nyílt arra, hogy buborékmemóriából logikai elemeket építsünk.

A legegyszerűbben megvalósítható utasítás, az eltolás önmagában erre a célra nem lehet minden esetben alkalmas, de ez más utasításokkal a célnak megfelelően mindig kiegészíthető, vagy helyettesíthető.

IRODALOM

R.L.Graham: A mathematical study of a model of magnetic domain interactions, The Bell Syst.Techn.J., Vol.49.No.8. /1970/

A.D.Friedman, P.R.Menon: Mathematical models of computation using magnetic bubble interactions, The Bell Syst.Techn.J. Vol.50.No.6./1971/.

W.Kluge: Computationally complete operations on magnetic bubbles, Electron. Lett.Vol.7.pp.749-751. /1971/.

A.Ádám: Truth functions and the problem of their realization by two-terminal graphs, Akadémiai Kiadó, Budapest, 1968.

Bagyinszki J.: Véges automaták. A Matematikai Kutató Intézet "A számítástechnika matematikai alapjai" c. tanfolyamának jegyzete, Bp.1972.

62.013



Kiadja a Központi Fizikai Kutató Intézet
Felelős kiadó: Varga László, a KFKI
Számítástechnikai Tudományos Tanácsának
elnöke
Szakmai lektor: Bagyinszki János, Zimmer György
Példányszám: 105 Törzsszám: 73+8368
Készült a KFKI sokszorosító üzemében,
Budapest, 1973. június hó