tanulmányok

102/1980

MTA Számitástechnikai és Automatizálási Kutató Intézet Budapest

MAGYAR TUDOMÁNYOS AKADÉMIA
SZÁMITÁSTECHNIKAI ÉS AUTOMATIZÁLÁSI KUTATÓ INTÉZETE

REKURZÍV ELJÁRÁS

Boole-függvényrendszerek nemredundáns diszjunktiv normálformáit előállitó, idő- és memóriaigény szempontjából optimális algoritmus

Irta:

DR. PÁSZTORNÉ VARGA KATALIN

Tanulmányok 102/1980.

A kiadásért felelős:

DR VÁMOS TIBOR

ISBN 963 311 100 5
ISSN 0324-2951

8011075 MTA KESZ Sokszorosító, Budapest. F. v.: dr. Héczey Lászlóné

TARTALOMJEGYZÉK

ELŐSZÓ

A Boole-függvények minimális vagy nemredundáns normálformáinak előállitására igen sok eljárás ismert. A főbb eredmények a 60-as években születtek meg. Ezeket olyan algoritmusok kidolgozása követte, amelyek a minimalizálási probléma különböző szempontból való megközelitésén alapultak. Az algoritmusok kidolgozásával és számitógépes realizációjukkal egyidőben elkezdődött azok általánositása nem teljesen meghatározott függvényekre és függvényrendszerekre. A sokváltozós és nagyszámu függvényekből álló függvényrendszerek számitógépes kezelése olyan nagy tárkapacitást és gépidőt követelt, ami irreálissá tette a módszerek gyakorlati alkalmazását. Ezzel magyarázható, hogy a 70-es évek elejére az elméleti vizsgálatok ujra előtérbe kerültek és uj eredmények is születtek [8, 9]. Ezekre az jellemző, hogy összegezik a korábbi módszerek tapasztalatait és javitják az információkezelést.
Az általunk kidolgozott és realizált algoritmus felhasználja a kételemü halmaznak B-nek, a B n-szeres direkt szorzatának B^n-nek, valamint az összes $f:B^n \to B$ leképzések halmazának (amelyek Boole algebrák) tulajdonságait, egymással való kapcsolatait, valamint a B^n tér egy k változós ($k \leq n$) konjunkció által kijelölt részhalmazának speciális számossági, szimetria és hasonlósági tulajdonságait. Az eljárás a függvény összes primimplikánsát szolgáltató fát nem állitja elő (1. ábra), hanem egyszerre csak egy olyan ágát generálja, amely biztosan adja az f függvény egy p primimplikánsát, majd az $f \wedge \neg p$ függvényre ujra kezdődik egy primimplikánst adó ág generálása mindaddig, amig $f \wedge \neg p \equiv 0$ nem lesz. Az ut csucsaihoz rendelt részfüggvényeket az eljárás során csak annyira határozzuk meg, hogy a továbblépés irányát meghatározzuk (lásd 2,3 függelék). Az eljárás függvényrendszer szimultán egyszerüsitésére is alkalmas. A tanulmányban az algoritmus számitógépes realizációját is ismertetjük.

A REKURZÍV ELJÁRÁS HELYE A BOOLE-FÜGGVÉNYEK MINIMÁLIS VAGY NEMREDUNDÁNS DISZJUNKTÍV NORMÁLFORMÁINAK ELŐÁLLÍTÁSÁBAN

B E V E Z E T É S

Feltételezve a klasszikus egyszerüsitő-minimalizáló eljárások ismeretét csak rövid áttekintést adunk ezekről az eljárásokról és megmutatjuk a rekurziv eljárással való kapcsolatukat. Látni fogjuk, hogy a rekurziv eljárás figyelembe vesz minden információt, ami a Boole-függvények értelmezési tartományának speciális strukturájából és abból következik, hogy a DNF-ban szereplő részfüggvények - konjunkciók - az értelmezési tartomány olyan részhalmazainak karakterisztikus függvényei, amelyek strukturális szempontból az értelmezési tartománnyal megegyeznek. Ennek következtében a primimplikánsok előállitásának stratégiája eltér a korábbi módszerekétől. Ez azt jelenti, hogy egy munkafázisban vagy több (formailag közeli) primimplikánst állitunk elő, vagy direkt módon állitjuk elő a soronkövetkező primimplikánst a benne biztosan előforduló változókra vonatkozó feltételek teljesülésének megvizsgálásával. Megmutatjuk azt is, hogy az eljárás gépi realizálásánál a memóriaigény előre becsülhető és a végrehajtási idő jóval rövidebb, mint a klasszikus eljárásoknál.

STRUKTURÁLIS KÉRDÉSEK

Egy n változós Boole-függvény a B^n térnek a B térbe való leképezése. A $\{0,1\} = B$ tér Boole-algebra. A $\{0,1\}^n = B^n$ tér (B n-szeres direkt szorzata) szintén Boole-algebra. Nevezik a B^n teret n dimenziós (0,1) térnek, ahol a tér pontjaira azok koordináta n-esei alapján (0<1) értelmezve a szokásos rendezést [3,5] a B^n tér egy félig rendezett halmaz, amelyről könnyen belátható, hogy nemcsak háló, hanem a fenti Boole-algebra. A B^n teret nevezik még n dimenziós Boole-kockának is, amelyben ha az egy Hamming távolságu (szomszédos)

pontokat összekötjük és két pontot összekötő élt a fenti rendezésnek megfelelően irányitjuk, akkor az előbb emlitett háló hálódiagramját kapjuk.

Mivel egy n változós Boole-függvény az értelmezési tartomány egy részhalmazába tartozó pontokhoz 1-et rendel - ennek a részhalmaznak karakterisztikus függvénye -, ezért az n változós Boole-függvények és a B^n tér részhalmazai között kölcsönösen egyértelmü megfeleltetés hozható létre. Mint ismeretes, egy halmaz összes részhalmazainak halmaza a tartalmazás szerinti rendezésre Boole-algebrát alkot. Igy az n változós Boole-függvények halmaza is Boole-algebrát alkot ugyanezen rendezésre. Könnyen belátható, hogy a rendezési reláció alapján definiálható hálómüveletek megegyeznek a B, illetve B^n térben lévő hálómüveletekkel [3,5].

NÉHÁNY ELJÁRÁSTIPUS PRIMIMPLIKÁNSOK ELŐÁLLITÁSÁRA

A primimplikáns előállitő eljárások nem teljesen meghatározott Boole-függvények esetére is kiterjeszthetők, de ennek módját itt nem tárgyaljuk.

1. Quine, Mc Cluskey algoritmus [1,2]

   Az eljárás alapja a B^n tér azon pontjainak halmaza, amelyekhez a függvény az 1 értéket rendeli, röviden a függvény 1 pontjai vagy mintermjei (teljes elemi konjunkció).

   Az eljárás megkeresi az összes 1 Hamming távolságu pontpárt és a két pontot egy olyan $n-1$ komponensü 0,1 sorozattal irja le, amelyben az egyforma komponensek szerepelnek csak. Más szóval két teljes elemi konjunkcióra alkalmazza az

$$x_1^{\alpha_1} \ldots x_i^{\alpha_i} \ldots x_n^{\alpha_n} \vee x_1^{\alpha_1} \ldots x_i^{\bar{\alpha}_i} \ldots x_n =$$

$$= x_1^{\alpha_1} \ldots x_{i-1}^{\alpha_{i-1}} \; x_{i+1}^{\alpha_{i+1}} \ldots x_n^{\alpha_n}$$

egyszerüsitési szabályt. Az igy kapott $n-1$-esek halmazain belül ugyanezen eljárással már $4=2^2$, majd az i-edik lépésben 2^i... pontot foglal össze egy $n-i$ változót tartalmazó konjunkcióba, vagy $n-i$ komponensü 0,1 sorozatba. Ez az eljárás nagy és előre nem becsülhető memória- és időigénye miatt nagy változószám esetén igen nehézkes. Ebbe az eljárástipusba tartozik minden olyan eljárás, amely a függvény 1 pontjai alapján összevonással ("hizlalással") állitja elő a primimplikánsokat. Ilyen eljárások a Veitch diagram, Karnaugh-módszer, Harvard-módszer [3]. Ide tartozik a függvény bináris fáját felhasználó algoritmus is [7], amelyben a maximiális részfák felelnek meg a primimplikánsoknak.

2. Consensus-módszer [1,6]

Az eljárás alapja a függvény egy tetszőleges DNF-ja. A primimplikáns előállitásához az $ax \vee b\bar{x} = ax \vee b\bar{x} \vee ab$ bővitési azonosságot használják fel az $ax \vee a\bar{x} = a$ egyszerüsitési azonosság mellett.

Az eredményességet biztositja az a tétel, amely kimondja, hogy a függvény tetszőleges DNF-jából a consensus (bővitési) és az egyszerüsitési azonosság segitségével az összes primimplikáns előállitható.

3. Kipróbálás módszere [3,10]

Az eljárás alapja a függvény 1 és 0 pontjainak halmaza.

A primimplikánsok kereséséhez megvizsgáljuk az összes i változós ($i=1,2,\ldots,n$) konjunkciót i növekvő sorrendjében és eldöntjük, hogy implikálja-e a függvényt. A kiválasztás sorrendje miatt az implikáns konjunkciók primimplikánsok. Az eljárás memóriaigénye nagy, n változós függvényre 2^n nagyságrendü. A müveletigény előre nem becsülhető, mivel a próbálkozás akkor fejeződhet be, amikor a legtöbb változót

tartalmazó primimplikánst is megtaláltuk. Az egyes lépések müveletigénye felülről jól becsülhető. A k változós konjunkciók száma $\binom{n}{k}\cdot 2^k$. Ez a szám csak akkor csökken (minimálisan), ha már találtunk primimplikánst.

4. Topológiai módszerek [3,11]

Az eljárás alapja a függvény 1 és 0 pontjainak halmaza.

A primimplikáns keresésnél sorra vesszük a függvény 1 pontjait és olyan, az illető pontot is magába foglaló konjunkciót (konjunkciókat) keresünk, amelyek primimplikánsok. A kiválasztás alapja az a tény, hogy ha P egy olyan B^n térbeli pont, amely a függvénynek 1-pontja és k szomszédja van a függvény nem 0 pontjai között, akkor a ráilleszthető, a függvényt implikáló konjunkció változóinak száma legalább $n-k$ és legfeljebb $n-1$. A 3-as és 4-es módszer ötvözete a [12]-ben leirt eljárás.

5. A rekurziv módszer [8]

A függvény 1 és 0 pontjaiból egy rekurziós formulával nyeri az összes primimplikánst, vagy egy nemredundáns DNF-ban szereplő primimplikánsokat.

A REKURZIV MÓDSZER JAVITÁSA, A JAVITOTT MÓDSZER HATÉKONYSÁGA

A rekurziv módszert sikerült általánositani ugy, hogy alkalmas legyen függvényrendszer kezelésére is, és hogy a kiinduló adat tetszőleges DNF lehessen [4,9].

A módszer lényegét a 2. függelék tartalmazza. A 2. függelék (3) formulája segitségével felirható egy rekurziós formula, amelynek kifejtése az összes primimplikánst adja. Jelölje

A_i, B_i, C_i a (3) -ban szereplő részfüggvényeket, ha az i-edik változót (x_i-t) emeljük ki. Ekkor

$$f = A_i \vee x_i B_i \vee \bar{x}_i C_i \qquad (1)$$

Megjegyzés:

Tegyük fel, hogy f n változós függvény. Ekkor A_i, B_i, C_i $n-1$ változós függvények. Mivel az $n-1$ dimenziós Boole-tér egy pontja az n dimenziós térben két, az n-edik (i-edik) koordináta szerinti szomszédos pontot jelent, ugy is tekinthetjük, hogy az A_i, B_i, C_i függvények értelmezési tartománya a B^n tér i-edik koordinátában szomszédos pontpárjainak halmaza.

Megjegyzés:

Ha a kifejtést egy másik (x_j) változó szerint folytatjuk, akkor szükség lesz a $\neg A_i, \neg B_i, \neg C_i$ függvényekre. Be lehet látni, hogy (1. ábra)

- ha F_{i-1} teljesen meghatározott függvény, akkor A_i teljesen meghatározott függvény, de B_i és C_i nem teljesen meghatározott függvények, melyeknek közös értelmezetlenségi tartománya A_i, vagyis B_i és C_i az 1 pontokat adják meg. Mivel egy $\{f_1, f_o\}$ ϕ-Boole-függvény negáltján az f_o függvényt értjük, ezért

1. $$\neg A_i = \neg(F_{i-1})_{x_i=1} \vee \neg(F_{i-1})_{x_i=o}$$

2. $$\neg B_i = \neg(F_{i-1})_{x_i=1} = ([\neg(F_{i-1})_{x_i=1} \vee \neg(\neg F_{i-1})_{x_i=o}] \wedge$$

$$\underbrace{[\neg(F_{i-1})_{x_i=1} \vee \neg(F_{i-1})_{x_i=o}]}_{\neg A_i}$$

3. $\neg C_i = \neg(F_{i-1})_{x_i=o}$

- ha F_{i-1} ϕ-Boole-függvény $(F_{i-1}=(F_{i-1})_1)$, akkor az A_i nem teljesen meghatározott függvény és

$$A_i = (F_{i-1})_{x_i=1}\,(F_{i-1})_{x_i=o} \vee (F_{i-1})_{x_i=1}\,(F_{i-1})_{\phi,x_i=o}$$

$$(F_{i-1})_{x_i=o}\,(F_{i-1})_{\phi,x_i=1}; \quad A_{i_o} = (F_{i-1})_o$$

B_i és C_i értelmezetlenségi tartománya pedig A_i, ezért ebben az esetben

1. $\neg A_i = (F_{i-1})_{o,x_i=1} \vee (F_{i-1})_{o,x_i=1}$

$$B_i = (F_{i-1})_{1,x_i=1} \wedge (F_{i-1})_{o,x_i=o}$$

2. $\neg B_i = (F_{i-1})_{o,x_i=1} \vee (F_{i-1})_{o,x_i=o} \wedge (F_{i-1})_{\phi,x_i=1}$

3. $\neg C_i = (F_{i-1})_{o,x_{i=o}} \vee (F_{i-1})_{o,x_i=1}\,(F_{i-1})_{\phi,x_i=o}$

Jelölje $P_i(f)$ az f függvény összes primimplikánsát, akkor

$$P_i(f) = P_i(A_i) \vee x_i P_i(B_i) \vee \bar{x}_i P_i(C_i) \qquad (2)$$

és

$$P_i(O) = O \qquad (3a)$$

$$P_i(1) = 1 \qquad (3b)$$

$$P_i(f) = 1, \quad \text{ha} \quad \bar{f} \equiv O \qquad (3c)$$

Ezáltal az összes primimplikánst leiró rekurziv formulához jutunk.

P é l d a:

Legyen $f=x_1\bar{x}_2x_3 \vee x_1\bar{x}_2\bar{x}_3x_4 \vee x_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1\bar{x}_2\bar{x}_4$

$\bar{f}=x_2 \vee \bar{x}_1x_4$

Fejtsük ki a rekurziv formulát a változók indexei szerinti sorrendben.

$$P_i(f) = P_i(\underbrace{\bar{x}_2x_3\bar{x}_4 \vee \bar{x}_2\bar{x}_3\bar{x}_4}_{A_1}) \vee x_1(P_i(\underbrace{\bar{x}_2x_3x_4 \vee \bar{x}_2\bar{x}_3}_{B_1}x_4)) \vee \bar{x}_1(P_i\underbrace{(0)}_{C_1})$$

$$\neg A_1 = x_2 \vee x_4, \qquad \neg B_1 = x_2$$

$$P_i(f) = [P_i(0) \vee x_2P_i(0) \vee \bar{x}_2P_i(\underbrace{x_3\bar{x}_4 \vee \bar{x}_3\bar{x}_4}_{C_2^A})] \vee$$

$$x_1[P_i(0) \vee x_2P_i(0) \vee \bar{x}_2P_i(\underbrace{x_3x_4 \vee \bar{x}_3x_4}_{C_2^B})] \vee \bar{x}_1 \cdot 0$$

$$\neg C_2^A = x_4, \qquad \neg C_2^B \equiv 0$$

A (3c) miatt, mivel $\neg C_2^B = 0$, ezért $P_i(C_2^B) = 1$.

Alkalmazva még a (3a)-t is:

$$P_i(f) = [\bar{x}_2(P_i\underbrace{(\bar{x}_4)}_{A_3} \vee x_3P_i(0) \vee \bar{x}_3P_i(0))] \vee x_1\bar{x}_2$$

$$\neg A_3 = x_4$$

$$P_i(f) = \bar{x}_2(P_i(0) \vee x_4 P_i(0) \vee \bar{x}_4 P_i(1) \vee x_1\bar{x}_2$$

$$P_i(f) = \bar{x}_2\bar{x}_4 \vee x_1\bar{x}_2$$

Ha a rekurziós formula formulagráfját megszerkesztjük, akkor egy olyan gráfot kapunk (1. ábra), amelynek minden csucsából három él indul ki. Azokhoz a csucsokhoz, amelyekhez ezek az élek vezetnek, olyan A_i, B_i, C_i függvények tartoznak, amelyekre $F_{i-1} = A_i \vee x_i B_i \vee \bar{x}_i C_i$. F_{i-1} az előttük lévő csucshoz tartozó függvény (A_{i-1} vagy B_{i-1} vagy C_{i-1} tipusu vagy az eredeti $F=F_o$). A_i, B_i, C_i rendre olyan függvények, amelyek nem függnek az i-edik változótól, tehát primimplikánsaikban nem szerepel az i-edik változó. A formulából látható, hogy A_i primimplikánsai F_{i-1} -nek primimplikánsai, de B_i primimplikánsainak az i-edik változóval, C_i primimplikánsainak az i-edik változó negáltjával kell a konjunkcióját képezni, hogy F_{i-1} primimplikánsait megkapjuk. A fa (gráf) utjainak utolsó eleme az azonosan O vagy 1 függvény. Ha a gráf i-edik szintjén lévő éleihez 1, x_i, $\bar{x}_i$-at rendelünk aszerint, hogy A_i, B_i vagy C_i tipusu függvényhez vezet, akkor a gráf 1-el lezáruló utjain szereplő változók konjunkciói az összes primimplikánst megadják.

A módszer hatékonysága az eljárás tulajdonságai alapján világos. Vagyis

- A fa előállitó algoritmus automatikusan befejeződik.
- A fa konstrukciója egyszerü és egy listastrukturával a teljes fa tárolható.
- A primimplikánsok a fa bejárásával megkaphatók.
- Függvényrendszer együttes primimplikánsainak előállitása lényegileg ezzel a módszerrel, megfelelő indexezési technikával megvalósitható (lásd 3. függelék).

- Ha csak nemredundáns lefedést akarunk, akkor nem kell a teljes fát felépiteni, hanem egyszerre csak egy primimplikánst meghatározó utat.
- Egy ut felépitése egyszerü és gyors, tárolni csak a következő lépéshez szükséges adatokat kell. Ezért egy primimplikáns előállitása után a primimplikánssal módositott (le nem fedett) függvényre ujrakezdjük egy ut generálását.

A rekurziv eljárás realizálása, mint az a 2. és 3. függelékben látható az (1) kifejezéshez kapcsolódó megjegyzésen alapszik.

A_i, B_i, C_i előállitásakor $(F_{i-1})_1$ és $(F_{i-1})_0$ DNF-jából elhagyjuk az x_i változót. Az igy kapott

$$(F_{i-1})_1^{x_i} \quad , \quad (F_{i-1})_0^{x_i}$$

értelmezési tartománya x_i-ben szomszédos pontpárokból (ponthalmaz párokból) áll. Más szóval mind az 1, mind a 0 pontokhoz hozzácsatoljuk azok x_i szerinti szomszédjait is.

- A_i a 0 pontot nem tartalmazó pontpárok halmaza
- $B_i(C_i)$ az olyan pontpárok halmaza, amelyekben az egyik pont 1 pont, a másik 0 pont és az 1 pontot leiró konjunkcióban $x_i(\bar{x}_i)$ szerepel.

Világos, hogy A_i adja azon pontok halmazát, ahol x_i kiegyszerüsithető, B_i és C_i pedig azokét, ahol x_i illetve $\bar{x}_i$ nem egyszerüsithető ki.

Összehasonlitva a Quine-Mc Cluskey eljárással, itt egy lépésben választhatók szét azok a pontok, amelyek x_i szerint összevonhatók és amelyek nem.

A keresési stratégia szempontjából nézve a két eljárást, a rekurziv eljárás a változók szerinti soros (szekvenciális), a Quine-Mc Cluskey eljárás párhuzamos technikát alkalmaz. Az 1. ábrán a rekurziv eljárást reprezentáló fa utjain kapjuk meg a változók egymásutáni vizsgálatával egy-egy primimplikánst.

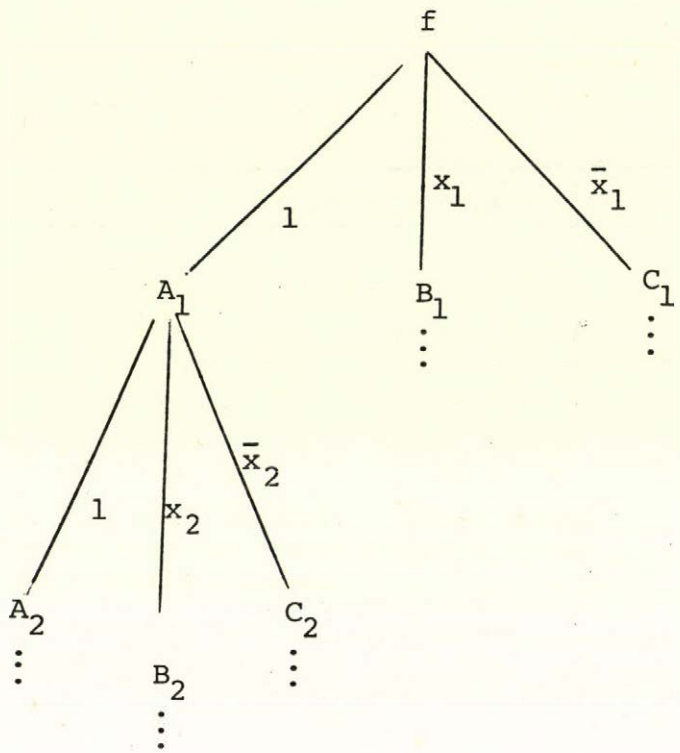
Jelölje A_{ij} $(j=j,2,\ldots,n)$ azt a pontpárhalmazt, ahol mindkét pont F_{i-1}-nek nem 0 pontja, tehát x_j kiegyszerüsithető és B_{ik} azt a pontpárhalmazt, amelyből az F_{i-1} -nek egyetlen változója sem egyszerüsithető ki (k az F_{i-1}-változóinak száma). Ezzel a jelöléssel a 2. ábra a Quine-Mc Cluskey algoritmus stratégiáját leiró fát adja. Itt adott változószámu primimplikánsok a fa egy szintjén állnak elő az összes változó együttes vizsgálatának eredményeképp.

A rekurziv eljárás szekvenciális keresési stratégiája nemcsak rugalmasabb és egyszerüsités orientált, hanem a keresett primimplikánsokra vonatkozó szempontok is figyelembe vehetők az eljárás során, ha ez szükséges.

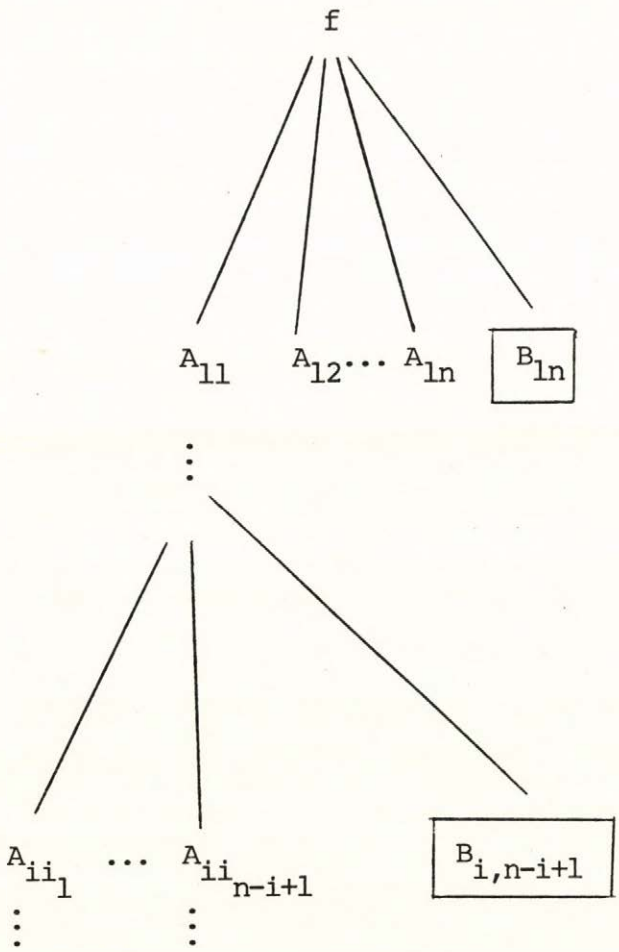
Illusztrációként néhány adat a módszer kis időigényének illusztrálására.

	Változók száma	Függvények száma	Hagyományos módszer		Rekurziv módszer
			minimalizálás	egyszerüsités	
1	8	3	15′ kevés	4′ 07.720	3′ 25.817
2	4	3	0′ 04.997	0′ 03.682	0′ 02.054
3	7	3	6′ 30.375	1′ 27.317	0′ 12.213
4	10	1	12′ 21.331	2′ 49.600	0′ 18.840
5	4	1	0′ 10.254	0′ 09.327	0′ 01.302

1. táblázat



1. ábra



2. ábra

2. AZ E ELJÁRÁS

PROGRAMRENDSZER NEM TELJESEN MEGHATÁROZOTT (ϕ) BOOLE-FÜGGVÉNY NEMREDUNDÁNS DISZJUNKTÍV NORMÁLFORMÁJÁNAK ELŐÁLLÍTÁSÁRA

Az algoritmus közvetlenül szolgáltatja a függvény egy nemredundáns (nem feltétlenül minimális) diszjunktiv normálformáját (DNF-jét) [4] az egyszerüsitendő függvény f_1 és f_o részfüggvényei tetszőleges DNF-jei alapján (lásd. 1. Függelék).

Az algoritmus igen gyors, a korábbi - Quine-Mc Cluskey [1,2], topológiai [3] módszereknél mintegy tizszer gyorsabb. Az algoritmus végrehajtása során kapott részeredmények terjedelme a kiinduló adatok alapján jól becsülhető - mintegy háromszorosa annak. Egy primimplikáns meghatározásához szükséges müveletszám lényegében a függvényben szereplő diszjunkciós tagok számától függ. Az algoritmus a [4]-ben leirt rekurziv módszer további egyszerüsitése (lásd a 2. Függelék).

A programban realizált algoritmus leirásában használt néhány jelölés [4]-vel összhangban:

P-vel jelöljük f_1 eredeti DNF-jében szereplő konjunkciók listáját

Q-val jelöljük f_o eredeti DNF-jében szereplő konjunkciók listáját

P_t illetve Q_t jelöli a P-ből, illetve Q-ból kapott részeredmény függvény DNF-jében szereplő konjunkciók listáját

P/Q-val jelöljük a P és Q listák egymásután irásával kapott konjunkciólistát

$R_{x_i}(P/Q)$-jelöli a redukció-operátor x_i változó szerinti alkalmazását

$I_{x_i^\alpha}(P/Q) = P_{x_i^\alpha}/Q_{x_i^\alpha}$ - jelöli az interszekció-operátor alkalmazását x_i szerint

$C_v(P/Q)$ - jelöli a lefedés-operátor alkalmazását egy adott v konjunkció mellett.

AZ ALGORITMUS LEIRÁSA:

Az algoritmusban P_j/Q_j jelöli mind a részeredményként kapott P_t/Q_t, mind az eredeti P/Q listát (P/Q mindig P_o/Q_o-nak felel meg).

1. Az $f(x_1,x_2,\ldots,x_n) = (f_1,f_o)$ felirása $P/Q = P_o/Q_o$ alakban
2. $j = 0$
3. $i = 0$, $P = 1$
4. $i = i + 1$
5. $I_{x_i}(P_j/Q_j)$ és $I_{\bar{x}_i}(P_j/Q_j)$ előállitása
6. Ha mind P_{jx_i}, mind $P_{j\bar{x}_i}$ üres, akkor a 15-dik lépés következik.
7. Ha P_{jx_i} és $Q_{j\bar{x}_i}$ metszete nem üres, akkor $\alpha=1$. A 10-dik lépés következik.
8. Ha $P_{j\bar{x}_i}$ és Q_{jx_i} metszete üres, akkor a 15-dik lépés következik.
9. $\alpha = 0$
10. x_i^α -t a primimplikáns soronkövetkező tényezőjeként tároljuk $p = p \,\&\, x_i^\alpha$
11. Ha $Q_{jx_i^\alpha} \neq 0$, vagyis p még nem primimplikáns, akkor $k=j$, $j=1$, $P_j/Q_j = P_{jx_i^\alpha}/Q_{jx_i^\alpha}$ a 4-dik lépés következik.

12. (p-rimimplikáns) $j = 0$, $P_j/Q_j = C_p(P_0/Q_0)$

13. Ha P_0 üres, akkor kiírjuk a primimplikánsokat. Az algoritmus befejeződik.

14. A 2-dik lépés következik.

15. $k = j$, $j = 1$, $P_j/Q_j = R_{x_i}(P_k/Q_k)$, 4-dik lépés következik.

/Ld. az 1. sz. mellékletet/

GÉPI REPREZENTÁCIÓ

Általános megjegyzések

- a függvény i-dik változójához a 2^{i-1} belső reprezentáció tartozik
- egy konjunkció gépi reprezentációját két egész tipusu szóban adjuk meg.

Az első szóban (első komponensben) a konjunkcióban negálatlanul szereplő változók 2^k alaku belső reprezentációinak összege szerepel. A második szóban (második komponensben) a konjunkcióban szereplő változók 2^k alaku belső reprezentációinak összege áll.

Például legyenek x,y,z,w egy függvény változói, ugy hogy a függvény változói közül az első az y, a második a w, a harmadik az x, a negyedik a z, akkor

- e változók belső reprezentációi

  x - 4
  y - 1
  z - 8
  w - 2

- az $x\bar{y}$ konjunkció gépi reprezentációja
  1. komponens 0100 - 4
  2. komponens 0101 - 5

Az ábrázolásmódból következik, hogy a programrendszerrel kezelhető minden n változós Boole-függvény, ahol $n \leq$ szóhossz-1 (a CDC-re $n \leq 23$).

Az algoritmust realizáló program

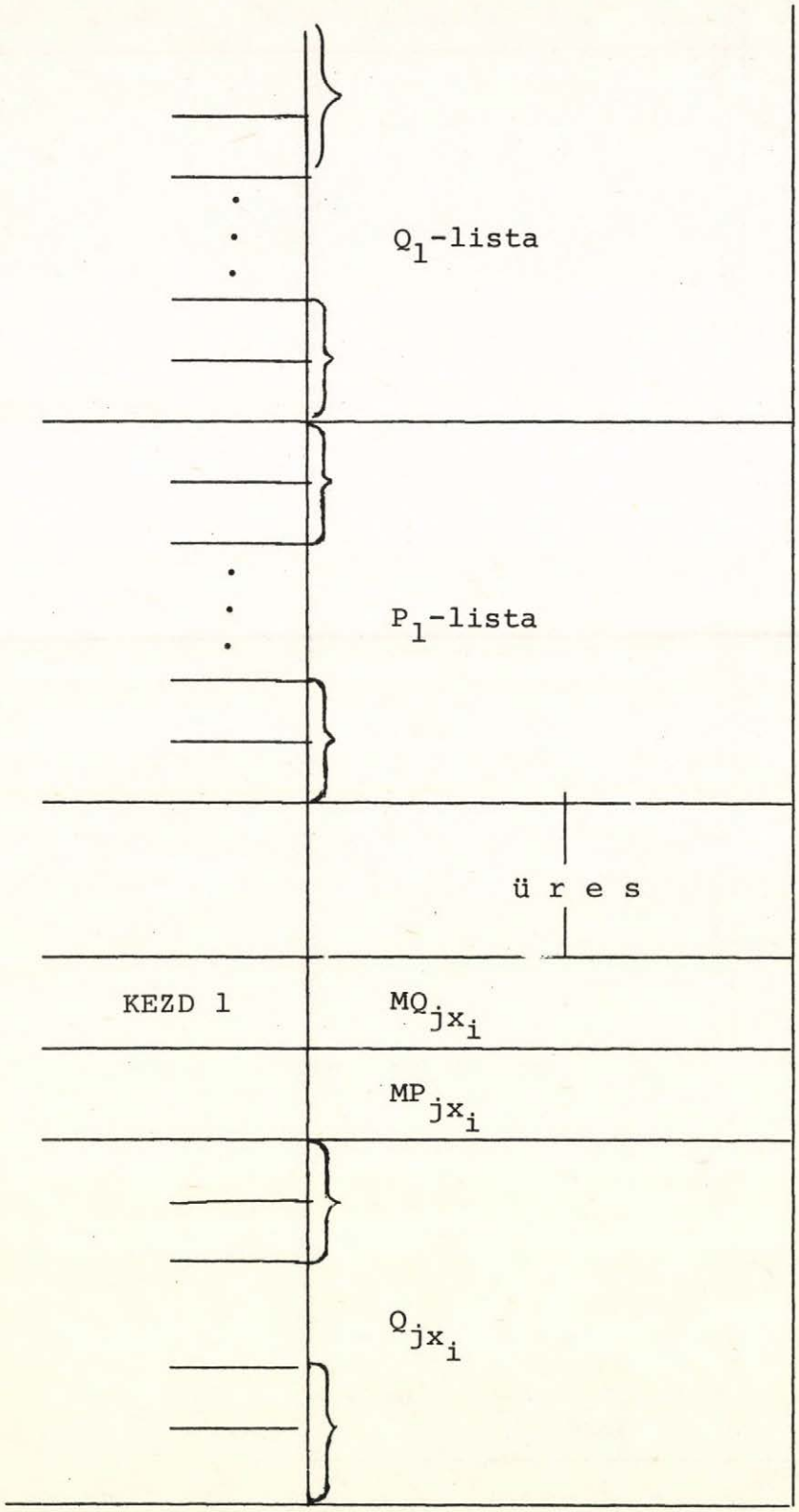
- A függvény adatait képező P/Q konjunkciólistát és a részeredményként kapott P_i/Q_i konjunkciólistákat egyetlen előre definiált méretü (MERET) tömbben

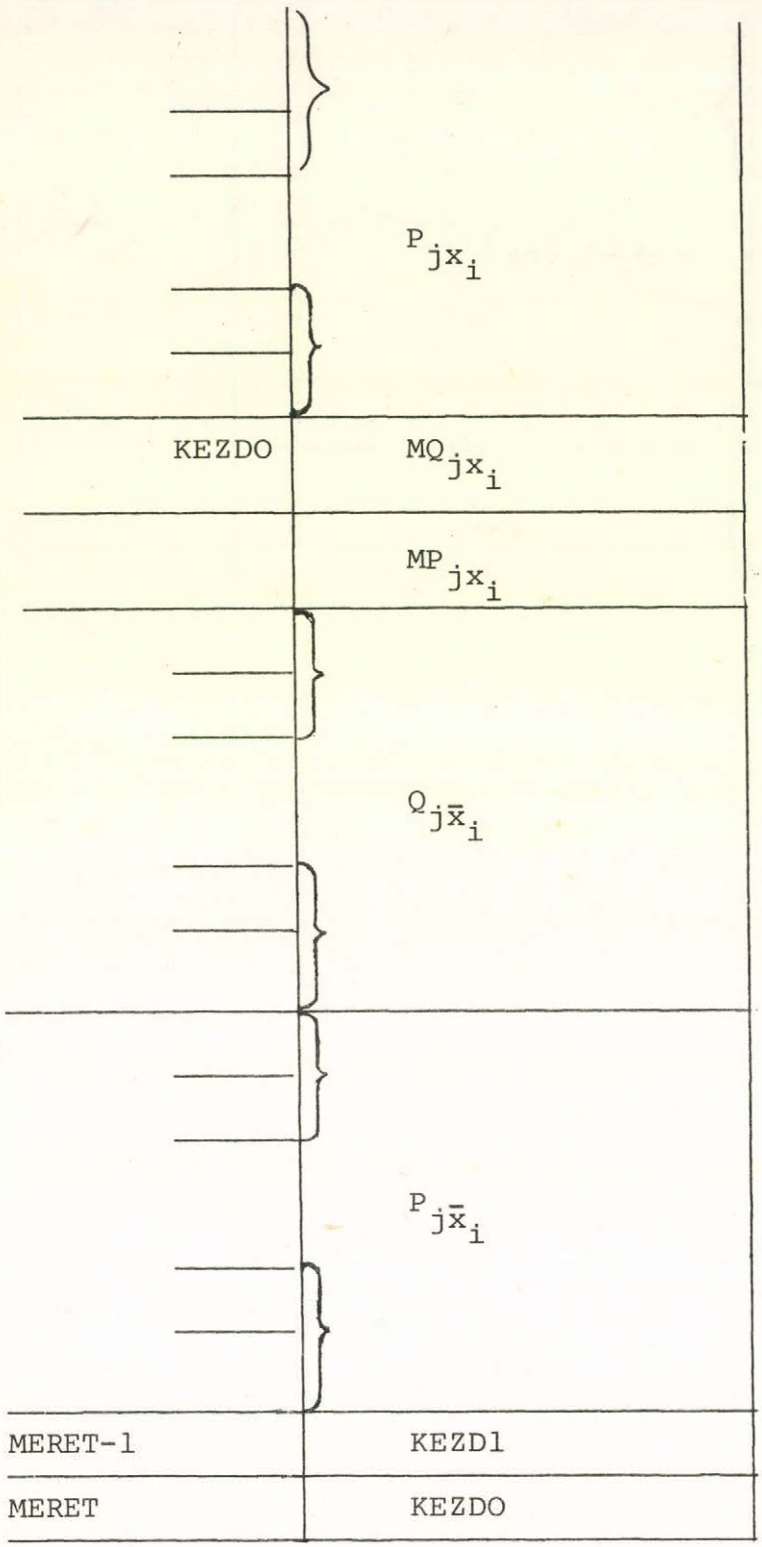
(MT-ben) helyezzük el az alábbi szervezés szerint:

MT i	
1	változók száma
2	$MQ_o=2K_{Q_o}$
3	$MP_o=2K_{P_o}$
4	Q_o-lista
5	
	(f_o konjunkciói)
MQ+3	
MQ+4	P_o-lista
MQ+5	
	(f_1 konjunkciói)
MQ+MP+3	
	$MQ_1=2K_{Q_1}$
	$MP_1=2K_{P_1}$

$K_Q=f_o$ konjunkcióinak száma

$K_P=f_1$ konjunkcióinak száma


Q_1-lista
P_1-lista
ü r e s
KEZD 1
MQ_{jx_i}
MP_{jx_i}
Q_{jx_i}


P_{jx_i}
KEZDO
MQ_{jx_i}
MP_{jx_i}
$Q_{j\bar{x}_i}$
$P_{j\bar{x}_i}$
MERET-1
KEZD1
MERET
KEZDO

- Az algoritmus leirásánál felsorolt R_{x_i}, $I_{\alpha x_i}$, C_v operátorok alkalmazását szubrutinok realizálják. Külön szubrutin végzi a $P_{jx_i}^{\alpha}$ & $Q_{jx_i}^{\bar{\alpha}} = 0$ vizsgálatát.

Az elmondottakból következik, hogy az eljárás programjának elvi blokkdiagramja nem különbözik az algoritmusétól.

- Az f_1, f_0 függvényeket tetszőleges DNF-jükkel kell megadni.
- A beolvasó program a formula szintaktikai vizsgálata után előállitja a konjunkciók belső reprezentációinak listáját.
- Az elhelyező program pedig a P_0/Q_0 MT-be való elhelyezését végzi el. MT iniciálása.
- Ezután a rekurziv eljárást megvalósitó program megkeres egy primimplikáns halmazt, amely nemredundáns módon lefedi f-et.
- A program szerkezete:

- A program lényeges szubrutinjai:

SUBROUTINE ELHDNF - MT iniciálása

SUBROUTINE PQREX - $\hat{R}$ operátor végrehajtása

SUBROUTINE PMET - $P_{x_i^\alpha}$

SUBROUTINE QMET - $Q_{x_i^\alpha}$

} előállitása $I_{x_i^\alpha}$ végrehajtásához

SUBROUTINE PQURES - $P_i \frown Q_j = 0$ eldöntése

SUBROUTINE LEFED - C_p operátor végrehajtása

SUBROUTINE REKEF - az algoritmus végrehajtását szervező program

SUBROUTINE ELHDNF

Adott a *K1*, *K2* tömb, ahol *K1(I)* és *K2(I)* egy konjunkció belső reprezentációja. IV a változók száma. *NF(3,I)= 1* vagy *0* aszerint, hogy az I-edik formula egy függvény 1-helyeit vagy 0-helyeit irja-e le. *NF(6,I)* és *NF(7,I)* az I-edik formula első konjunkciójának, illetve utolsó konjunkciójának cime *K1*, *K2*-ben.

A szubrutin egy nem teljesen meghatározott függvény 1-helyeit és 0-helyeit megadó két formula konjunkcióit *K1*, *K2*-ből átirja az MT tömbbe a következő módon. A függvény 0-helyeit leiró formula konjunkcióit konjunkciónként (komponenspáronként) átirja MT-be az MT(4)-től folyamatosan. A függvény 1-helyeit leiró formula konjunkcióit pedig ugyanigy ezek után folyamatosan helyezi el.

MT(1)-be beirja a változók számát, MT(2)-be és MT(3)-ba a 0-helyek illetve az 1-helyek konjunkciói számára szükséges szavak számát.

Ha a függvényt negálni kell, akkor felcseréli a *0* és az *1*-helyeket leiró formulákat.

KÖZÖS ADATMEZŐK:

```
COMMON/MUNK/ N9,IV,IPE,IPO,LF,MPR,KP,KO,M2,N,NMA1,K3,IS,L1,M,L,
             NF/7,23/,JV/200/,MUV,MOD,BIT/23/,FULBIT,WORDL
COMMON/55/ MT/9900/,K1/3000/, K2/3000/, JVB/600/
COMMON/TABL/ IR/6,100/
COMMON/JELEK/ MF/45/
COMMON/HELY/ IH/24/
```

/Ld. a 2. sz. mellékletet/

SUBROUTINE PQREX /PQ,PQU,MPQ,I,JP/

A szubrutin a *PQ* tömbben lévő adatokon sorpáronként elvégzi az R_{x_i} operációt, vagyis törli az x_i-nek megfelelő oszlopot. Az eredményt a PQU tömbbe teszi el, ugy hogy az egyforma sorpárokat csak egyszer helyezi el.

BEMENŐ PARAMPTEREK:

PQ - az adattömb
PQU - az eredmény elhelyezésére szolgáló tömb
MPQ - a PQ tömb mérete
I - az x_i-nek megfeleltetett helyérték

KIMENŐ PARAMÉTER:

JP - a PQU sorpárjainak száma

KÖZÖS TERÜLETEK:

```
COMMON/MUNK/ N9,IV,IPE,IPO,LF,MPR,KP,KO,M2,N,NMA1,K3,IS,L1,M,L,
             NF/7,23/,JV/200/,MUV,MOD,BIT/23/,FULBIT,WORDL
COMMON/HELY/ IH/24/
```

/Ld. a 3. sz. mellékletet/

SUBROUTINE PMET/P,PU,MP,I,ALFA,IK/

Metszés az x_i^{α} változó szerint.

A szubrutin a *P* tömbből ugy állitja elő a *PU* tömböt, hogy P azon sorpárjait teszi át PU-ba, amelyekben a második sor I-edik helyértékén 1-es és az első sor I-edik helyértékén α áll. Az átirt sorpárokból törli az I-edik helyértéken lévő információt.

A müvelet során kapott egyforma konjunkciókat összevonja.

KÖZÖS TERÜLETEK:

```
COMMON/MUNK/ N9,IV,IPE,IPO,LF,MPR,KP,KO,M2,N,NMA1,K3,IS,L1,
             M,L, NF/7,23/,JV/200/,MUV,MOD,BIT/23/,FULBIT,WORDL
COMMON/HELY/ IH/24/
```

BEMENŐ PARAMÉTEREK:

P - az a tömb, amelyen a metszés müveletet végrehajtjuk
PU - a metszés eredményét tartalmazó tömb
MP - a P és PU mérete
I - az x_i-nek megfelelő helyérték (a metszeni és törölni kivánt oszlop sorszáma)
ALFA - az α értéke

KIMENŐ PARAMÉTER:

IK - PU sorpárjainak száma

/Ld. a 4. sz. mellékletet/

SUBROUTINE QMET/Q,QU,MQ,I,ALFA,IK/

Metszés az x_i^{α} változó szerint.

A szubrutin a *Q* tömbből ugy állitja elő a *QU* tömböt, hogy Q azon sorpárjait teszi át QU-ba, amelyekben vagy a második sor I-edik helyértékén 0 áll, vagy a második sor I-edik helyértékén 1 és az első sor I-edik helyértékén α áll. Az átirt sorpárokból törli az I-edik helyértéken lévő információt.

A müvelet során kapott egyforma konjunkciókat összevonja.

KÖZÖS TERÜLETEK:

```
COMMON/MUNK/ N9,IV,IPE,IPO,LF,MPR,KP,KO,M2,N,NMA1,K3,IS,L1,M,L,
             NF/7,23/,JV/200/,MUV,MOD,BIT/23/,FULBIT,WORDL
COMMON/HELY/ IH/24/
```

BEMENŐ PARAMÉTEREK:

Q - az a tömb, amelyen a metszés müveletet végrehajtjuk
QU - a metszés eredményét tartalmazó tömb
MQ - a Q és QU mérete
I - az x_i-nek megfelelő helyérték (a metszeni és törölni kivánt oszlop sorszáma)
ALFA - az α értéke

KIMENŐ PARAMÉTER:

IK - *QU* sorpárjainak száma

/ld. az 5. sz. mellékletet/

SUBROUTINE PQURES /P, Q, MP, MQ, KI/

A szubrutin eldönti, hogy a *P* tömbben lévő *P1*, *P2* sorpárok és a *Q* tömbben lévő *Q1*, *Q2* sorpárok között van-e olyan, hogy

$$F2 = Q2 \wedge P2\text{-re} \quad F2 \wedge P1 = F2 \wedge Q1.$$

KÖZÖS TERÜLETEK:

```
COMMON/MUNK/  N9,IV,IPE,IPO,LF,MPR,KP,KO,M2,N,NMA1,K3,IS,L1,M,L,
              NF/7,23/,JV/200/,MUV,MOD,BIT/23/,FULBIT,WORDL
COMMON/HELY/ IH/24/
```

BEMENŐ PARAMÉTEREK:

P - az adattömb
Q - az adattömb
MP - a *P* tömb mérete
MQ - a *Q* tömb mérete

KIMENŐ PARAMÉTER:

KI - a sorpárok száma

/ld. a 6. sz. mellékletet/

SUBROUTINE LEFED /P,PU,MP,MPU,K10,K20,JP/

A *P* tömbben egy *K1*, *K2* sorpár dominálo hatását regisztrálja. Ez a következőt jelenti:

- törli (nem irja át *PU*-ba) azon *P*-beli *P1*, *P2* sorpárokat, amelyekre

$$K2 \wedge P2 = K2 \quad \text{és}$$
$$K2 \wedge P1 = K1$$

(*P2* és *K2* egy-egy konjunkció 2. komponense,
P1 és *K1* egy-egy konjunkció 1. komponense)

- Ha $K2 \wedge P2 \neq K2$, de $M2 \wedge P1 = M2 \wedge K1$, akkor *P1* és *P2*-ből annyi uj sorpárt állit elő és ir át *PU*-ba, ahány pozición *K2=1* és *P2=0*. Sorravesszük ezeket a pozíciókat és P1, P2-ből az éppen soronkövetkező pozición végrehajtott változtatással előálló sorpárt átirjuk PU-ba. A változtatás a következő.
A megfelelő pozicióra P1-be a K1 ezen pozicióján lévő információ negáltja, P2-be pedig 1-es kerül.

KÖZÖS ADATMEZŐK:

```
COMMON/MUNK/ N9,IV,IPE,IPO,LF,MPR,KP,KO,M2,N,NMA1,K3,IS,L1,M,L,
             NF/7,23/,JV/200/,MUV,MOD,BIT/23/,FULBIT,WORDL
COMMON/HELY/ IH/24/
```

BEMENŐ PARAMÉTEREK:

P - az a tömb, amelyen a lefedés müveletet végrehajtjuk
PU - a lefedés eredményét tartalmazó tömb
MP - a *P* tömb mérete
MPU - a *PU* tömb mérete
K10 - a konjunkció 1. komponense
K20 - a konjunkció 2. komponense

KIMENŐ PARAMÉTER:

JP - a PU sorpárjainak száma

/Ld. a 7. sz. mellékletet/

SUBROUTINE REKEF

Az algoritmus végrehajtását vezérli. A szubrutin felépitése megegyezik a leirás elején megadott algoritmussal, mivel az egyes operációkat önálló szubrutinok végzik. A paraméterátadás a közös adatmezőn történik. A memóriafelhasználás gazdaságosságát a FORTRAN keretei között lehetséges dinamikus index kezeléssel biztositja a program. Ez azt jelenti, hogy ha az egyszerüsitendő függvény elhelyezéséhez N szó szükséges, akkor az egyszerüsitő eljáráshoz szükséges memóriaterület 4N szó.

KÖZÖS ADATMEZŐK:

```
COMMON/MUNK/ N9,IV,IPE,IPO,LF,MPR,KP,KO,M2,N,NMA1,K3,IS,L1,M,
            L1,NF/7,23/,JV/200/,MUV,MOD,BIT/23/,FULBIT,WORDL
COMMON/55/ MT/9900/, K1/3000/, K2/3000/, JVB/600/
COMMON/HELY/ IH/24/
```

/ld. a 8. sz. mellékletet/

ILLUSZTRÁCIÓK A PROGRAM EREDMÉNYÉRE

```
 LMOD = 0
F=-x1.-x2.-x3.-x4.-x5.-x6.-x7.-x8.-x9+-x .-x2.-x3.x4.-x5 x6.-x7.x8.x -+-x
1.-x2.-x3.x4.x5.x6.-x7.-x8.x9+-x .-x2.x3.-x4.x5.-x6.-x7.-x8.-x9+-x1.x2.-
x3.-x4.x5.-x6.x7.x8.-x9+-x .x2.-x3.x4.-x5.-x6.-x7.x8.-x9.-x1.x2.x3.-x4.x
.x4.x7.-x .x9+x1.-x2.- .-x4.-x5.-x6.-x .x8.-x9+x1.-x2. x3.x4.-x5.x .-x
7.x .x9+x1.-x2.x3.-x4.x .-x6.-x7.-x8.x +x1.-x2.x3.x4.x5. x6.x7.-x8.x +x1
.x2.-x3.x4.-x5.x6.x7.x8 x9+x1.x2.-x3.x4.x5.-x6.x7.-x .-x +x1.x2.x3.x .x5
.-x4.-x7.-x .x9.
-F=-x .-x2.-x3.-x4.-x5.-x6.x7.-x .x9+-x1.-x .-x3.-x4.x5. 6.x7.-x .-x +-x
 .-x .x3.x4.-x5.-x6.x7. .-x9+-x .-x2.x3. .x5.x6.x7.-x8.x +-x1.x2.-x3.x
4.x5.x6.-x7.x8.x +-x1.x , x .-x4.x5.-x6.-x7.-x .-x9+-x1.x .x3.x4.-x .x6.-
x7.x8.-x9+x .-x2.-x3.-x .x .x6.-x7.-x8.x +x .-x2.-x3.-x4.x5.x .x7.-x .x
+x1.x2.-x3.x4.-x5.x6.-x .x .-x9+x1.x2.x3.-x4.x .x6.-x7.-x8.x +x1.x2.x3.x
4.-x5.x6.x7.-x8.-x9*
```

```
     PRIMIMPL.
1.      x .-x3.x7
2.      -x .-x2.-x7
3.      -x .x2.-x4.x
4.      x .x4.x5
5.      -x .x2.-x .-x5
6.      x .-x2.-x5
7.      x .-x2.x3
```

(Az 1. táblázat 4-dik sorában lévő függvény.)

```
 LMOD = 0
F=-Y.-Z.-U+-X.-Z.-U,
-F=X.Y.-Z+U.Y.X,
*


     PRIMIMPL.
1.       Y.X
```

(Az 1. táblázat 5-dik sorában lévő függvény.)

3. A T ELJÁRÁS

PROGRAMRENDSZER NEM TELJESEN MEGHATÁROZOTT (Φ) BOOLE-FÜGGVÉNYEK EGYÜTTES NEMREDUNDÁNS DISZJUNKTÍV NORMÁLFORMÁJÁNAK ELŐÁLLÍTÁSÁRA

Az algoritmus közvetlenül szolgáltatja egy nemredundáns (nem feltétlenül minimális) diszjunktiv normálformáját a primimplikánsok függvényekhez való tartozását is feltüntető primimplikánlista alakjában az egyszerüsitendő függvények f_{i_1}, f_{i_o} részfüggvényei tetszőleges DNF-jei alapján.

A módszer a [4]-ben leirt eljárás továbbfejlesztése (lásd 3. Függelék).

Az algoritmus végrehajtásához szükséges idő nem függ a függvényrendszer elemszámától, csak a konjunkciók számától. Ez az idő átlagosan kétszerese azonos konjunkciószám mellett az E eljárás végrehajtásához szükséges időnek.

A programban realizált algoritmus leirásában használt jelölések [4]-gyel összhangban és az E eljárásbeliekkel egyeztetve a következők:

- P-vel jelöljük $\mathcal{F}_1=\{f_1^{(1)}, f_1^{(2)}, \ldots, f_1^{(k)}\}$ megadott DNF-jeiben szereplő az egyes függvényekhez való tartozást mutató indexekkel ellátott konjunkciók listáját.
- Q-val jelöljük $\mathcal{F}_o=\{f_o^{(1)}, f_o^{(2)}, \ldots, f_o^{(k)}\}$ megadott DNF-jeiben szereplő az egyes függvényekhez való tartozást mutató indexekkel ellátott konjunkciók listáját.
- P/Q-val jelöljük a P és a Q lista egyesitését
- P_t illetve Q_t jelöli a P-ből illetve Q-ból az algoritmus egy fázisában kapott részeredmény függvény "P"

illetve "Q" listáját

- $R_{x_i}(P/Q)$ jelöli a redukció operátor x_i szerinti alkalmazását a P/Q-val leirt függvényrendszerre
- $I_{x_i}{}^{\alpha}(P/Q) = P_{x_i}{}^{\alpha} / Q_{x_i}{}^{\alpha}$ jelöli a metszés (interszekció) operátor x_i^{α} szerinti alkalmazását.
- $C_{(v,i)}(P/Q)$ jelöli a lefedés operátor alkalmazását egy adott i indexü v konjunkció mellett

AZ ALGORITMUS LEIRÁSA

Az algoritmusban P_j/Q_j $(j=0,1,2,\ldots)$ jelöli mind a részeredményként kapott P_t/Q_t, mind az eredeti P/Q listát. (P/Q a P_o/Q_o -nak felel meg.)

1. Az $\mathcal{F}=\{f^{(1)}, f^{(2)}, \ldots, f^{(k)}\}=(\mathcal{F}_1, \mathcal{F}_o)$ felirása $P/Q = P_o/Q_o$ alakban
2. $j = 0$
3. $i = 0, \quad p = 1$
4. $i = i + 1$
5. $I_{x_i}(P_j/Q_j) = P_{jx_i}/Q_{jx_i}$ és $I_{\bar{x}_i}(P_j/Q_j) = P_{j\bar{x}_i}/Q_{j\bar{x}_i}$ előállitása
6. Ha mind P_{jx_i}, mind $P_{j\bar{x}_i}$ üres, akkor a 15-dik lépés következik
7. Ha P_{jx_i} és $Q_{j\bar{x}_i}$ metszete nem üres, akkor $\alpha = 1$. A 10-dik lépés következik.
8. Ha $P_{j\bar{x}_i}$ és Q_{jx_i} metszete üres, akkor a 15-dik lépés következik.

9. $\alpha = 0$

10. $p = p \ \& \ x_i^{\alpha}$

11. Ha $Q_{jx_i}{}^{\alpha} \neq 0$, akkor $k=j$, $j=1$, $P_j/Q_j = P_{kx_i}^{\alpha} / Q_{kx_i}^{\alpha}$ és a 4-dik lépés következik.

12. p-primimplikáns indexe r, $j=0$ $P_j/Q_j = C_{(p,r)}\ (P_o/Q_o)$

13. Ha P_o üres, akkor kiirjuk a primimplikánsokat. Az algoritmus befejeződik.

14. A 2-dik lépés következik.

15. $k=j$, $j=1$, $P_j/Q_j = R_{x_i}\ (P_k/Q_k)$. A 4-dik lépés következik.

Az algoritmus blokkdiagramja megegyezik az E algoritmus blokkdiagramjával.

GÉPI REPREZENTÁCIÓ

A gépi reprezentáció konjunkció ábrázolására vonatkozó része megegyezik az E algoritmus leirásánál elmondottakkal.

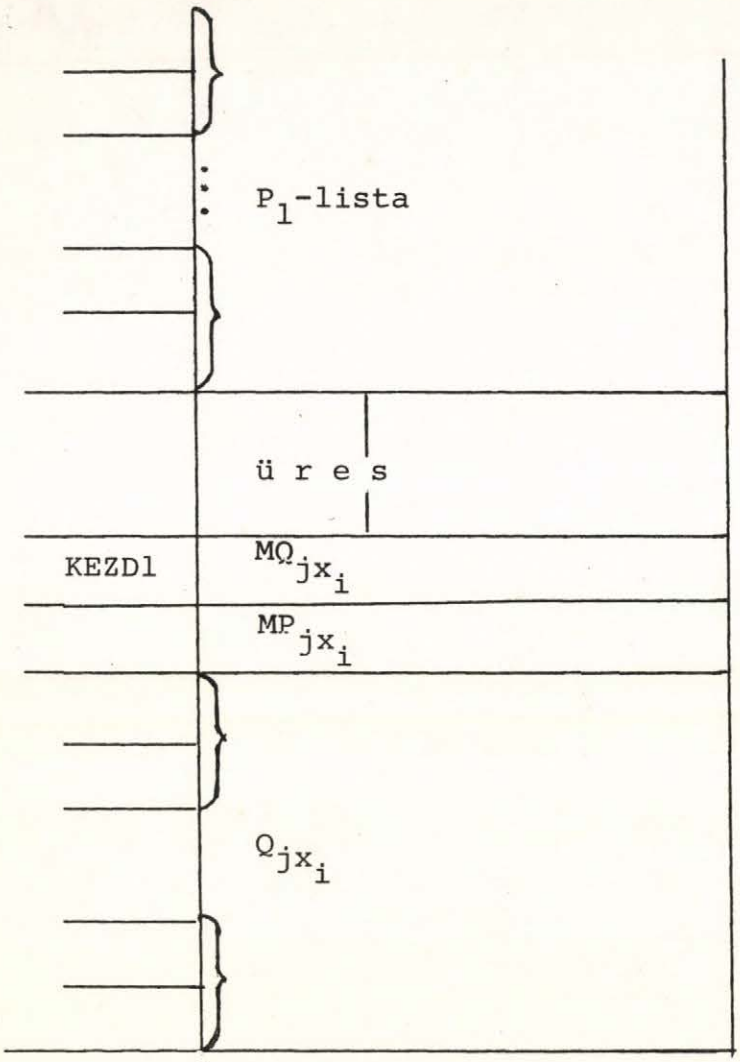
AZ ALGORITMUST REALIZÁLÓ PROGRAM

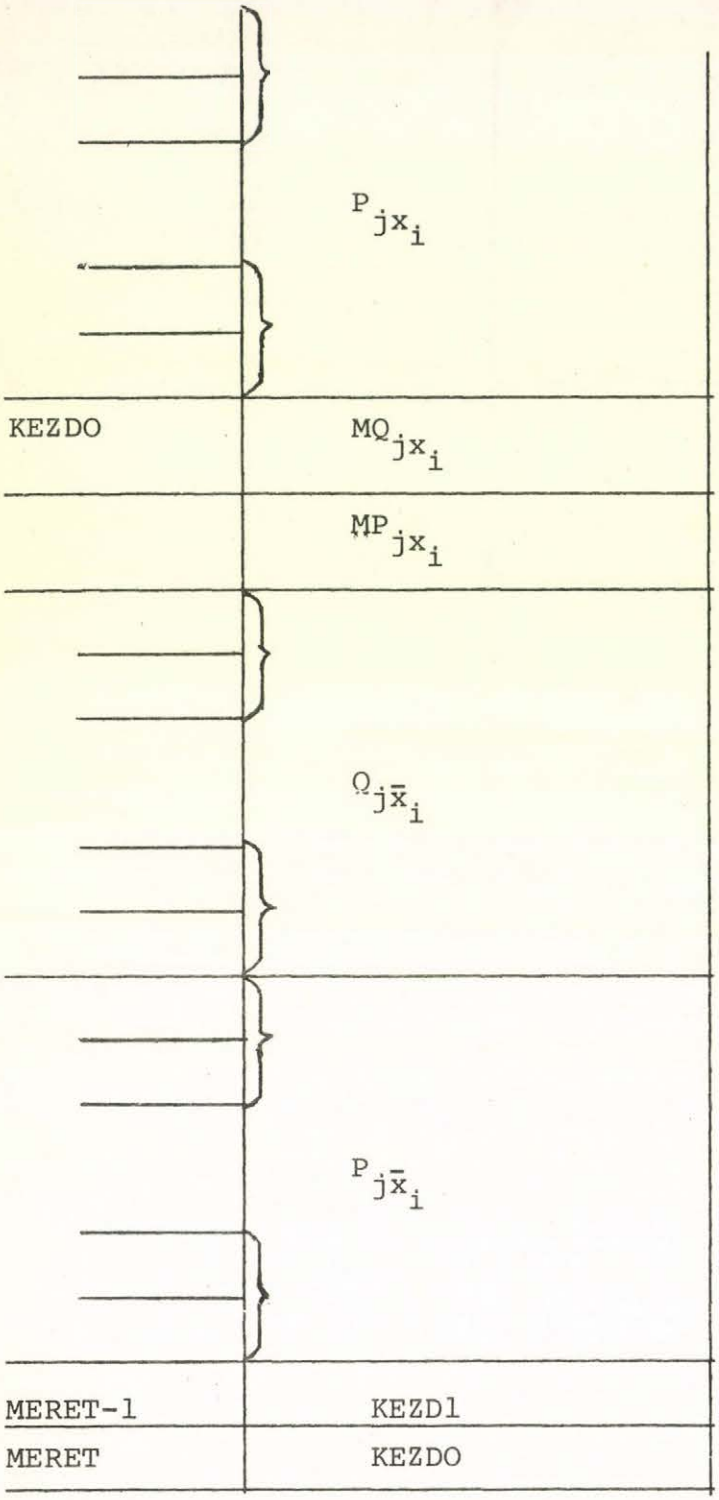
A függvények adatait képező P/Q illetve P_i/Q_i konjunkció + index listákat két előre megadott méretü tömbben (MT és MINDEX-ben) helyezzük el az alábbi szervezés szerint.

MT (i)	
1	változók száma
2	$MQ_o=2K_{Q_o}$
3	$MP_o=2K_{P_o}$
4	Q_o-lista
5	
	($\{f_o^{(i)}\}$ konjunkciói)
	$(i=1,2,\ldots,k)$
MQ+3	
MQ+4	P_o-lista
MQ+5	
	($\{f_1^{(i)}\}$ konjunkciói)
	$(i=1,2,\ldots,k)$
MQ+MP+3	
	$MQ_1=2K_{Q_1}$
	$MP_1=2K_{P_1}$
	Q_1-lista

$K_Q=f_o$ konjunkcióinak száma

$K_P=f_1$ konjunkcióinak száma


P_l-lista
ü r e s
KEZDl
MQ_{jx_i}
MP_{jx_i}
Q_{jx_i}


P_{jx_i}
KEZDO
MQ_{jx_i}
MP_{jx_i}
$Q_{j\bar{x}_i}$
$P_{j\bar{x}_i}$
MERET-1
KEZDl
MERET
KEZDO

Ezen kivül MINDEX(K)-ban annak a konjunkciónak az indexe található, amelynek MT(2K) egy komponense.

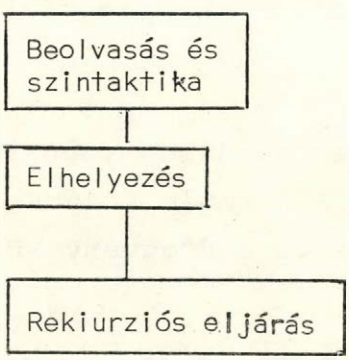
Az algoritmus leirásánál felsorolt R_{x_i}, $I_{x_i^\alpha}$, C_{v_i} operátorokat szubrutinokkal realizáltuk. Külön szubrutin végzi a $P_{jx_i^\alpha}$ & $Q_{jx_i^{\bar{\alpha}}} = \emptyset$ valamint $Q_{jx_i^\alpha} = \emptyset$ vizsgálatát.

Ebből következik, hogy az eljárás programjának elvi blokkdiagramja nem különbözik az algoritmusétól.

Az $\mathcal{F}_1$, $\mathcal{F}_0$ függvényhalmazt tetszőleges DNF-ival kell megadni.

- a beolvasó program a formulák szintaktikai vizsgálata után előállitja a konjunkciók belső reprezentációinak listáját.
- az elhelyező program a konjunkció lista és a függvényekhez való tartozás alapján P_0/Q_0-al iniciálja az MT és a MINDEX tömböket.
- ezután a rekurziv eljárást megvalósitó program megkeres egy az F függvényrendszert nemredendáns módon lefedő indexezett primimplikáns halmazt.

A PROGRAM SZERKEZETE



A PROGRAM LÉNYEGES SZUBRUTINJAI:

SUBROUTINE TDNFEL - MT és MINDEX inicializálására

SUBROUTINE PQTREX - R operátor végrehajtása

SUBROUTINE PTMET - $P_{x_i^{\alpha}}$

SUBROUTINE QTMET - $Q_{x_i^{\alpha}}$ } előállitása $I_{x_i^{\alpha}}$ végrehajtásához

SUBROUTINE PQTUR - $P_i \wedge Q_j = \emptyset$ eldöntése

SUBROUTINE TLEFED - C_p operátor végrehajtása

SUBROUTINE REKTFV - az algoritmus végrehajtását szervező program

FUNCTION QNULL - annak eldöntése, hogy Q P-re nézve üres-e

SUBROUTINE TDNFEL

Adott a *K1*, *K2* tömb.

- A *K1(I)*, *K2(I)* egy konjunkció belső reprezentációja.
- IV a változók száma (*IV=NF(5,I)*)
- *NF(3,J)=1* vagy *0* aszerint, hogy a J-edik formula az NF(1,J) nevü NF(2,J) indexü függvény 1-helyeit vagy 0-helyeit irja-e le.
- *NF(6,J)* és *NF(7,J)* a J-edik formula első konjunkciójának illetve utolsó konjunkciójának cime K1, K2-ben.

A szubrutin A db. nem teljesen meghatározott függvény 1-helyeit és 0-helyeit megadó 2A formula konjunkcióit K1, K2-ből átirja az MT tömbbe indexezve a függvényekhez tartozás szerint a következő módon.

- A függvények 0-helyeit leiró formulák konjunkcióit K1, K2-ből függvényenként és konjunkciónként átirja MT-be MT(4)-től folyamatosan és INDEX tömbbe elhelyezi az átirt konjunkció indexét, ha az eddig már elhelyezett konjunkciók között

  1. még nem szerepel ez a konjunkció
  2. még nem szerepel egy ezt a konjunkciót implikáló konjunkció
  3. ez a konjunkció nem implikál egyetlen már elhelyezett konjunkciót sem.

Az 1. esetben csak a konjunkcióval megegyező, valamint azt implikáló, már elhelyezett konjunkciók indexét módositjuk. Ha az 1. eset nem áll fenn, akkor a konjunkciót elhelyezzük a lista végére és a konjunkció módositja mindazon már elhelyezett konjunkciók indexét, amelyek őt implikálják. Valamint minden olyan már elhelyezett konjunkció, amelyet ez a konjunkció implikál, módositja az ő indexét.

- A függvények 1-helyeit leiró formulák konjunkcióit pedig ugyanigy ezek után folyamatosan helyezi el az MT tömbben.
- MT(1)-be beirja a változók számát, MT(2)-be és MT(3)-ba a 0-helyek illetve az 1-helyek konjunkciói számára szükséges szavak számát (MQ, MP).

KÖZÖS ADATMEZŐK:

```
COMMON(MUNK) N9,IV,IPE,IPO,LF,MPR,KP,KO,M2,N,MMA1,K3,IS,L1,
             M,L,NF(7,23),JV(200),MUV,MOD,BIT(23),FULBIT,WORDL
COMMON(55) MT(9900),K1(3000),K2(3000),JVB(30),INDEX(570)
COMMON(TABL)MINDEX(700)
COMMON(JELEK) MF(45)
COMMON(HELY) IH(24)
```

/ld. a 9. sz. mellékletet/

SUBROUTINE REKTFV

Az algoritmus végrehajtását vezérli. A szubrutin felépitése megegyezik az E eljárás leirásában is megadott algoritmussal, mivel az egyes operációkat itt is külön szubrutinok végzik. A paraméterátadás és memóriaszervezés megegyezik az E eljárásával.

A memóriaigény 6N szó, amennyiben a függvények konjunkcióinak elhelyezéséhez N szó szükséges.

Közös területek:

```
COMMON(MUNK) N9,IV,IPE,IPO,LF,MPR,KP,KO,M2,N,MMA1,K3,IS,L1,M,L,
             NF(7,23),JV(200),MUV,MOD,BIT(23),FULBIT,WORDL
COMMON(55) MT(9300),INDEX(2550),MINDEX(4650)
COMMON(TABLA) K1(230),K2(230),KONX(240)
COMMON(HELY) IH(24)
```

Ha az 1. eset nem áll fenn, akkor a konjunkciót elhelyezzük a lista végére és a konjunkció módositja mindazon már elhelyezett konjunkciók indexét, amelyek őt implikálják. Valamint minden olyan már elhelyezett konjunkció, amelyet ez a konjunkció implikál, módositja az ő indexét.

- A függvények 1-helyeit leiró formulák konjunkcióit pedig ugyanigy ezek után folyamatosan helyezi el az MT tömbben.
- MT(1)-be beirja a változók számát, MT(2)-be és MT(3)-ba a 0-helyek illetve az 1-helyek konjunkciói számára szükséges szavak számát (MQ,MP).

Közös adatmezők:

```
COMMON(MUNK) N9,IV,IPE,IPO,LF,MPR,KP,KO,M2,N,MMA1,K3,IS,L1,M,L,
             NF(7,23),JV(200), MUV,MOD,BIT(23),FULBIT,WORDL
COMMON(55) MT(9900),K1(3000),K2(3000),JVB(30),INDEX(570)
COMMON(TABL) MINDEX(700)
COMMON(JELEK) MF(45)
COMMON(HELY) IH(24)                    /ld. a 10. sz. mellékletet/
```

SUBROUTINE PQTREX(PQ,PQU,MPQ,PQIND,PQUIND,MPQIND,I,JP)

A PQ tömb elempárjai konjunkciókat reprezentálnak. A szubrutin törli a tömb I-edik oszlopát és a törlés utáni konjunkciókra alkalmazza az elnyelési szabályt, ha a konjunkciók PQIND tömbbeli indexe is megegyezik.

A szubrutin tehát az R_x operátor végrehajtását végzi a P vagy a Q mátrixra.

Bemenő paraméterek:

PQ - az adattömb
PQU - az eredmény elhelyezésére szolgáló tömb
MPQ - a PQ tömb mérete
PQIND- a PQ tömb indextömbje
PQUIND - az eredménytömb indextömbje
MPQIND - az indextömb mérete
I - az x_i-nek megfeleltetett helyérték

Kimenő paraméter:

JP - a PQU sorpárjainak száma

Közös adatmezők:

```
COMMON(MUNK) N9,IV,IPE,IPO,LF,MPR,KP,KO,M2,N,MMA1,K3,IS,L1,M,L,
             NF(7,23),JV(200),MUV,MOD,BIT(23),FULBIT,WORDL
COMMON(HELY) IH(24)
```

/ld. a 11. sz. mellékletet/

SUBROUTINE PIMET(P,PU,MP,PIND,PUIND,MPIND,I,ALFA,IK)

A P tömb elempárjai konjunkciókat reprezentálnak. A PIND tömb a konjunkciók indexeit tartalmazza. A szubrutin kiválasztja a P tömbből azokat a konjunkciókat, ahol az I-edik változó kitevője ALFA, majd az igy kapott tömbből elhagyja az I-edik oszlopot és alkalmazza az elnyelési szabályt az indexek figyelembevételével. A szubrutin az $I_x\alpha$ operációt végzi el P-re.

Bemenő paraméterek:

P	- az adattömb
PU	- az eredmény elhelyezésére szolgáló tömb
MP	- a P tömb mérete
PIND	- a konjunkciók indexeit tartalmazó tömb
PUIND	- az eredmény tömb indextömbje
MPIND	- az indextömbök mérete
I	- az x_i-nek megfeleltetett helyérték
ALFA	- az I-edik változó kitevője

Kimenő paraméter:

IK	- a PU sorpárjainak száma

Közös adatmezők:

```
COMMON(MUNK) N9,IV,IPE,IPO,LF,MPR,KP,KO,M2,N,MMA1,K3,IS,L1,M,L,
             NF(7,23),JV(200),MUV,MOD,BIT(23),FULBIT,WORDL
COMMON(HELY) IH(24)
```

/ld. a 12. sz. mellékletet/

SUBROUTINE QTMET(Q,QU,MQ,QIND,QUIND,MQIND,I,ALFA,IK)

A Q tömb elempárjai konjunkciókat reprezentálnak. A QUIND tömb a konjunkciók indexeit tartalmazza. A szubrutin elhagyja a Q tömbből azokat a konjunkciókat, ahol az I-edik változó kitevője ALFA negáltja, majd a kapott tömbből elhagyja az I-edik oszlopot és alkalmazza az elnyelési szabályt az indexek figyelembevételével. A szubrutin az $I_{x^{\alpha}}$ operációt végzi el a Q-ra.

Bemenő paraméterek:

Q	- az adattömb
QU	- az eredmény elhelyezésére szolgáló tömb
MQ	- a Q tömb mérete
QIND	- a konjunkciók indexeit tartalmazó tömb
QUIND	- az eredmény tömb indextömbje
MQIND	- az indextömbök mérete
I	- az x_i-nek megfeleltetett helyérték
ALFA	- az I-edik változó kitevője

Kimenő paraméter:

IK	- a QU sorpárjainak száma

Közös adatmezők:

```
COMMON(MUNK)N9,IV,IPE,IPO,LF,MPR,KP,KO,M2,M.MMA1,K3,IS,L1,M,L,
            NF(7,23),JV(200),MUV,MOD,BIT(23),FULBIT,WORDL
COMMON(HELY) IH(24)
```

/Ld. a 13. sz. mellékletet/

SUBROUTINE PQTUR(P,Q,MP,MQ, INP, INQ, MINP, MINQ, KI)

A P és Q tömb elempárjai konjunkciókat reprezentálnak. INP, INQ tömbök a konjunkciók indexeit tartalmazzák. A szubrutin megvizsgálja, hogy van-e olyan konjunkció P-ben, amelynek Q valamelyik konjunkciójával való konjunkciója az indexet is figyelembevéve nem 0. A KI=0, ha a két konjunkcióhalmaz idegen.

Bemenő paraméterek:

P	- az adattömb
Q	- az adattömb
MP	- a P tömb mérete
MQ	- a Q tömb mérete
INP	- a P tömb konjunkcióinak indexeit tartalmazó tömb
INQ	- a Q tömb konjunkcióinak indexeit tartalmazó tömb
MINP	- az INP tömb mérete
MINQ	- az INQ tömb mérete

Kimenő paraméter:

KI	- =0, ha a két konjunkcióhalmaz idegen
	>0, egyébként

Közös adatmezők:

```
COMMON(MUNK) N9,IV,IPE,IPO,LF,MPR,KP,KO,M2,N,MMA1,K3,IS,L1,M,L,
             NF(7,23),JV(200),MUV,MOD,BIT(23),FULBIT,WORDL
COMMON(HELY) IH(24)
```

/ld. a 14. sz. mellékletet/

SUBROUTINE TLEFED(P,PU,MP,MPU,K1,K2,PID,MPID,PUID,MPUID,KIND,JP)

A P tömb elempárjai konjunkció párokat, az MPID elemei azok indexeit reprezentálják. K=(K1,K2) egy konjunkció, KIND az indexe. A szubrutin törli P-ből a (K,KIND) konjunkció által elnyelt konjunkciókat és elvégez részleges lefedést is. Ez azt jelenti, hogy

- törli azokat a P-beli((P1,P2),Pind) konjunkciókat, amelyekre

  $K2 \wedge P2 = K2$

  $K2 \wedge P1 = K1$ és $KIND \wedge Pind = Pind$

  (P2 és K2 egy-egy konjunkció második komponense,
  P1 és K1 egy-egy konjunkció első komponense)

- Ha a fenti esetben $KIND \wedge Pind \neq Pind$, akkor P1, P2-t nem töröljük, de indexéből elhagyjuk a KIND-ben is meglévőket.

- Ha $K2 \wedge P2 = M2 \neq K2$, de $M2 \wedge P1 = M2 \wedge K1$ és $KIND \wedge Pind = Pind$, akkor P1, P2-ből annyi uj sorpárt állit elő Pind indexszel, ahány pozición K2=1, de P2=0. Sorravesszük ezeket a pozíciókat és P1, P2-ből az éppen soronkövetkező pozición végrehajtott változtatással előálló sorpárt irjuk át Pu-ba. Azaz a megfelelő pozicióra P2-be 1-es, K1-be pedig a K1 ezen pozicióján lévő információ negáltja kerül.

Bemenő paraméterek:

P	- az adattömb
PU	- a lefedés eredményét tartalmazó tömb
MP	- a P tömb mérete
MPU	- a PU tömb mérete
K1	- a konjunkció I. komponense
K2	- a konjunkció II. komponense
PID	- a P tömb konjunkcióinak indexeit tartalmazó tömb
MPID	- a PID tömb mérete

PUID - a PU tömb konjunkcióinak indexeit tartalmazó tömb
MPUID - a PUID tömb mérete
KIND - egy konjunkció indexe

Kimenő paraméter:

JP - PU sorpárjainak száma

Közös adatmezők:

```
COMMON(MUNK) N9,IV,IPE,IPO,LF,MPR,KP,KO,M2,N,MMA1,K3,IS,L1,M,L,
             NF(7,23),JV(2OO),MUV,MOD,BIT(23),FULBIT,WORDL
COMMON(HELY) IH(24)
```

/Ld. a 15. sz. mellékletet/

LOGICAL FUNCTION QNULL(MQ,INQ,MINQ, INP, MINP, IIMPL)

A QNULL függvény értéke TRUE, ha a Q tömb elemeinek száma 0 vagy az INQ szavaiban lévő bináris információt 0,1 jelsorozatnak tekintve, ezek olyanok, hogy van legalább egy helyérték, ahol mindegyikben 0 áll, az INP-ben viszont van legalább egy olyan szó, ahol a 0,1 jelsorozat ezen pozicióján 1 áll.

Bemenő paraméterek:

MQ - a Q tömb mérete
INQ - bináris információt tartalmazó tömb
MINQ - az INQ tömb mérete
INP - bináris információt tartalmazó tömb
MINP - az INP tömb mérete

Kimenő paraméter:

IIMPL - értéke 0, ha QNULL értéke FALSE

/ld. a 16. sz. mellékletet/

INTEGER FUNCTION SUM (JQP,MHAT,IFV)

A SUM függvény MHAT db szóban lévő 0,1 sorozat bitenkénti diszjunkcióját végzi el. JQP-ben vannak a bitsorozatok.

Bemenő paraméterek:

JQP - a bitsorozatokat tartalmazó tömb
MHAT - a JQP tömb mérete
IFV - egy bitsorozat hossza

Közös adatmezők:

```
COMMON(MUNK) N9,IV,IPE,IPO,LF,MPR,KP,KO,M2,N,MMA1,K3,IS,L1,M,L,
             NF(7,23),JV(200),MUV,MOD,BIT(23),FULBIT,WORDL
COMMON(JELEK) MF(45)
COMMON(HELY) IH(24)
```

/ld. a 17. sz. mellékletet/

SUBROUTINE KONIRT(KSZ,KQ1,KQ2,KQIND,JEL)

Kiirja a KQ1, KQ2 tömbökben tárolt KSZ db. konjunkciót alfanumerikus alakban és a konjunkciókhoz tartozó, a KQIND-ben lévő indexeket bináris alakban.

Bemenő paraméterek:

KQ1 - konjunkciókat tartalmazó tömb
KQ2 - konjunkciókat tartalmazó tömb
KSZ - a KQ1, KQ2 tömbök mérete
KQIND - indexeket tartalmazó tömb
JEL - az index tömb mérete

Közös adatmezők:

```
COMMON(MUNK) J6,IV,IPE,IPO,LF,MPR,KP,KO,M2,N,MMA1,K1,IS,L1,M,L,
             NF(7,23),JV(130),K(70),MUV,MOD,BIT(23),FULBIT,WORDL
COMMON(JELEK) MF(45)
```

/ld. a 18. sz. mellékletet/

ILLUSZTRÁCIÓK A PROBRAM EREDMÉNYIRE

```
 LMOD = 0
F=D.-F.-E.Z.A.-F1+D.-E.F.
D=-D.E.-F.A.Z.-F1+-D.E.-F.Z.A.F1+D.E.-F.-Z+D.E.-F.Z.A+D.E.-F.Z.D1+D.-E.-
F+D.-E.F.
E=-D.-E.-F.A+-D.E.-F.Z. .-F1+-D.E.-F.-Z+D.E.-F.-Z+D.E.-F.Z.A.
-G=D.F.F+-D.-E.F+-D.E.F+ .D1+A.E1+D.E1.*
```

PRIMIMPL INDEX

1.	D.E.-Z																						
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
2.	D.E.A																						
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
3.	D.-E																						
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
4.	D.-E.Z.A.-F1																						
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
5.	D.D1																						
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
6.	-D.-E.A																						
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
7.	F																						
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
8.	E.Z.A																						
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
9.	E.-Z																						
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10.	E.A.-F1																						
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

```
 LMOD = 0
T1=X1.-X2.-X3.-X4.
-T1=-X1.X2.X3.X4+X1.-X2.X3.-X4+-X1.-X2.-X3.-X4+X1.X2.X3.-X4,
T2=X1.-X2.X3+X1.-X2.-X3.-X4,
-T2=-X1.X2.X3.X4,
T3=X1.-X2.X3+X1.-X2.-X3.-X4,
-T3=-X1.-X2.X3.-X4,
*
```

```
   PRIMTMPL                                                    INDEX

1.     X1
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0
2.     X1.-X3
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
```

(Az 1. táblázat 2-dik sorában lévő függvény.)

```
 LMOD = 0
F=A.-B.-D+A.-C.-D+A.B.-C.E.F,
-F=-A.B.D+A.B.-D.C,
-G=A.B.-C+-A.B.D.C.G,
H=-B.-C.-D+B.-C.D+-A.-C.D+-A.-B.-C,
G=A.-B.D+-A.B.-C.D,
-H=A.B.C+-A.B.-C.-D+-A.-B.C.D,
*
```

PRIMIMPL INDEX

```
1.    A.-C
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1
2.    -A.D.-C
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0
3.    -B
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1
4.    -B.-C
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0
```

(Az 1. táblázat 3-dik sorában lévő függvény.)

1. FÜGGELÉK

Φ - BOOLE-FÜGGVÉNYEK VAGY NEM TELJESEN MEGHATÁROZOTT BOOLE-FÜGGVÉNYEK MEGADÁSI MÓDJA

Jelölje az $\alpha_1,\alpha_2,\ldots,\alpha_n$ ($\alpha_i=0$ vagy 1) értékkombinációt $\underline{\alpha}$.

Jelölje $f(\underline{\alpha})$ az $f(x_1,x_2,\ldots,x_n)$ függvény értékét az $\underline{\alpha}$ helyen.

Nem teljes vagy parciális, vagy Φ-Boole-függvénynek nevezzük az olyan n változós $f(x_1,x_2,\ldots,x_n)$ függvényt, amelynek értéke a változók bizonyos értékkombinációira nem meghatározott. Tehát a függvény értékei 0 vagy 1 vagy Φ (nem meghatározott) lehetnek. Boole-függvényen (röviden: függvényen) a továbbiakban teljesen meghatározott, vagy parciális Boole-függvényt értünk.

Egy f Φ-Boole-függvényhez tehát három teljesen meghatározott Boole-függvényt rendelhetünk.

$$f_1(\underline{\alpha}) = \begin{cases} 1, & \text{ha f meghatározott az } \underline{\alpha} \text{ helyen és } f(\underline{\alpha})=1 \\ 0, & \text{egyébként} \end{cases}$$

$$f_0(\underline{\alpha}) = \begin{cases} 1, & \text{ha f meghatározott az } \underline{\alpha} \text{ helyen és } f(\underline{\alpha})=0 \\ 0, & \text{egyébként} \end{cases}$$

$$f_\phi(\underline{\alpha}) = \begin{cases} 1, & \text{ha f nem meghatározott az } \underline{\alpha} \text{ helyen} \\ 0, & \text{egyébként} \end{cases}$$

Az $\underline{\alpha}$ értékkombinációt az f nem teljesen meghatározott Boole-függvény

1 - kombinációjának (pontjának) nevezzük, ha $f(\underline{\alpha})=1$

0 - kombinációjának pontjának nevezzük, ha $f(\underline{\alpha})=0$

Φ vagy határozatlansági kombinációjának (pontjának) nevezzük, ha $f(\underline{\alpha})$ nem meghatározott

f_1-et a függvény alsó határának is nevezik. Jele: $\underset{\wedge}{f}$.

$f_1 \vee f_\phi$ -t a függvény felső határának is nevezik. Jele: $\hat{f}$.

D e f i n i c i ó:

Egy Φ-Boole-függvénnyel kompatibilis minden olyan teljesen meghatározott Boole-függvény, melynek értéke a Φ-Boole-függvény meghatározottsági helyein megegyezik annak értékével, a többi helyen pedig tetszőlegesen veheti fel a 0 vagy 1 értéket. Nyilvánvaló, hogy ha egy ϕ-Boole-függvény k helyen nincs meghatározva, a vele kompatibilis teljesen meghatározott Boole-függvények száma 2^k.

Az f-el kompatibilis függvényeket

$$f_k = \underset{\wedge}{f} \vee \gamma \wedge f_\phi$$

alakban is felirhatjuk, ahol γ tetszőleges teljesen meghatározott Boole-függvény.

2. FÜGGELÉK

A REKURZIV OPERÁTOR FELHASZNÁLÁSA NEM TELJESEN MEGHATÁROZOTT BOOLE-FÜGGVÉNY NEMREDUNDÁNS LEFEDÉSÉNEK MEGHATÁROZÁSÁRA

Jelölések:

kisbetük	- Boole-változók, Boole-függvények
nagybetük	- az ugyanazon kisbetüvel jelölt Boole-függvény diszjunktiv normálformája
"0" és "1"	- a logikai "hamis" és "igazi"
$\bar{x}$	- az x Boole-változó negáltja
v	- diszjunkció jele
egymásmellé irás	- Boole-kifejezések konjunkciója

$$x^{\alpha} = \begin{cases} x & \text{ha } \alpha=1 \\ \bar{x} & \text{ha } \alpha=0 \end{cases}$$

Legyen $f(x_1,x_2,\ldots,x_n)$ egy nem teljesen meghatározott n változós Boole-függvény.

Legyen P az f_1 egy tetszőleges diszjunktiv normálformájában szereplő konjunkciók halmaza.

Legyen Q az f_0 egy tetszőleges diszjunktiv normálformájában szereplő konjunkciók halmaza.

($f_1=1$, ahol f=1 és 0 egyébként; $f_0=1$, ahol f=0 és 0 egyébként)

Vezessük be a következő jelölést:

$$f_{x_i=\alpha}(x_1,x_2,\ldots,x_n) = f(x_1,x_2,\ldots,x_{i-1},\ \alpha,x_{i+1},\ldots,x_n) \qquad (1)$$

és az

$f=f(x_1,x_2,\ldots,x_n)$ röviditett felirást

Mint ismeretes:

$$f=x_i\ f_{x_i=1}\ \vee\ \bar{x}_i f_{x_i=0} \tag{2}$$

alkalmazva az $x\ \bar{x}=1$ és az $xl=x$ összefüggést az

$$f=\underbrace{f_{x_i=1}\ f_{x_i=0}}_{A}\ \vee\ \underbrace{x_i f_{x_i=1}\ \bar{f}_{x_i=0}}_{B}\ \vee\ \underbrace{\bar{x}_i f_{x_i=0}\ \bar{f}_{x_i=1}}_{C} \tag{3}$$

kifejezést kapjuk, amelyet nevezzünk az f x_i szerinti diszjunktiv alakjának. Megvizsgálva az egyes tagokat, megállapithatjuk, hogy

- A a függvény x_i-ben szomszédos pontjait jelenti
- B (C) a függvény azon x_i-t ($\bar{x}_i$-at) tartalmazó pontjaiból állt elő x_i törlésével, amelyeknek x_i szerinti szomszédja az $\bar{f}$-hoz tartozik.

Alkalmazzuk a (3) kifejezést $x_j (i\neq j)$ szerint a (3)-ban szereplő A,B,C függvényekre, majd a kapott formulában szereplő függvényekre f még nem használt x_t változója szerint mindaddig, amig minden tag az azonosan 0 vagy azonosan 1 függvénnyel lesz konjunkciós kapcsolatban.

1. *Tétel:* A fenti kifejtési eljárással a redukált diszjunktiv normálformát (az összes primimplikáns diszjunkcióját) kapjuk meg.

Bizonyitás:

Legyen $p = x_{i_1}^{\alpha_{i_1}} x_{i_2}^{\alpha_{i_2}} \dots x_{i_k}^{\alpha_{i_k}}$

1. Tegyük fel, hogy p szerepel a (3) kifejtés szerint kapott diszjunktiv normálformában, de nem primimplikáns. Tehát a p-ből pl. az x_{ij} változót elhagyva is implikánst kapunk. Ez azt jelenti, hogy az a kifejtés során

$x_{i_1}^{\alpha_{i_1}} \dots x_{i_{j-1}}^{\alpha_{i_{j-1}}} \cdot \rho$-ben $\rho \neq 0$, de $\rho_{x_{i_j}} = \alpha_{i_j} \cdot \bar{\rho}_{x_{i_j}} = \bar{\alpha}_{i_j} = 0$,

ami lehetetlen.

2. Tegyük fel, hogy p primimplikánsa f-nek, de nem szerepel a kifejtésben. Ez azt jelenti, hogy van olyan j ($1 \leq j \leq k$) hosszuságu kezdőszelete p-nek, hogy (az 1.-beli okoskodás jelölését használva) $\rho \neq 0$, de $\rho_{x_{i_j}} = \alpha_{i_j} \cdot \rho_{x_{i_j}} = \bar{\alpha}_{i_j} = 0$

vagyis f-nek nincs olyan

$\dots x_{i_1}^{\alpha_{i_1}} \dots x_{i_{j-1}}^{\alpha_{i_{j-1}}} x_{i_j}^{\alpha_{i_j}} \dots$ mintermje, amelyből x_{i_j} ne

lenne kiegyszerüsithető. Ez viszont ellentmondásban van azzal, hogy p primimplikáns.

A (3) összefüggés szukcessziv alkalmazásával tehát előállithatjuk a függvény összes primimplikánsát.

Az eljárás számitógépes realizációja nem egyszerü a nagy müveleti és (függvények konjunkciója, negációja) memóriaigény miatt. Ezért a realizációk egy sor adatkezelési trükköt alkalmaznak. Ezek a DNF és a Boole-függvény karakterisztikus halmaza közötti szemléletes összefüggést használják ki [3].

Mi az alábbiakban a fenti elven alapuló, nem teljesen meghatározott Boole-függvények esetére nemredundáns DNF-et előállitó eljárást ismertetünk. f_1 és f_o DNF-ben adott.

Az eljárás a (3) összefüggést használja fel olymódon, hogy először a B,C függvényeket számitja ki annak eldöntésére, hogy a soronkövetkező x_i változó maga vagy a negáltja szerepel-e a már kiválasztott változókat tartalmazó primimplikánsban vagy nem. Ha $x_i^{\alpha_i}$ szerepel, akkor $x_i^{\alpha_i}$-t a kiválasztott változók közé soroljuk, ha nem akkor a soronkövetkező változóval folytatjuk az eljárást. Igy egy primimplikánst megkapunk. A kapott primimplikáns által lefedett pontokat töröljük a függvény 1-pontjai közül és az eljárást a módositott f_m fügvénnyel folytatjuk mindaddig, amig $f_m=0$ nem lesz.

Az ilymódon kapott $\{p_i\}$ i=1,2,...,k primimplikánshalmaz nemredundáns módon "lefedi" az f függvényt, azaz

$$\sum_{i=1}^{k} p_i = f, \quad \text{de} \quad \sum_{i=1}^{k}{}' p_i \neq f \quad \text{bármely } p_i \text{ elhagyásával.}$$

Az eredmény természetesen függ a változók kiválasztási sorrendjétől. Ennek vizsgálatával itt nem foglalkozunk.

Az eljárás realizálása DNF-ben megadott függvény esetén

A DNF-ban felirt függvény esetén jól meghatározott részfeladatokkal, a DNF-en végzett operációkkal kiválthatók az egyes müveletek és müveleti eredmény kiértékelések.

Egy n változós f függvény k diszjunkciós tagból (konjunkcióból) álló F diszjunktiv normálformájának feleltessünk meg egy kxn-es mátrixot, amelyben az oszlopokat a változókhoz, a sorokat a konjunkciókhoz rendeljük. (*1. ábra*)

	x_1	x_2	...	x_n
q_1				
q_2				
⋮				
q_k				

1. ábra

DNF mátrix

Az $x_{i_1}^{\alpha_{i_1}} x_{i_2}^{\alpha_{i_2}} \ldots x_{i_k}^{\alpha_{i_k}}$ konjunkció sorában az x_{i_j} $(1 \leq j \leq k)$ oszlopában α_{i_j} áll a többi oszlopban "-".

Például egy ötváltozós függvény $x_i \bar{x}_2 x_4$ konjunkcióját a "1O-1-" sorozattal irhatjuk le.

Példa: $f = x_1 \bar{x}_2 x_4 \vee \bar{x}_1 x_5 \vee x_2 x_3 \bar{x}_5 \vee x_2 \bar{x}_4$

$$F = \begin{vmatrix} 1 & 0 & - & 1 & - \\ 0 & - & - & - & 1 \\ - & 1 & 1 & - & 0 \\ - & 1 & - & 0 & - \end{vmatrix}$$

Legyen f nem teljesen meghatározott függvény.

Jelölje f_1 DNF mátrixát P, f_o DNF mátrixát Q. Mint ismeretes f_1 és f_o meghatározza f-et.

Jelölje f=P/Q azt, hogy f-et f_1 és f_o DNF mátrixával adtuk meg. A továbbiakban a mátrixban nem a változókat és negáltjukat, hanem azok kitevőit szerepeltetjük.

Például: $f_1 = xy\bar{z} \vee uxy \vee \bar{x}u$

$f_o = \bar{y}\bar{z}\bar{u} \vee \bar{x}\bar{z}\bar{u} \vee x\bar{y}$

$$f(x,y,z,u) = \begin{array}{cccc} 0 & - & - & 1 \\ 1 & 1 & 0 & - \\ 1 & 1 & - & 1 \\ \hline - & 0 & 0 & 0 \\ 0 & - & 0 & 0 \\ 1 & 0 & - & - \end{array} = \frac{P}{Q}$$

A következő operációkat definiáljuk.

Redukció-operátor R_{x_i}. Az operátor

- törli P/Q-ból az x_i-nek megfelelő oszlopot.
- mind P-ben, mind Q-ban alkalmazza az elnyelési szabályt.

Például: $R_x(P/Q) = R_x \left(\begin{array}{cccc} 0 & - & - & 1 \\ 1 & 1 & 0 & - \\ 1 & 1 & - & 1 \\ \hline - & 0 & 0 & 0 \\ 0 & - & 0 & 0 \\ 1 & 0 & - & - \end{array} \right) = \begin{array}{cccc} - & - & - & 1 \\ - & 1 & 0 & - \\ - & 1 & - & 1 \\ \hline - & 0 & 0 & 0 \\ - & - & 0 & 0 \\ - & 0 & - & - \end{array} = \begin{array}{cccc} - & - & - & 1 \\ - & 1 & 0 & - \\ \hline - & - & 0 & 0 \\ - & 0 & - & - \end{array}$

Az R_{x_i} mind f_1, mind f_o pontjaihoz hozzáveszi a velük x_i szerint szomszédosakat is függetlenül attól, hogy igy kötelezően 1 pontok is kerülnek a 0 pontok közé és forditva. Ezek a P-beli sorok olyan konjunkciókat reprezentálnak, amelyekben x_i nem szerepel. E konjunkciók közül azokból, amelyek 0 pontot nem tartalmaznak, esetleg még további változók elhagyásával kaphatunk x_i-t nem tartalmazó primimplikánsokat.

Metszés-operátor: $I_{x_i^{\alpha}}$. Az operátor P-ből csak azokat a sorokat tartja meg, ahol x_i^{α} szerepel, Q-ból pedig csak azokat a sorokat törli, ahol $x_i^{\bar{\alpha}}$ szerepel. Az eredményre R_{x_i}-t alkalmazza.

Például: $I_x = \left(\begin{array}{cccc} 0 & - & - & 1 \\ 1 & 1 & 0 & - \\ 1 & 1 & - & 1 \\ \hline - & 0 & 0 & 0 \\ 0 & - & 0 & 0 \\ 1 & 0 & - & - \end{array}\right) = \begin{array}{cccc} - & 1 & 0 & - \\ - & 1 & - & 1 \\ \hline - & 0 & 0 & 0 \\ - & 0 & - & - \end{array} = \begin{array}{cccc} - & 1 & 0 & - \\ - & 1 & - & 1 \\ \hline - & 0 & - & - \end{array} = P_x/Q_x$

Az I_{x_i} operátor megtartja f_1 DNF-jában azokat a konjunkciókat, amelyekben eredetileg szerepelt az x_i^{α}, az f_o DNF-jában pedig azokat a konjunkciókat, amelyek lefednek x_i^{α}-t tartalmazó konjunkciókat.

Lefedés-operátor: C_p. Törli P-ből azon k_i konjunkcióknak megfelelő sorokat, amelyekre $p \cdot k_i = k_i$ elvégzi a lefedést, ha $p=r \cdot q$, $k_i = r \cdot t$ ahol r,q,t változóidegen konjunkciók, ugy hogy, ha $q = x_{i_1}^{\alpha_1} x_{i_2}^{\alpha_2} \ldots x_{i_k}^{\alpha_k}$, akkor k_i sorából k uj sort generál, amelyek rendre a

$$k_i x_{i_j}^{\bar{\alpha}_j} \quad (j=1,2,\ldots,m)$$

konjunkciókat reprezentálják.

Például: $p=xu$

$$C_p \left(\begin{array}{cccc} 0 & - & - & 1 \\ 1 & 1 & 0 & - \\ 1 & 1 & - & 1 \\ \hline - & 0 & 0 & 0 \\ 0 & - & 0 & 0 \\ 1 & 0 & - & - \end{array}\right) = \begin{array}{cccc} 0 & - & - & 1 \\ 1 & 1 & 0 & 0 \\ \hline - & 0 & 0 & 0 \\ 0 & - & 0 & 0 \\ 1 & 0 & - & - \end{array}$$

A C_p operátor törli f_1 pontjai közül a p konjunkció által lefedetteket. Ezuton biztositja a nemredundáns lefedést.

Vizsgáljuk meg a (3) képletben lévő A,B,C kifejezések és az operátorok viszonyát.

- Az R_{x_i} operátor az

$$\underbrace{f_{1_{x_i=0}} \vee f_{1_{x_i=1}}}_{P_R} \;/\; \underbrace{f_{0_{x_i=0}} \vee f_{0_{x_i=1}}}_{Q_R}$$ -t állitja elő, amely

tartalmaz minden információt A-ra ugyanis P_R ekvivalens A-val, ha elhagyjuk belőle a Q_R-el közös pontokat.

- Az $I_{x_i}^{\alpha}$ operátor eredményére a B,C előállitásánál van szükség. B és C olyan x_i-ban szomszédos pontpárokat fed le, amelyek közül mindig csak az egyik, az x_i^{α}-t tartalmazó tartozik f-hez. Ennek x_i^{α} -val való konjunkciója az f azon pontjait jelenti, amelyek csak x_i^{α}-t is tartalmazó primimplikánsokkal fedhetők le. A $P_{x_i}\alpha$-ban azok a konjunkciók szerepelnek, amelyekből x_i^{α} esetleg nem egyszerüsithető ki. Mivel $Q_{x_i}^{\alpha}$ tartalmaz minden olyan x_i-ben szomszédos pontpárt, amelyek közül az x_i^{α}-t tartalmazó mintermmel lefedett az f_o-nak pontja, ezért ha $P_{x_i}^{\alpha}\, Q_{x_i}^{\bar{\alpha}}$ konjunkciója 0, akkor B=0 ha α=1 és C=0 ha α=0 (minden f_1-hez tartozó pont x_i szerinti szomszédja vagy f_1-hez vagy f_ϕ-hez tartozik). E konjunkció kiszámitására a realizációban nincs szükség, mivel a döntés anélkül is meghatározható.

Az elmondottak alapján kimondhatjuk a következő tételt:

2. *Tétel:* Ha $P_{x_i}^{\alpha}\, Q_{x_i}^{\bar{\alpha}} = \emptyset$.

akkor az $x_i^{\alpha}\, P_{x_i}^{\alpha}$ -val leirt függvény pontjai lefedhetők

x_i^α -t nem tartalmazó primimplikánssal.

- Ahhoz, hogy $x_i(\bar{x}_i)$ primimplikáns legyen a B≡1 (C≡1) feltételnek kell teljesülnie. Ez azt jelenti a (3) szerint, hogy $f_{x_i=1} = \bar{f}_{x_i=o}$ $(f_{x_i=o} = \bar{f}_{x_i=1})$.

Más szóval f-nek az n dimenziós Boole-tér minden olyan s_{x_i} pontja 1-pontja, amely x_i-t ($\bar{x}_i$-at) tartalmazó konjunkcióval fedhető le. Tehát az x_i-től különböző változóktól a függvény nem függ, igy a DNF-ből kiegyszerüsithetők. Esetünkben a B≡1 (C≡1) feltétel azt jelenti, hogy a fent emlitett s_{x_i} pontok közül egy sem O pontja a függvénynek, ezért $Q_{x_i}=\emptyset$ $(Q_{\bar{x}_i}=\emptyset)$.

E megállapitással bebizonyitottuk az alábbi tételt:

3. Tétel: Ha $P_{x_i^\alpha} / Q_{x_i^\alpha} = \emptyset$, akkor x_i^α primimplikánsa az $x_i^\alpha P_{x_i^\alpha}$ -val leirt függvénynek.

3. FÜGGELÉK

A REKURZIV OPERÁTOR FELHASZNÁLÁSA NEM TELJESEN MEGHATÁROZOTT BOOLE-FÜGGVÉNYEKBŐL ÁLLÓ FÜGGVÉNYRENDSZER NEMREDUNDÁNS EGYÜTTES DNF-JÁNAK MEGHATÁROZÁSÁRA

A jelölésekben a 2. függelékben megadottakhoz egy kiegészitést adunk.

Irott nagybetük - függvényrendszer jele

$$\text{Pl. } \mathcal{F} = \{f^{(1)}, f^{(2)}, \ldots, f^{(k)}\}$$

Legyen $\mathcal{F}$ egy k nem teljesen meghatározott Boole-függvényből, mint komponens függvényekből álló függvényrendszer.

Legyen $\mathcal{F}_1 = \{f_1^{(1)}, f_1^{(2)}, \ldots, f_1^{(k)}\}$ és

$$\mathcal{F}_o = \{f_o^{(1)}, f_o^{(2)}, \ldots, f_o^{(k)}\}$$

Legyen P az $\mathcal{F}_1$-hez tartozó komponens függvények DNF-jaiban szereplő konjunkciók egyesitett listája, ahol a komponensekhez való tartozást is hozzárendeljük a konjunkciókhoz.

Legyen Q az $\mathcal{F}_o$-hoz tartozó komponens függvények DNF-jaiban szereplő konjunkciók egyesitett listája, ahol a komponensekhez való tartozást is hozzárendeljük a konjunkciókhoz.

A 2. függelékben bevezetett (1) jelölés és $f=f(x_1,x_2,\ldots,x_n)$ röviditett felirás mellett vezessük be az

$$\mathcal{F}_{x_i=1}\ \mathcal{F}_{x_i=o} = \{f^{(1)}_{x_i=1}\ f^{(1)}_{x_i=o}, f^{(2)}_{x_i=1}\ f^{(2)}_{x_i=o}, \ldots, f^{(k)}_{x_i=1}\ f^{(k)}_{x_i=o}\} \quad (1)$$

jelölést. Ekkor a 2. függelék (3) összefüggése $\mathcal{F}$-re értelmezve

$$\mathcal{F} = \mathcal{F}_{x_i=1}\,\mathcal{F}_{x_i=0} \vee x_i \mathcal{F}_{x_i=1}\,\bar{\mathcal{F}}_{x_i=0} \vee \bar{x}_i\,\mathcal{F}_{x_i=0}\,\bar{\mathcal{F}}_{x_i=1} \qquad (2)$$

A B C

lesz, amelyet $\mathcal{F}$ x_i szerinti együttes diszjunktiv alakjának nevezünk. Megvizsgálva az egyes tagokat, megállapithatjuk, hogy

- A a komponens függvények x_i-ben szomszédos pontpárjait jelenti.
- B(C) a komponens függvények azon x_i-t ($\bar{x}_i$-at) tartalmazó pontjaiból állt elő x_i törlésével, amelyeknek x_i szerinti szomszédja ugyanazon $f^{(t)}$ komponensfüggvény negáltjához $\bar{f}^{(t)}$-hez tartozik.

Alkalmazzuk a (2) összefüggés szerinti kifejtést a (2)-ben szereplő függvényekre valamely a kifejtésben még fel nem használt x_t változó szerint mindaddig, amig minden tagban az A, B és C szerepét játszó függvény minden komponense azonosan O vagy a komponensei közül legalább egy azonosan 1 és a többi azonosan O nem lesz.

1. Tétel:

A fenti kifejtési eljárással az együttes redukált DNF-át (az összes együttes primimplikáns [2] diszjunkcióját) kapjuk meg. Egy-egy primimplikáns a vele konjunkciós kapcsolatban álló B vagy C tipusu függvény azonosan 1 komponens függvényeinek közös primimplikánsa.

Bizonyitás:

A 2. függelék 1. tétele és az (1) értelmezéseinek felhasználásával következik.

Tehát a (2) összefüggés szukcessziv alkalmazásával előállithatjuk a függvény összes együttes primimplikánsát.

Az eljárás számitógépes realizációjára méginkább érvényesek a 2. függelékben elmondottak.

Az alábbiakban nem teljesen meghatározott Boole-függvény komponensekből álló függvényrendszerre olyan a fenti elven alapuló eljárást ismertetünk, amely egy nemredundáns együttes DNF-et ad. Az $f_i^{(i)}$, $f_o^{(i)}$ komponensfüggvényeket tetszőleges DNF-jukkal adjuk meg.

Az eljárásbeli stratégia megegyezik a 2. függelékben lévővel. Először kiszámitjuk x_i-hez a B, C függvényrendszert annak eldöntésére, hogy x_i vagy $\bar{x}_i$ feltétlenül szerepel-e a kifejtésben már felhasznált változók közül a rekurziv formula szerintieket tartalmazó primimplikánsban.

Ha $x_i^{\alpha_i}$ szerepel, akkor $x_i^{\alpha_i}$-t is a kiválasztott változók közé soroljuk, ha nem, akkor a soronkövetkező változóval folytatjuk az eljárást. Igy megkapunk egy primimplikánst (p_i-t) és az implikált komponensfüggvényeket (ind_{p_i}-t).A kapott együttes primimplikáns által lefedett pontokat töröljük a megfelelő komponensfüggvények 1-pontjai közül. Az eljárást a módositott $\mathcal{F}_m$ függvénnyel ujrakezdjük mindaddig, amig $\mathcal{F}_m$ azonosan 0 nem lesz.

Az ilymódon kapott $\{(p_i, ind_{p_i})\}$ $(i=1,2,\ldots,r)$ együttes primimplikánshalmaz nemredundáns módon "lefedi" az $\mathcal{F}$ általános Boole-függvényt.

Az eredmény és a változók kiválasztási sorrendjének összefüggésével itt sem foglalkozunk.

AZ ELJÁRÁS REALIZÁLÁSA DNF-BEN MEGADOTT KOMPONENS FÜGGVÉNYEK ESETÉN

A 2. függelékben bemutatott reprezentációt felhasználjuk (lásd ott *1. ábra* és *példa*).

Ezenkivül az i-edik komponens függvény indexének nevezzük a 2^{i-1} kettes számrendszerbeli alakját és egy konjunkció indexe a konjunkció által implikált függvények indexeinek összege lesz.

Például:

$$\mathcal{F} = \{f^{(1)}, f^{(2)}\}$$

$$f^{(1)} = x\bar{y} \vee zy, \quad f^{(2)} = x\bar{y} \vee xzy \vee \bar{x}\bar{z}\bar{y}$$

A komponens függvények indexei:

$$f^{(1)} - 01$$

$$f^{(2)} - 10$$

A konjunkciókhoz rendelt indexek:

$$x\bar{y} - 11$$
$$zy - 01$$
$$xzy - 11$$
$$\bar{x}\bar{z}\bar{y} - 10$$

Az $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_0)$ függvényrendszer DNF mátrixa indexezett konjunkciókat tartalmaz.

Jelölje $\mathcal{F}_1$ DNF mátrixát P, $\mathcal{F}_0$ DNF mátrixát Q.

Jelölje $\mathcal{F} = P/Q$ azt, hogy $\mathcal{F}$-et az $\mathcal{F}_1$ és $\mathcal{F}_0$ DNF mátrixával adtuk meg.

A mátrix konjunkciókat reprezentáló része a 2. függelékben lévővel megegyezik és kiegészül a konjunkció indexével.

Például:

$$\mathcal{F} = \{f^{(1)}, f^{(2)}\}$$

$$f_1^{(1)} = xy \vee \bar{x}\bar{y}\bar{z}$$

$$f_1^{(2)} = yz\bar{u} \vee \bar{x}z\bar{u} \vee \bar{x}\bar{y}\bar{z}\bar{u}$$

$$f_o^{(1)} = \bar{x}\bar{y}\bar{z} \vee \bar{y}zu \vee \bar{x}y$$

$$f_o^{(2)} = x\bar{y}\bar{z}\bar{u} \vee x\bar{z}u \vee \bar{y}zu \vee \bar{x}y\bar{z}$$

x-1, y-2, z-4, u-8, $f^{(1)}$-1, $f^{(2)}$-2 megfeleltetés mellett:

$$\mathcal{F}(x,y,z,u) = \frac{\begin{array}{cccc|cc} - & - & 1 & 1 & 0 & 1 \\ - & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & - & 1 & 0 \\ 0 & 1 & - & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{array}}{\begin{array}{cccc|cc} - & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & - & 1 & 1 \\ - & - & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & - & 1 & 1 & 0 \\ - & 0 & 1 & 0 & 1 & 0 \end{array}} = \frac{P}{Q}$$

A következő operációkat definiáljuk és ezeket példaként $\mathcal{F}$-re alkalmazzuk is.

Redukció operátor R_{x_i}. Az operátor

- törli P/Q -ból az x_i-nek megfelelő oszlopot
- mind P-ben, mind Q-ban alkalmazza az elnyelési szabályt

Példa:

$$R_u(P/Q)=R_u$$

-	-	1	1		0	1						-	-	1	1		0	1
-	0	0	0		0	1						-	0	0	0		0	1
0	1	1	-		1	0						-	1	1	-		1	0
0	1	-	0		1	0						-	1	-	0		1	0
1	0	0	0		1	1						-	0	0	0		1	1
-	0	0	1		0	1						-	0	0	1		0	1
1	1	0	-		1	1						-	1	0	-		1	1
-	-	1	0		0	1						-	-	1	0		0	1
0	0	0	1		1	1						-	0	0	1		1	1
1	0	-	1		1	0						-	0	-	1		1	0
-	0	1	0		1	0						-	0	1	0		1	0

(Példánkban az elnyelési szabály nem alkalmazható.)

Metszési operátor $I_{\times_i^{\alpha}}$. Definiciója ugyanaz, mint a 2. függelékben.

Példa:

$$I_u$$

-	-	1	1		0	1													
-	0	0	0		0	1													
0	1	1	-		1	0													
0	1	-	0		1	0													
1	0	0	0		1	1						-	0	0	0		1	1	
-	0	0	1		0	1						-	0	0	1		0	1	
1	1	0	-		1	1						-	1	0	-		1	1	
-	-	1	0		0	1						-	-	1	0		0	1	
0	0	0	1		1	1						-	0	-	1		1	0	
1	0	-	1		1	0						-	0	1	0		1	0	
-	0	1	0		1	0													

$= \frac{P_u}{Q_u}$

Megjegyzés: A 2. függelékben az R_{x_i} -ből és az I_x^{α}-ról mondottak itt is érvényesek az $\mathcal{F}_1$-re és $\mathcal{F}_o$-ra értelmezve.

Lefedés operátor $C_{(p,i)}$. Törli P-ből azon $(k_j.ind_{kj})$ konjunkcióknak megfelelő sorokat, amelyekre $pk_j=k_j$ és p implikál minden függvényt, amelyet k_j is implikál. Ha ez utóbbi nem áll fenn, akkor k_j indexét módositja ugy, hogy elhagyja belőle az i-ben is szereplő függvényindexeket. Elvégzi a lefedést, ha $p=rq$, $k_j=rt$ ahol r, q, t változóidegen konjunkciók és p implikál minden függvényt, amelyet k_j is implikál. A lefedéskor

annyi uj konjunkciót (sort) generál, amennyi a q-ban lévő változók száma. Ezek indexe ind_{k_j} lesz. A generált sorok,

ha $q = x_{i_1}^{\alpha_{i_1}} x_{i_2}^{\alpha_{i_2}} \ldots x_{i_k}^{\alpha_{i_k}}$ rendre

$k_j x_{i_r}^{\bar{\alpha}_{i_r}}$ ind_{k_j} (r=1,2,...,k) lesznek.

Példa:

legyen $(p,i) = (yz\bar{u},10)$

$$C_{(p,i)} \quad \begin{array}{cc} - - 1 1 & 0 1 \\ - 0 0 0 & 0 1 \\ 0 1 1 - & 1 0 \\ 0 1 - 0 & 1 0 \\ 1 0 0 0 & 1 1 \\ \hline - 0 0 1 & 0 1 \\ 1 1 0 - & 1 1 \\ - - 1 0 & 0 1 \\ 0 0 0 1 & 1 1 \\ 1 0 - 1 & 1 0 \\ - 0 1 0 & 1 0 \end{array} = \begin{array}{cc} - - 1 1 & 0 1 \\ - 0 0 0 & 0 1 \\ 0 1 0 0 & 0 1 \\ 1 0 0 0 & \\ \hline - 0 0 1 & 0 1 \\ 1 1 0 - & 1 1 \\ - - 1 0 & 0 1 \\ 0 0 0 1 & 1 1 \\ 1 0 - 1 & 1 0 \\ - 0 1 0 & 1 0 \end{array}$$

A lefedés operátor biztositja a nemredundáns lefedést.

Vizsgáljuk meg a (2) képletben lévő A,B,C kifejezések és az operátorok viszonyát.

<u>D e f i n i c i ó:</u> Két $\mathcal{F}$ és G azonos számu komponens függvényből álló függvényrendszer $\mathcal{F}$ V G diszjunkciója alatt az egymásnak megfeleltetett komponens függvényeik diszjunkcióiból álló függvényrendszert értünk.

$$\mathcal{F} = \{f^{(1)}, f^{(2)}, \ldots, f^{(k)}\}$$

$$= \{g^{(1)}, g^{(2)}, \ldots, g^{(k)}\}$$

$$\mathcal{F}_{\vee G} = \{f^{(1)} \vee g^{(1)}, f^{(2)} \vee g^{(2)}, \ldots, f^{(k)} \vee g^{(k)}\}$$

Ezek után a 2. függelékben f-re elmondottak $\mathcal{F}$-re érvényesek, minimális értelemszerü módositással.

- Az R_{x_i} operátor és A viszonya ugyanaz, mint a 2. függelékben leirt.

- Az $I_{x_i}^{\alpha}$ operátor és B, C viszonya ugyanaz, mint a 2. függelékben leirt, ha az alábbi értelmezés mellett $P_{x_i}^{\alpha} Q_{x_i}^{\bar{\alpha}} = \emptyset$, ha P és Q konjunkcióinak egymással való konjunkciója O vagy ha minden nem nulla eredmény esetén a két konjunkció indexe diszjunkt. Pl. $(-11-,01)\wedge(011-,10) = (011-,00) = 0$

Kimondhatjuk ennek alapján a következő tételt:

2. *Tétel:*

Ha $P_{x_i}^{\alpha} Q_{x_i}^{\bar{\alpha}} = \emptyset$, akkor az $x_i^{\alpha} P_{x_i}^{\alpha}$ -val leirt függvény pontjai lefedhetők x_i^{α} -t nem tartalmazható primimplikánssal.

Ahhoz, hogy $x_i(\bar{x}_i)$ primimplikáns legyen B(C) komponens függvényeinek konstansoknak kell lenni, amelyek közül legalább egy az 1 konstans. Ez azt jelenti (2) szerint, azokra az $f^{(i)}$ komponens függvényekre, amelyek $b^{(i)}\equiv 1$ $(c^{(i)}\equiv 1)$, hogy $f^{(i)}_{x_i=1} = \bar{f}^{(i)}_{x_i=0}$ $(f^{(i)}_{x_i=0} = f^{(i)}_{x_i=1})$.

Más szóval ezeknek az $f^{(i)}$-knek az n dimenziós Boole-tér minden olyan S_{x_i} pontja 1-pontja, amely x_i-t $(\bar{x}_i$-t) tartalmazó teljes elemi konjunkcióval fedhető le. Tehát az x_i-től különböző változóktól e függvények nem függnek. Ezért ezek a változók a DNF-ből kiegyszerüsithetők. A $B\equiv 1$ $C\equiv 1$ feltétel tehát azt jelenti, hogy a fenti S_{x_i} pontok közül egy sem O-pontja az $f^{(i)}$ függvényeknek, ezért $Q_{x_i}=\emptyset$ $(Q_{\bar{x}_i}=0)$,

ezek szerint az $f^{(i)}$ függvények szerint. Más szóval, ha a Q_{x_i} ($\underline{Q}_{x_i}$)-ban a konjunkciólista nem üres, akkor a konjunkciók indexeiben az emlitett $f^{(i)}$ függvények jele nem szerepel. Ezzel bebizonyitottuk az alábbi tételt.

3. *Tétel:*

Ha $P_{x_i}{}^{\alpha}$ /$Q_{x_i}{}^{\alpha}$ -ban $Q_{x_i}{}^{\alpha} = 0$, akkor x_i^{α} primimplikánsa az x_i^{α} $P_{x_i}{}^{\alpha}$ -val leirt függvénynek.

IRODALOMJEGYZÉK

1 W.V. Quine: A Way to Simplify Truth Functions. Amer. Math. Monthly, Vol. 62. Nov. 1955. N^{o} 9. 627-631.

2 Mc Cluskey, E.J. & H. Shorr: Essential Multiple Output Prime Implicants. Proc. Symposium on Mathematical Theory of Automation, v. 12, 1962, pp. 437-452.

3 Pásztorné Varga Katalin: Módszerek Boole-függvények minimális vagy nem redundáns $\{\wedge, \vee, \neg\}$ vagy {NOR} vagy {NAND} bázisbeli, zárójeles vagy zárójel nélküli formuláinak előállitására. MTA SZTAKI Tanulmányok, 1/1973.

4 J.C. Torgue, K. Pásztor, P. Azema: Couverture irrédondante des Fonctions booléennes définies par leurs monomes vrais et faux - fonctions simultanées, Acta Cybernetica, Tom. 2, Fasc. 3, Szeged, 1975.

5 Michel Carvallo: Monographie des treillis et algébre de Boole, Gauthier-Villars, Paris, 1966.

6 P. Tison: Generalisation of Consensus Theory and Application to the Minimization of Boolean Functions, IEEE Transactions on EC Vol. EC-16. N^{o}4. Apr. 1964. 446-456.

7 A. Varga, P. Ecsedi-Tóth and F. Móricz: On a connection between algebraic and graph theoretic minimization of truth function, Proc. of the Int. Coll- on Algebraic Methods in Graph Theory, 1978.

8 Eugenio Morreale: Recursive Operators for Prime Implicant and Irredundant Normal Form Determination, IEEE Transactions on Computers, Vol. C-19, N°6, June, 1970.

9 P. Azema, M. Diaz, J.C. Torgue: Minimisation des Fonctions Logiques Incomplétement Spécifiées, Elektronics Letters, Vol. 8, N° 15, 27th July 1972.

10 M.A. Gavrilov: Minimizacija bulevüh funkcij, harakterizujustyih relejnüh cepi. Avtomatika i telemehanika, 1967, N°9.

11 R.H. Urbano, R.K. Mueller: A Topological Method for the Determination of Minimal Forms of a Boolean Functions, IRE Transactions on EC Vol. EC-5 N°9, September, 1956, 126-132.

12 B. Papp: Procédé pour determiner les formes normales des fonctions boolennes en utilisant les régles de minimisation de la fonction de cout. Acta Cybernetica, 1972, N°4, 241-251.

A **TANULMÁNYOK** sorozatban 1979-ben megjelentek:

88/1979 Renner G. - Gaál B. - Hermann Gy. - Horváth L. - Várady T.: Szoborszerü felületek tervezése és megmunkálása

89/1979 Ruda Mihály: A SIS77 statisztikai információs rendszer /a felhasznált számitástechnikai eszközök, a rendszer szerkezete és programjai/

90/1979 Bányász Cs. - Keviczky L.: Optimum Insensitivity of the Linear-continuous Transformation

91/1979 Téli iskola /Szentendre/

92/1979 Bolla M. - Csáki P. - Fischer J. - Herodek S. - Hoffman Gy. - Kutas T. - Telegdi L. - Wittmann I.: A balatoni ökoszisztéma modellezése

93/1979 Andor László: Kisgépes adatbázis kezelő rendszer

94/1979 Gertler János: Egy statisztikus szürési eljárás számitógépes folyamatirányitásához

95/1979 Báthory M. - Galló V. - Kovács E. - Mérő L. - Siegler A. - Vajta L.: Festőrobot vezérlésére alkalmas alafelsimerési berendezés

96/1979 Mérő László: Konturkeresés zajos digitalizát képekben

97/1979 Pásztorné - Matavovszky T.: Boole-függvény kezelőrendszer

98/1979 Kecskés Zsuzsa: Három dimenziós tárgyak drótvázának ábrázolása vonalrajzoló grafikus berendezésekkel

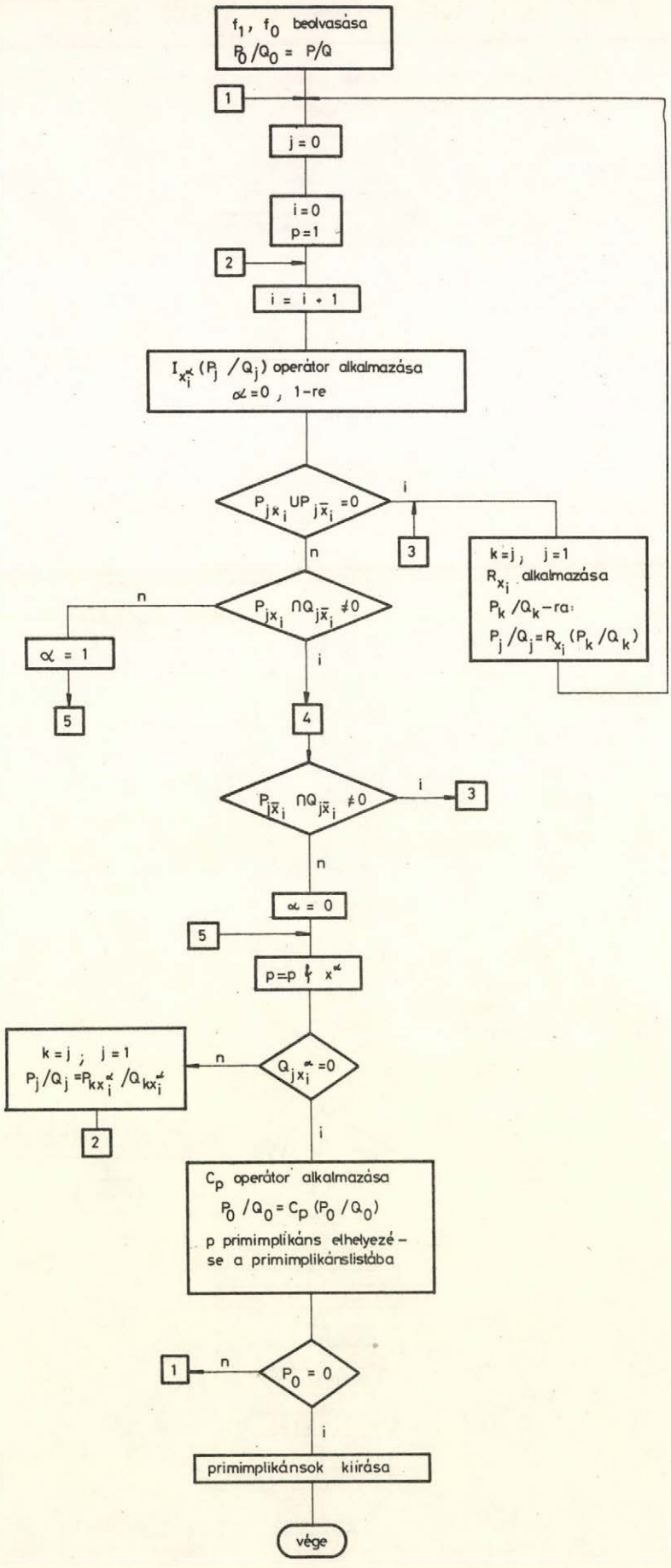
99/1979 Ivics József: KGST Riga

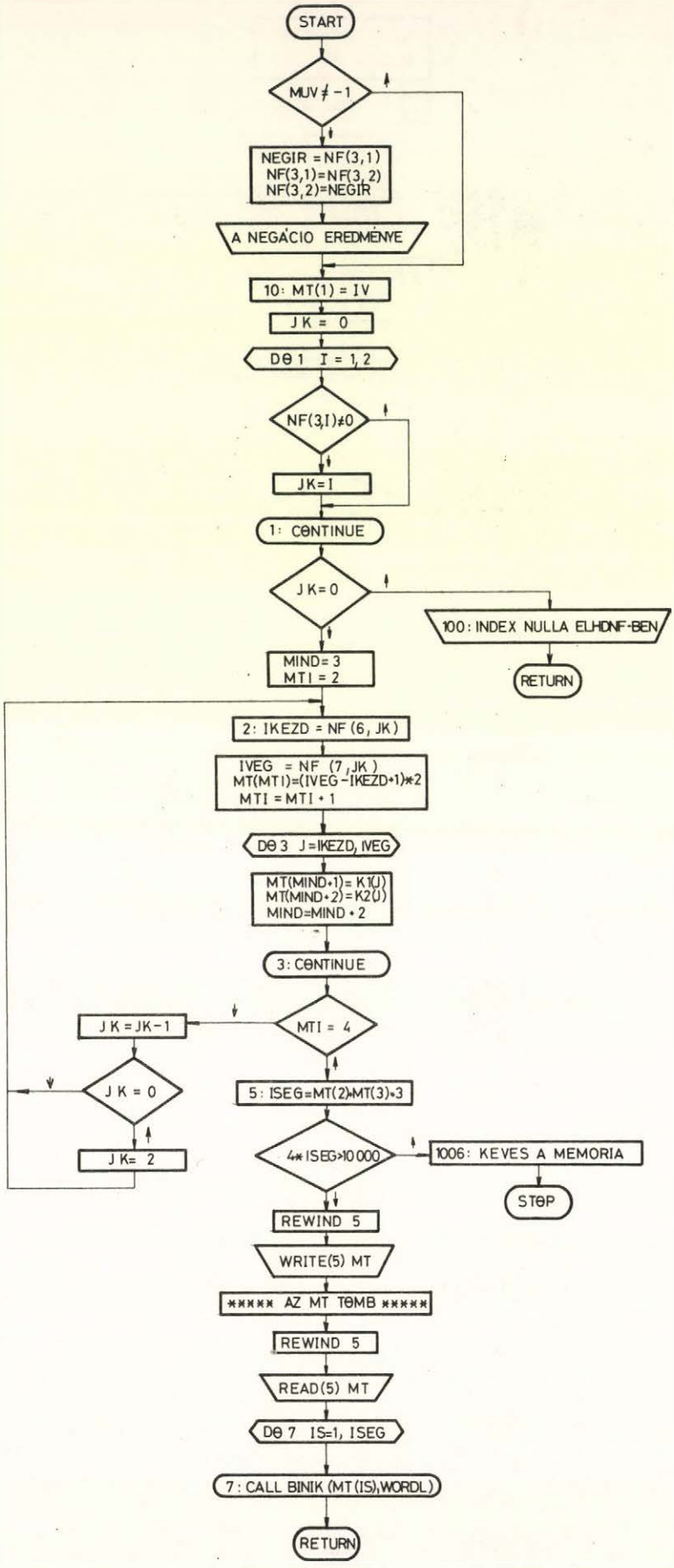
100/1979 Téli iskola

101/1980. Hangos Katalin - Gerencsér László:
Diszkrét lineáris sztochasztikus rendszerek
önhangoló szabályozása

MELLÉKLETEK

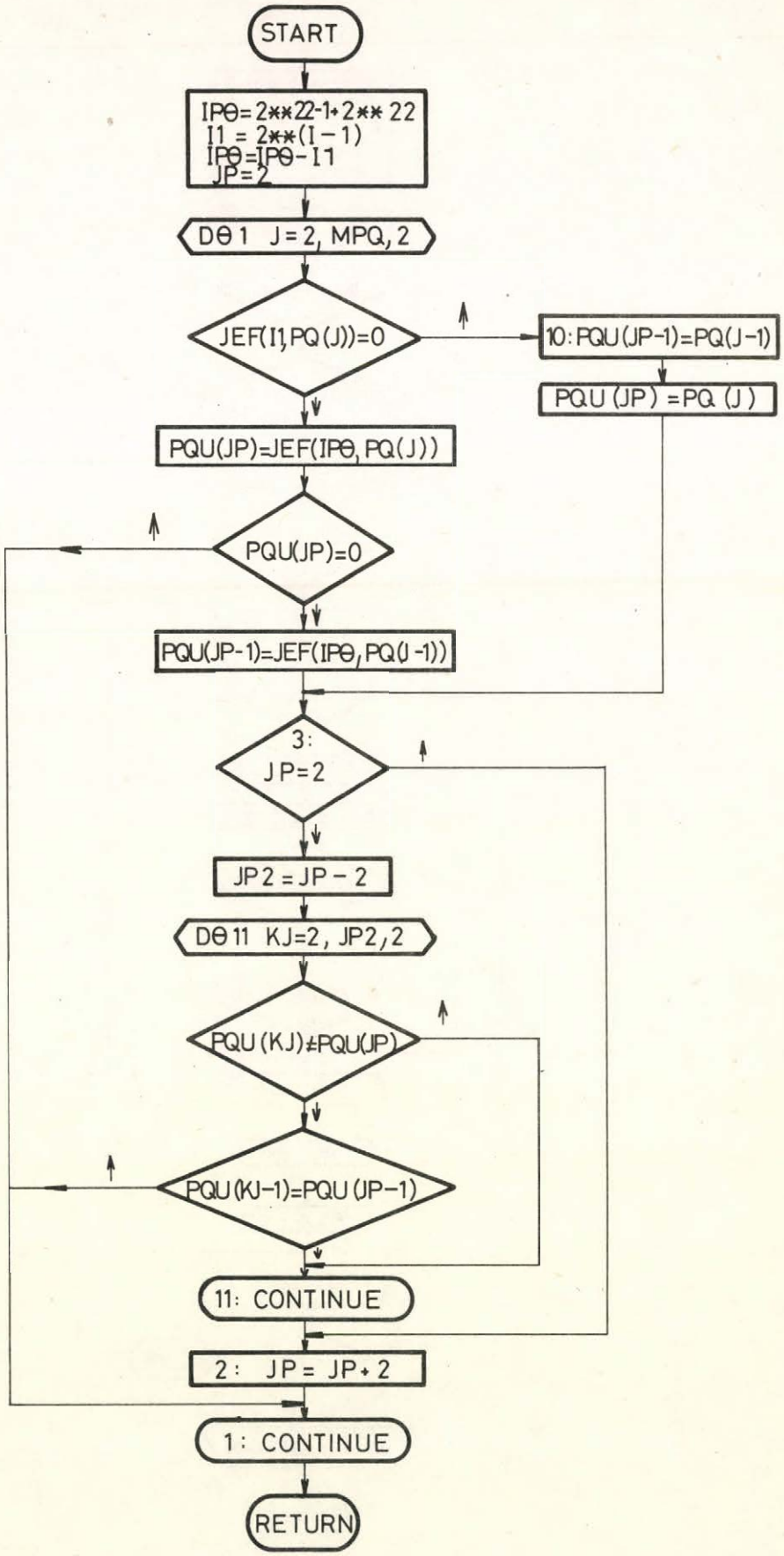
Az algoritmus blokkdiagrammja: 1.sz. melléklet




START
MUV ≠ -1
NEGIR = NF(3,1)
NF(3,1)=NF(3,2)
NF(3,2)=NEGIR
A NEGÁCIO EREDMÉNYE
10: MT(1) = IV
JK = 0
DO 1 I = 1,2
NF(3,I)≠0
JK=1
1: CONTINUE
JK=0
100: INDEX NULLA ELHDNF-BEN
RETURN
MIND=3
MTI = 2
2: IKEZD = NF (6, JK)
IVEG = NF (7, JK)
MT(MTI)=(IVEG -IKEZD+1)*2
MTI = MTI + 1
DO 3 J=IKEZD, IVEG
MT(MIND+1)= K1(J)
MT(MIND+2)= K2(J)
MIND=MIND + 2
3: CONTINUE
MTI = 4
JK = JK-1
JK = 0
JK= 2
5: ISEG=MT(2)*MT(3)+3
4*ISEG>10000
1006: KEVES A MEMORIA
STOP
REWIND 5
WRITE(5) MT
***** AZ MT TOMB *****
REWIND 5
READ(5) MT
DO 7 IS=1, ISEG
7: CALL BINIK (MT(IS),WORDL)
RETURN

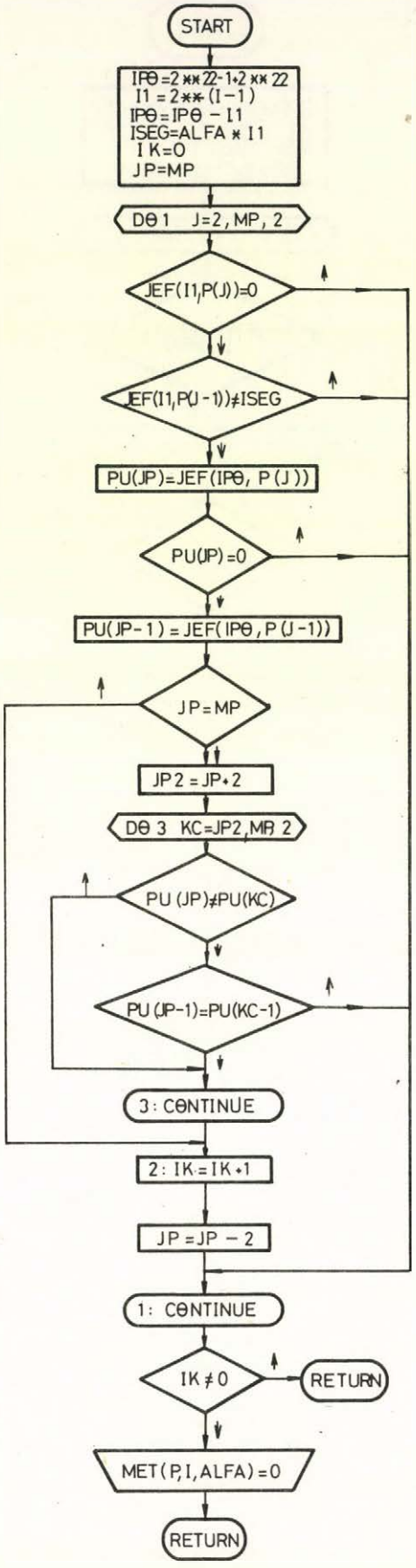
SUBROUTINE PQREX (PQ,PQU,MPQ,I,JP)

3. sz. melléklet



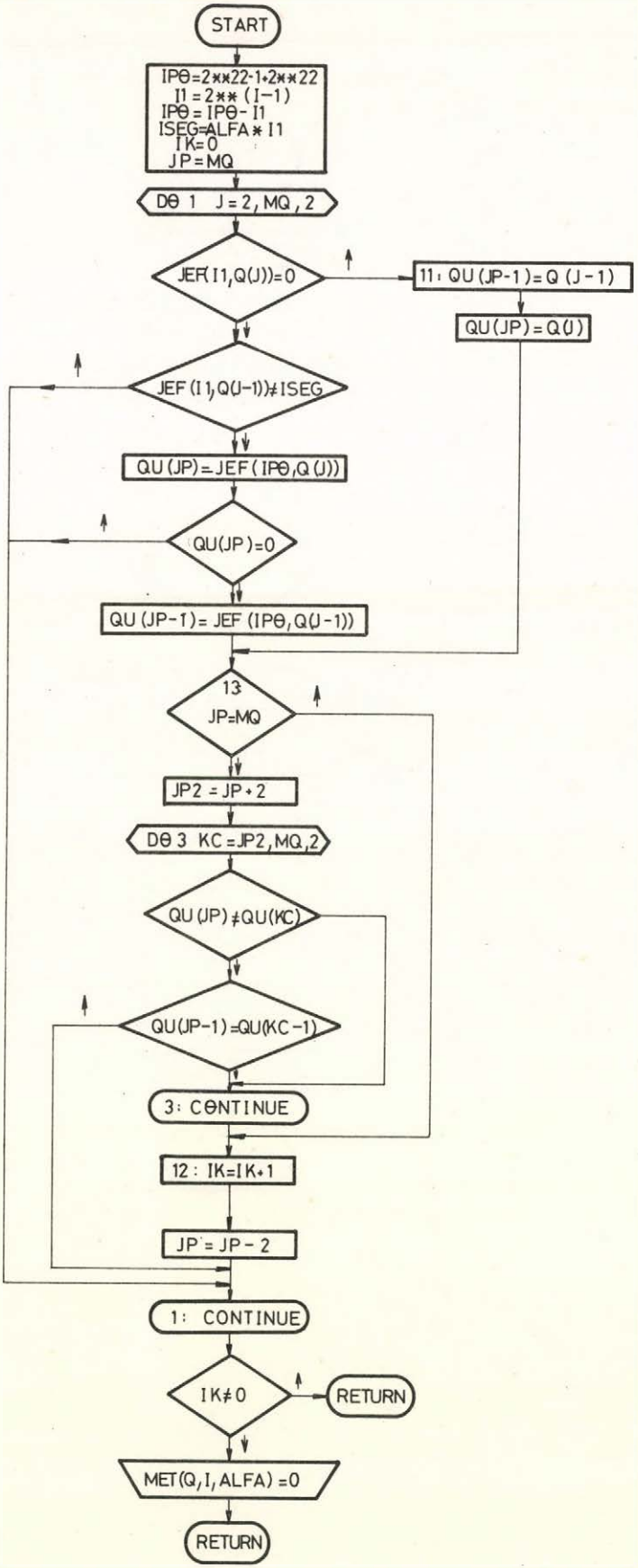
SUBROUTINE PMET (P, PU, MP, I, ALFA, IK)

4. sz. melléklet

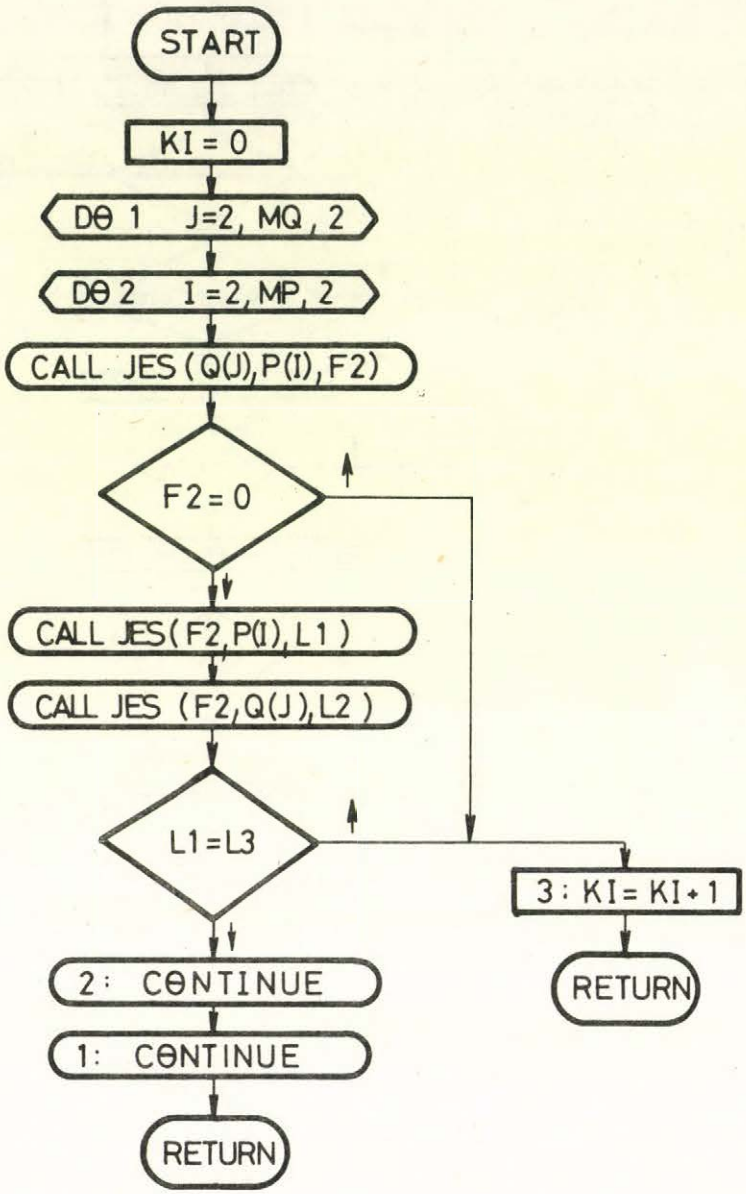


SUBROUTINE QMET (Q, QU, MQ, I, ALFA, IK)

5. sz. melléklet

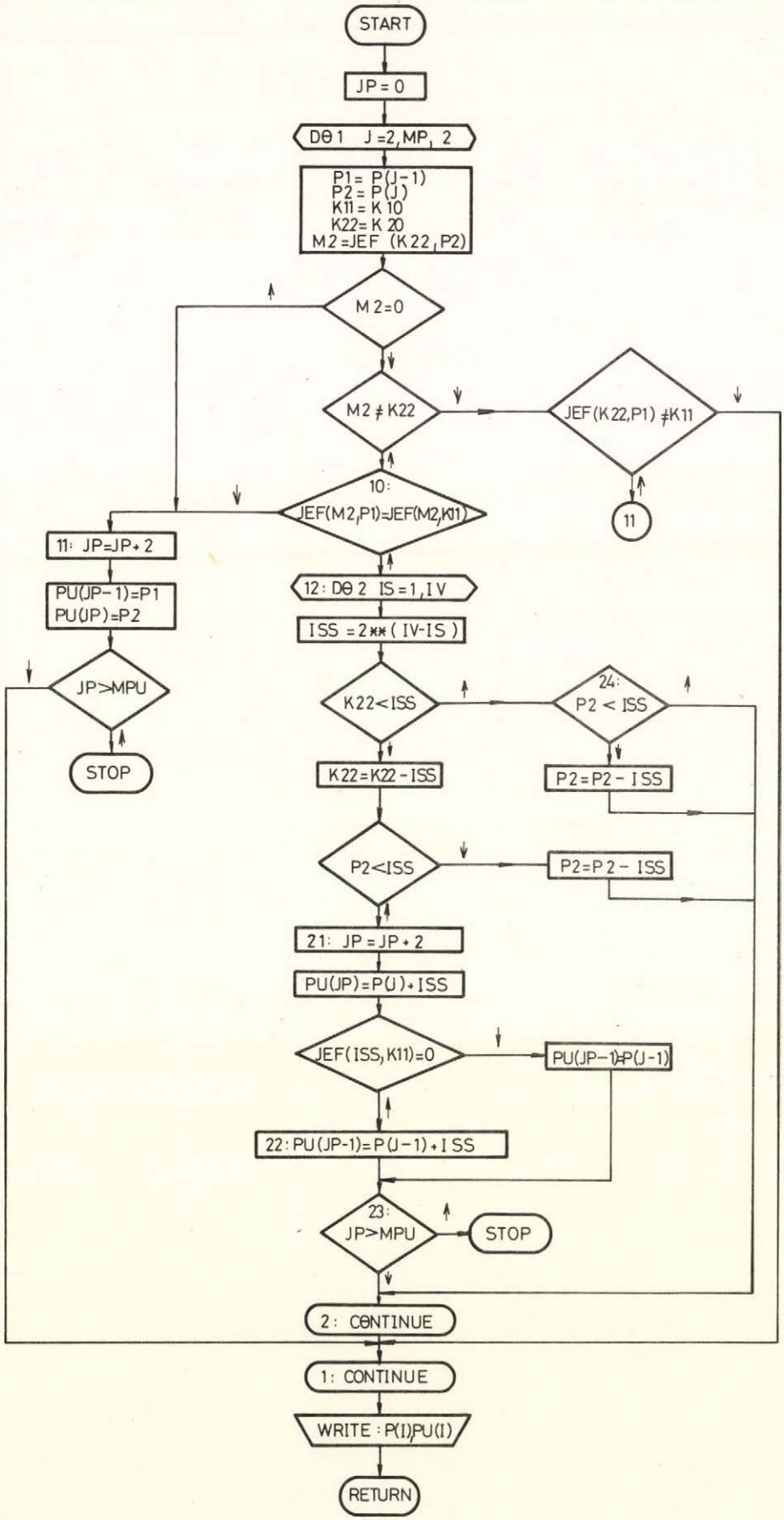


SUBROUTINE PQURES (P,Q, MP, MQ, KI)



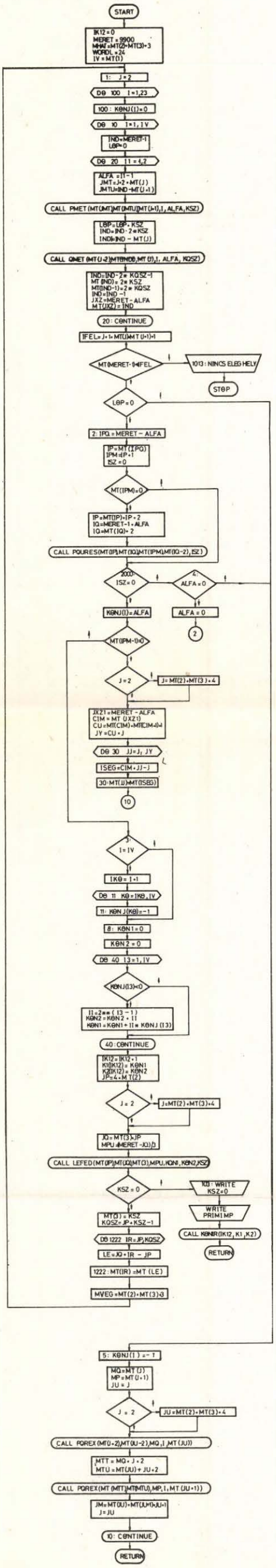
SUBROUTINE LEFED (P, PU, MP, MPU, K10, K20, JP)

7. sz. melléklet



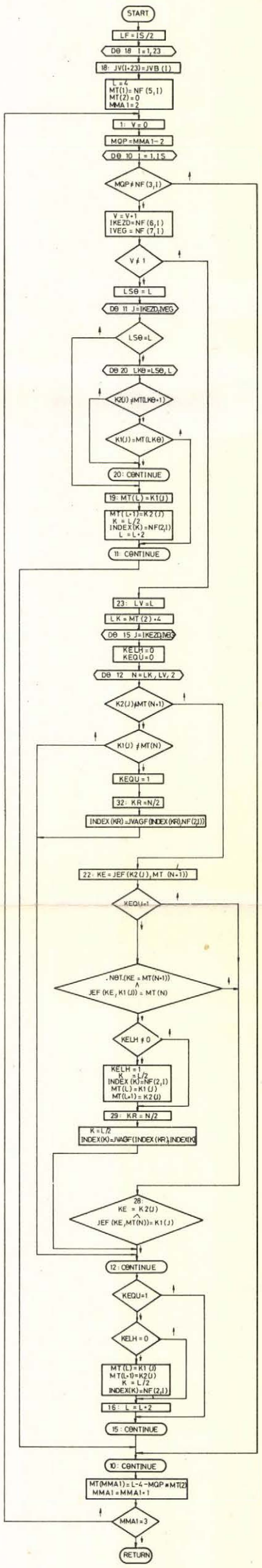
SUBROUTINE REKEF

3. sz. melléklet



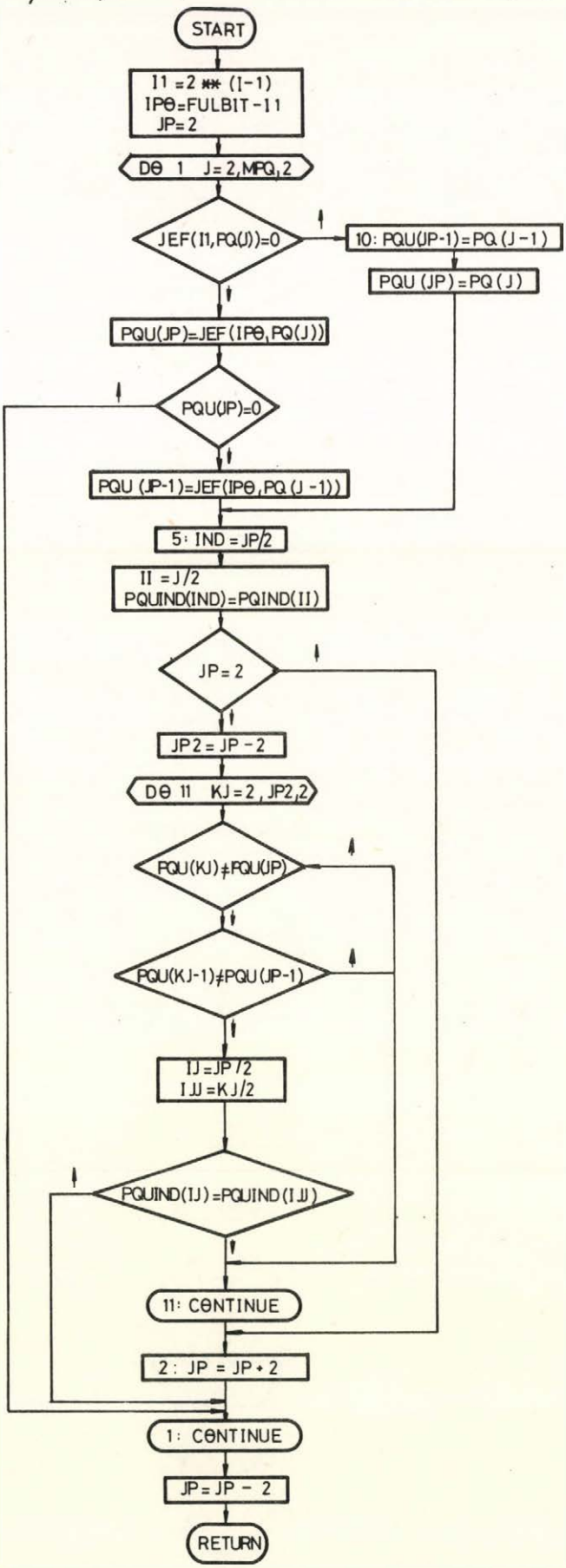
SUBROUTINE TDNFEL

9. sz. melléklet

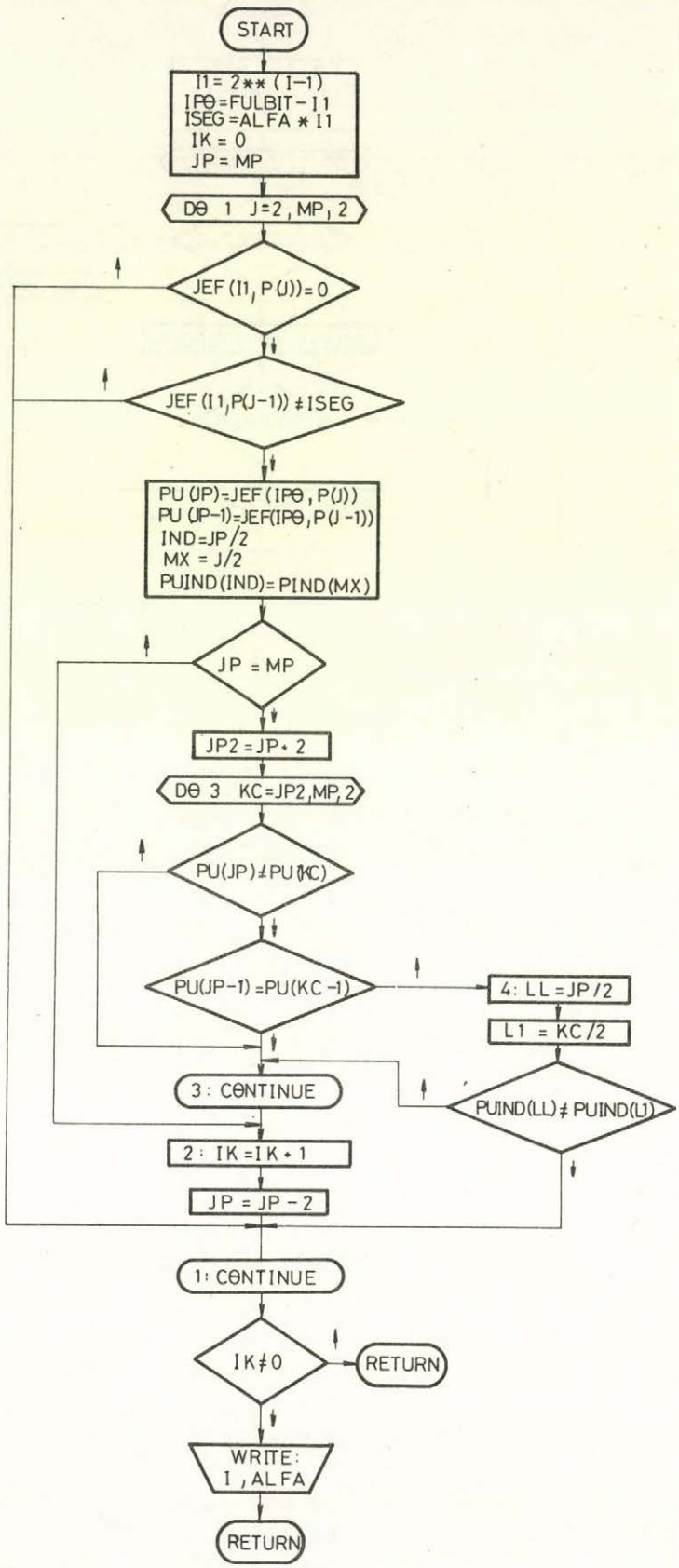


SUBROUTINE REKTFV

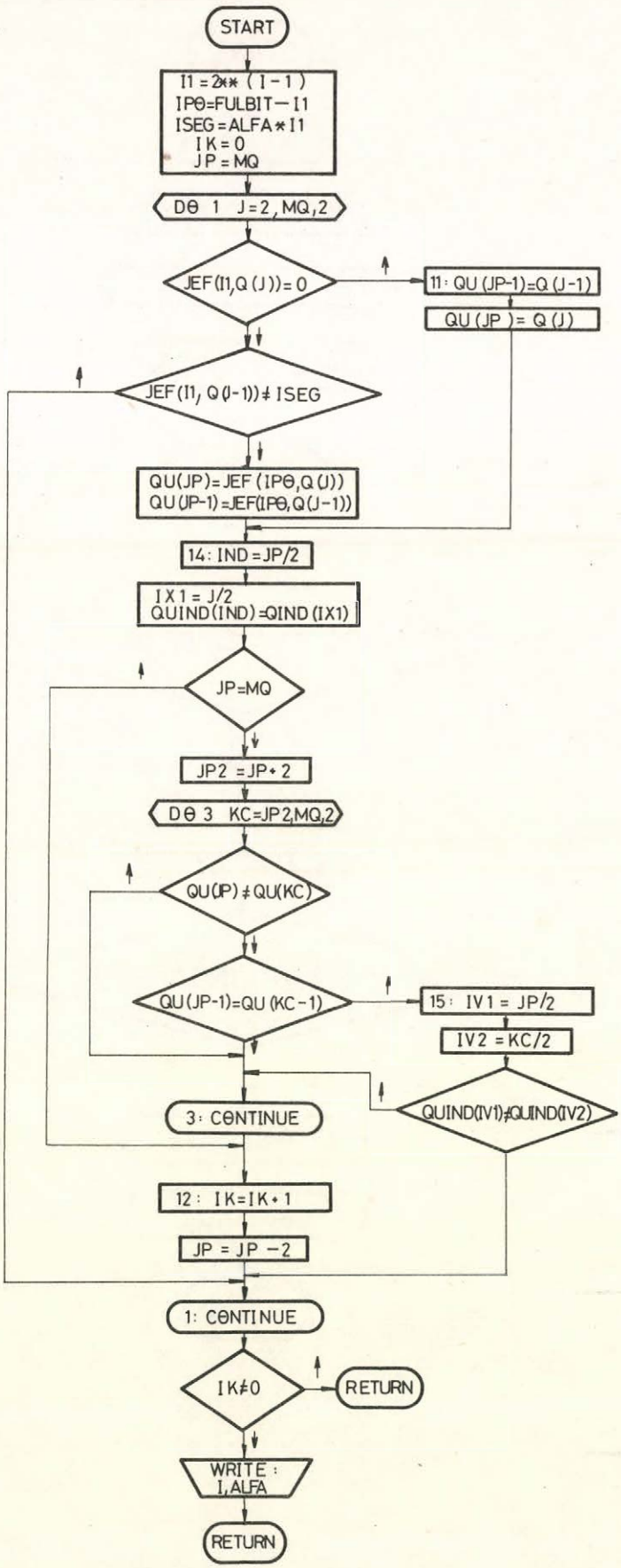
SUBRØUTINE PQTREX (PQ, PQU, MPQ, PQIND, PQUIND, MPQIND, I, JP)



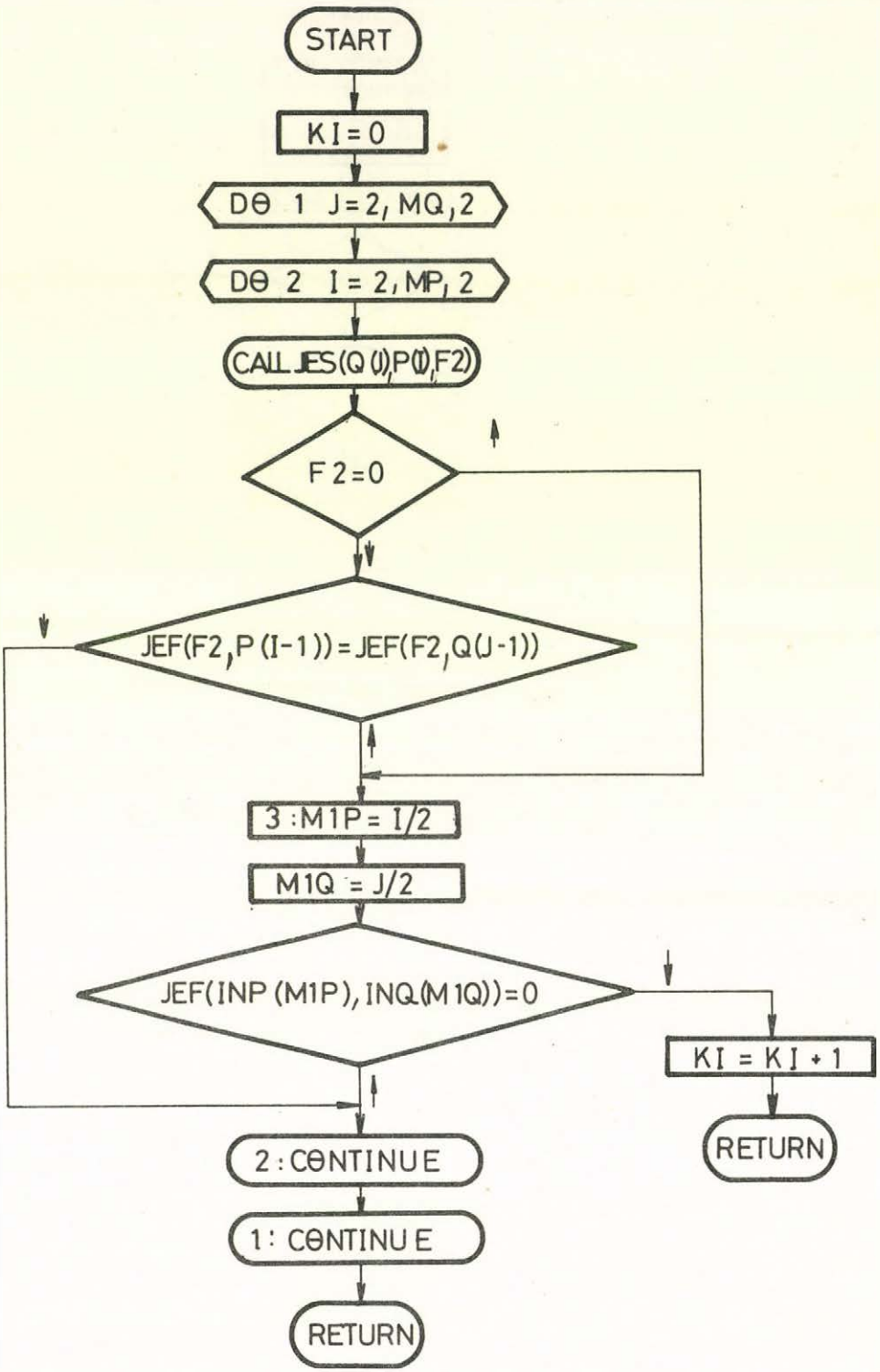
SUBROUTINE PTMET (P, PU, MP, PIND, PUIND, MPIND, I, ALFA, IK)



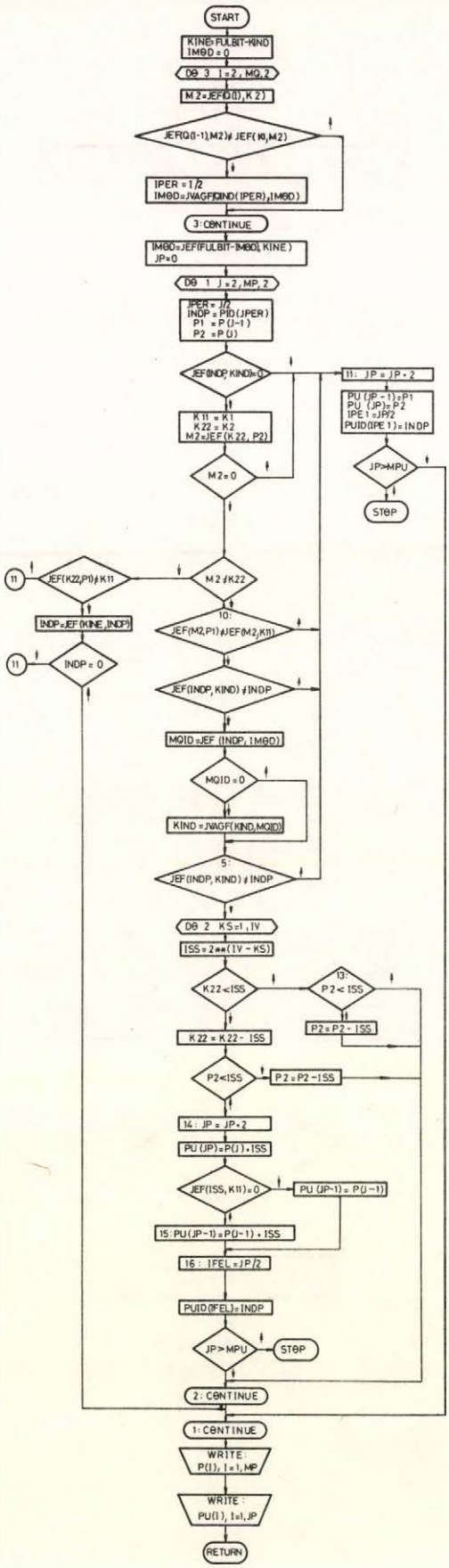
SUBRØUTINE QTMET (Q, QU, MQ, QIND, QUIND, MQIND, I , ALFA , IK)



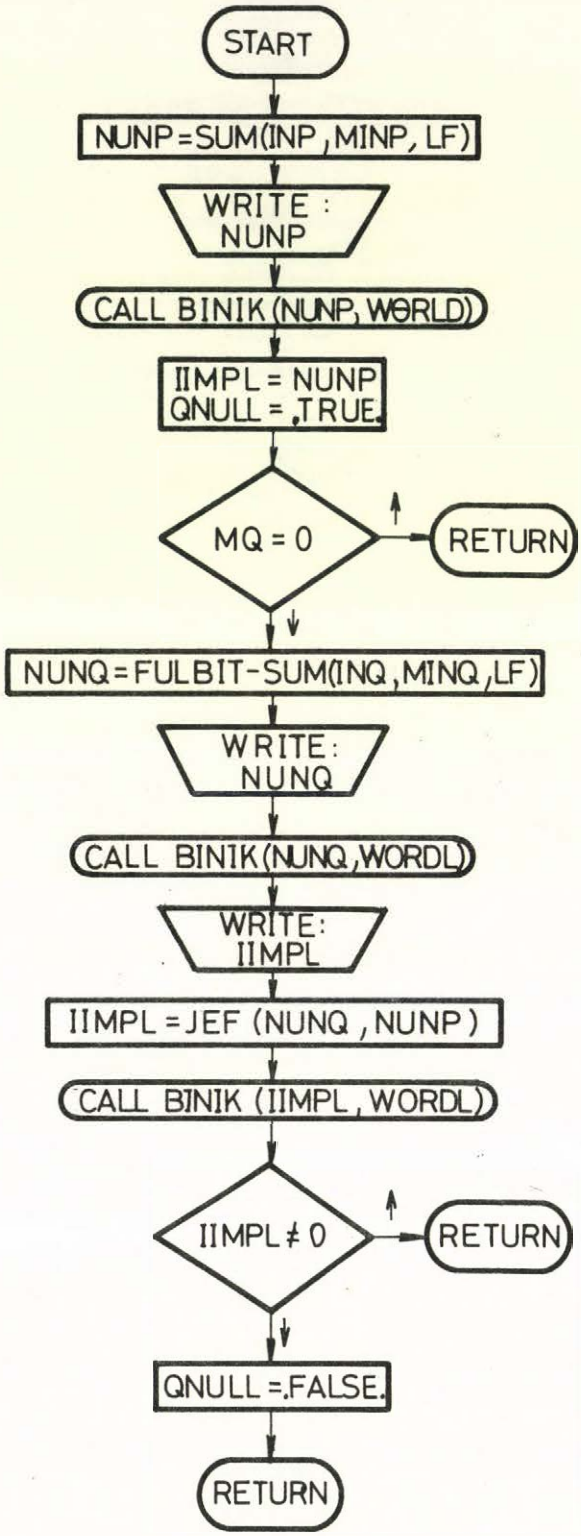
SUBROUTINE PQTUR (P, Q, MP, MQ, INP, INQ, MINP, MINQ, KI)



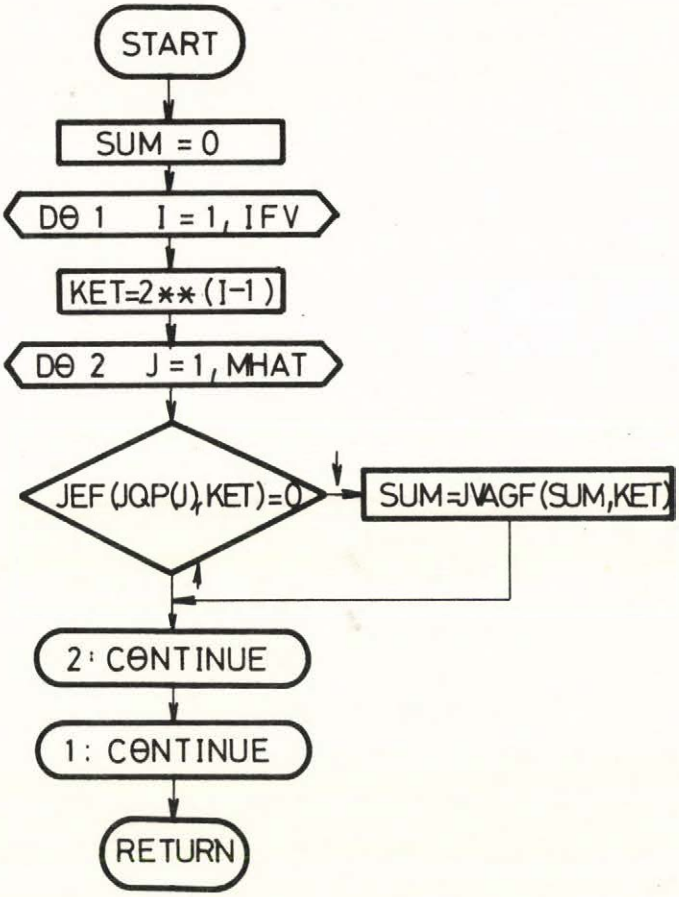
SUBROUTINE TLEFED (P, PU, MP, MPU, K1, K2, PID, MPID, PUID, MPUID, KIND, JP, Q, MQ, QIND, MQIND)



LOGICAL FUNCTION QNULL (MQ, INQ, MINQ, INP, MINP, IIMPL)



INTEGER FUNCTION SUM (JQP, MHAT, IFV)



SUBROUTINE KONIRT (KSZ, KQ1, KQ2, KQIND, JEL) 13. sz. melléklet