Minimizarea sistemelor decizionale multivalente deterministe şi nedeterministe

Teză destinată obţinerii
titlului ştiinţific de doctor inginer
la
Universitatea „Politehnica” din Timişoara
în domeniul „Ştiinţa calculatoarelor”
de către

Ing. Adrian Zafiu

Conducător ştiinţific:	prof.univ.dr.ing. Ştefan Holban
Referenţi ştiinţifici:	prof.univ.dr.ing. Ştefan Pentiuc
	prof.univ.dr.ing. Ilie Popa
	prof.univ.dr.ing. Horia Ciocârlie

Ziua susţinerii tezei: 05.07.2008

Seriile Teze de doctorat ale UPT sunt:

1. Automatică
2. Chimie
3. Energetică
4. Ingineria Chimică
5. Inginerie Civilă
6. Inginerie Electrică
7. Inginerie Electronică şi Telecomunicaţii
8. Inginerie Industrială
9. Inginerie Mecanică
10. Ştiinţa Calculatoarelor
11. Ştiinţa şi Ingineria Materialelor

Universitatea „Politehnica” din Timişoara a iniţiat seriile de mai sus în scopul diseminării expertizei, cunoştinţelor şi rezultatelor cercetărilor întreprinse în cadrul şcolii doctorale a universităţii. Seriile conţin, potrivit H.B.Ex.S Nr. 14 / 14.07.2006, tezele de doctorat susţinute în universitate începând cu 1 octombrie 2006.

România, 300159 Timişoara, Bd. Republicii 9,
tel. 0256 403823, fax. 0256 403221
e-mail: editura@edipol.upt.ro

Cuvânt înainte

Procesul de minimizare are ca scop transcrierea unei specificaţii date a unui sistem decizional într-o specificaţie echivalentă, redusă din punct de vedere al complexităţii de exprimare, fără a denatura comportamentul sistemului, cu scopul de a obţine implementări cu cost redus. Minimizarea funcţiilor logice multivalente este intens studiată datorită numeroaselor domenii în care sunt utilizate rezultatele acestor cercetări. Problema minimizării este deschisă deoarece nu a fost descoperită o metodă care să minimizeze specificaţiile multivalente într-un timp polinomial.

La ora actuală nu există o aplicaţie care să surprindă în mod unitar toate problemele legate de minimizarea funcţiilor logice multivalente. Fiecare dintre metodele existente tratează numai o subclasă de sisteme decizionale sau soluţionează parţial minimizarea, fără a pune la dispoziţie toate modelele de operare.

În cadrul acestei teze am elaborat un sistem unitar bazat pe determinarea disjuncţiilor pentru minimizarea funcţiilor multivalente complet sau incomplet specificate, deterministe sau nedeterministe, cu ieşiri corelate sau necorelate, care să pună la dispoziţia utilizatorului diverse strategii de procesare în concordanţă cu particularităţile fiecărei probleme în parte.

Pentru realizarea actualei lucrări doresc să aduc alese mulţumiri conducătorului ştiinţific, domnului prof. univ. dr. ing. Ştefan Holban pentru sprijinul şi competenta îndrumare acordată pe întreaga perioadă a elaborării tezei.

Îmi exprim întreaga consideraţie faţă membrii comisiei de doctorat, domnul preşedinte al comisiei prof. univ. dr. ing. Mircea Stratulat, prodecan al Facultăţii de Automatică şi Calculatoare Timişoara şi domnii prof. univ. dr. ing. Ştefan Pentriuc de la Universitatea „Ştefan cel Mare” din Suceava, prof. univ. dr. ing. Ilie Popa de la Universitatea din Piteşti şi prof. univ. dr. ing. Horia Ciocârlie de la Universitatea „Politehnica” Timişoara, care au răspuns solicitării de a face parte din comisia de analiză a tezei, pentru observaţiile făcute şi pentru timpul acordat lucrării.

Nu în ultimul rând, doresc să mulţumesc domnului prof. univ. dr. ing. Ion Ştefănescu de la Universitatea din Piteşti pentru tot sprijinul acordat pe parcursul elaborării acestei teze.

Piteşti, iulie 2008 Adrian Zafiu

Fiului, soţiei şi mamei

Zafiu, Adrian
Minimizarea sistemelor decizionale multivalente deterministe Şi nedeterministe

Teze de doctorat ale UPT, Seria , Nr. , Editura Politehnica, 2008, pagini, figuri, tabele.
ISSN:
ISBN:
Cuvinte cheie:
minimizarea sistemelor decizionale, logică multivalentă, principii de minimizare, strategii de minimizare

Rezumat:

Abordările existente în domeniul minimizării sistemelor decizionale sunt strâns legate de modul de implementare al acestora. Pe măsură ce suportul hardware Şi software evoluează, sunt necesare tehnici analitice din ce în ce mai performante, care să permită procesarea sistemelor decizionale complexe (atât din punct de vedere al logicii integrate cât Şi al dimensiunii) Şi integrarea conceptelor teoretice (multivalenţă, nedeterminism, funcţii incomplet specificate, funcţii cu ieŞiri corelate) în aplicaţiile dedicate minimizării.

Calitatea procesului de minimizare este dată de efortul necesar implementării sistemului decizional rezultat. Utilizarea unui sistem este cu atât mai răspândită cu cât cantitatea sau costul componentelor utilizate pentru implementare sunt mai reduse. Reducerea costurilor se poate face prin descoperirea unei forme de reprezentare echivalente care să implice utilizarea unui număr mai mic de componente. Pentru sistemele hardware este evidentă necesitatea minimizării. În cazul sistemelor software minimizarea se traduce prin optimizarea algoritmilor în sensul reducerii numărului de componente decizionale.

Cercetările efectuate în cadrul tezei au urmărit crearea unui sistem unitar de minimizare a sistemelor multivalente deterministe şi nedeterministe ce poate fi aplicat atât asupra sistemelor hardware cât şi software.

Cuprins

Listă de tabele

Listă de figuri

1 Introducere

Rădăcinile minimizării logice pot fi urmărite până la tratatul de logică al lui George Bool (1815-1864) care defineşte termenul de algebră booleană. În 1938, Claude Shannon a arătat că operaţiile unui circuit în comutaţie pot fi descrise cu ajutorul algebrei booleane. Primele abordări ale domeniului minimizării logice au utilizat reprezentările grafice. Minimizarea logicii binare a fost aplicată asupra tabelelor de adevăr reprezentate prin diagrame Karnaugh. Minimizarea bazată pe diagrame este guvernată de un set de reguli ce determină modul în care pot fi combinate intrările din aceasta. Utilizarea diagramelor în proiectare este realistă pentru funcţii cu un număr mic de variabile (uzual între 4 şi 6). Pentru funcţiile cu un număr mai mare de variabile a fost necesară găsirea unor metode analitice de minimizare.

Primul pas în automatizarea procesului a fost făcut prin introducerea metodei Quine-McCluskey, aceasta putând fi implementată software. Metodele care au permis abordarea sistematică a problemei minimizării erau grafo-analitice sau pur analitice. Utilizarea unor tehnici mai laborioase pentru abordarea unor probleme mai complexe depindea direct de puterea de calcul a sistemelor disponibile.

Revigorarea majoră a domeniului a fost marcată de descoperirea reprezentării sistemelor decizionale cu ajutorul diagramelor de decizie. Cercetările din domeniu s-au axat pe elaborarea de principii, strategii, metode, tehnici şi algoritmi care au la bază diagramele de decizie. În prezent, metoda Espresso este considerată cea mai eficientă metodă de sinteză pentru funcţii binare. Logica multivalentă a apărut ca o extensie naturală a logicii binare. Funcţiile multivalente pot fi minimizate prin codarea binară a valorilor multivalente [1] sau fără codare prin metode pur multivalente. MVSIS este considerată cea mai avansată aplicaţie pentru minimizarea sistemelor multivalente.

În ultimii ani au apărut şi alte abordări care au la bază reprezentarea tabelară a sistemelor decizionale. Această reprezentare a impus alte principii şi strategii, conducând la obţinerea unor noi tehnici şi metode care completează sau substituie algoritmi bazaţi pe diagrame de decizie. Singura metodă de minimizare a funcţiilor logice multivalente bazată pe reprezentări tabelare este metoda discriminării.

Metoda discriminării este o metodă generalizată pentru minimizarea, optimizarea şi sinteza de circuite logice multivalente (cu mai mult de două valenţe pe fir). Faţă de Espresso, MV şi MVSIS, metoda discriminării operează direct cu variabile multivalente. ESPRESSO impune o codare binară a valorilor multivalente, ceea ce implică existenţa unei serii de

combinaţii neutilizate. Acestea conduc la o utilizare mai puţin eficientă a memoriei (nu se poate mapa perfect o valoare multivalentă cu 5 valente în binar - este necesară utilizarea a 3 biţi care definesc 8 configuraţii din care 3 nu vor fi folosite). Metodele MVSIS I şi MVSIS II utilizează diagramele de decizie multivaloare (noţiunea de valoare este înlocuită cu o mulţime de valori) pentru a îngloba unitar conceptul de nedeterminism. Acest fapt conduce la o complexitate crescută a sistemului deoarece numărul de valori posibile pentru o variabilă cu n valori este $2^n - 1$.

Procesul de minimizare are ca scop transcrierea unei specificaţii date a unui sistem decizional într-o specificaţie echivalentă, redusă din punct de vedere al complexităţii de exprimare, fără a denatura comportamentul sistemului, cu scopul de a obţine implementări cu cost redus. Problema minimizării este deschisă deoarece nu a fost descoperită o metodă care să minimizeze specificaţiile multivalente într-un timp polinomial.

Minimizarea funcţiilor logice multivalente (sisteme decizionale multivalente) este intens cercetată datorită numeroaselor domenii în care pot fi folosite rezultatele acestor cercetări. Domeniile vizate dezvoltă sisteme formate din componente decizionale şi operaţionale. În procesul de optimizare a unui sistem se poate acţiona atât asupra părţii decizionale cât şi a celei operaţionale.

1.1 Motivaţie

Abordările existente sunt strâns legate de modul de implementare al sistemelor decizionale (modul de reprezentare al valorilor şi modul de operare asupra acestora). Pe măsură ce suportul hardware şi software evoluează, sunt necesare tehnici analitice din ce în ce mai performante, care să permită integrarea conceptelor teoretice (multivalenţă, nedeterminism, funcţii incomplet specificate, funcţii cu ieşiri corelate) precum şi rezolvarea problemelor din ce în ce mai complexe (cu un număr tot mai mare de date iniţiale).

Calitatea rezultatului unui proces de minimizare este dată de efortul necesar implementării sistemului decizional. Cu cât cantitatea sau costul componentelor utilizate pentru implementarea unui sistem sunt mai mici, cu atât este mai răspândită utilizarea respectivului sistem. Sistemele devin din ce în ce mai complexe, integrând o logică din ce în ce mai evoluată, ceea ce implică o creştere a costurilor de utilizare. Reducerea costurilor se poate face prin reducerea costurilor elementelor componente ale sistemului dar şi prin descoperirea unei forme de reprezentare adecvate, echivalente, care să implice utilizarea unui număr mai mic de componente în implementare. Cele de mai sus sunt evidente pentru sistemele hardware. Pentru sistemele software acestea corespund utilizării unor algoritmi performanţi din punct de vedere al timpului de procesare (de exemplu numărul de teste ce trebuie efectuat pentru a afla rezultatul unei funcţii este minim) şi utilizării unor structuri de date care să faciliteze procesarea.

La ora actuală nu există o aplicaţie care să surprindă în mod unitar toate problemele legate de minimizarea funcţiilor logice multivalente. Fiecare dintre metodele existente tratează numai o subclasă de sisteme decizionale sau soluţionează parţial minimizarea, fără a pune la dispoziţie toate modelele de operare. Este necesară o aprofundare a domeniului minimizării funcţiilor logice multivalente prin care să fie determinate operaţiile ce pot fi efectuate asupra acestora.

1.2 Obiective

Obiectivul propus al acestui demers este crearea unui sistem unitar, bazat pe metoda discriminării, pentru minimizarea funcţiilor multivalente complet sau incomplet specificate, deterministe sau nedeterministe, cu ieşiri corelate sau necorelate, care să pună la dispoziţia utilizatorului diverse strategii ce pot fi selectate în concordanţă cu particularităţile fiecărei probleme în parte.

Atingerea obiectivului principal este posibilă prin urmărirea unei secvenţe de obiective secundare:

- specificarea nativă a problemelor decizionale multivalente în software şi hardware;
- eliminarea graniţei dintre hardware şi software prin descrierea unitară a sistemelor software şi hardware;
- crearea unui sistem de procesare paralelă prin:
  - minimizarea simultană a ieşirilor multiple;
  - eliminarea redundanţei în minimizarea paralelă;
- crearea unui sistem pentru minimizarea în reţea, eliminând calculele redundante;
- stabilirea unui mecanism de tratare a valorilor implicite:
  - pentru reducerea dimensiunii rezultatelor;
  - pentru a crea o metodă euristică performantă;
- tratarea automată (algoritmică) a redundanţelor din specificările descriptive:
  - determinarea redundanţelor;
  - eliminarea redundanţelor;
- tratarea hazardului combinaţional:
  - clasificarea analitică a hazardului în logic sau funcţional;
  - determinarea unei soluţii algoritmice pentru eliminarea hazardului logic;
  - anunţarea hazardului funcţional.

În acest demers am pornit de la metoda discriminării deoarece corespunde cel mai bine obiectivelor propuse. Etapele parcurse pentru atingerea obiectivului urmărit sunt:

1. Conceperea unui algoritm pentru metoda discriminării, necesar pentru analiza acestei metode în comparaţie cu cele existente.

Studiul vizează calitatea rezultatelor, analiza performanţelor şi clasificarea problemelor ce pot fi rezolvate.

2. Informaţiile obţinute urmează a fi utilizate pentru realizarea unui model matematic consistent, care să ofere un suport acoperitor pentru minimizarea combinaţională multivalentă. Modelarea matematică va integra conceptele dezvoltate în cadrul metodei discriminării, tipurile de date şi operatorii specifici, astfel încât să permită prezentarea omogenă a metodelor şi algoritmilor ce urmează a fi implementaţi.
3. Studierea strategiilor de minimizare, modelarea lor cu ajutorul aparatului matematic şi evaluarea performanţelor acestora în funcţie de caracteristicile problemelor abordate. Testarea strategiilor are ca scop secundar determinarea unor metode rapide. Se va urmări în special obţinerea unor rezultate superioare din punct de vedere calitativ în raport cu alte strategii şi cu alţi algoritmi de minimizare.
4. Pe baza rezultatelor obţinute, va urma definirea unui set coerent de principii şi strategii care să permită dezvoltarea unor tehnici performante din punct de vedere al rezultatelor minimizării şi al opţiunilor puse la dispoziţia utilizatorilor.
5. În final, principiile şi strategiile stabilite vor fi utilizate în dezvoltarea unui set de aplicaţii care să vizeze atât aspectul calitativ al rezultatelor minimizării cât şi aspectul performanţei pentru procesarea unor cantităţi mari de informaţii. În cadrul aplicaţiilor dezvoltate se va pune accent şi pe performanţele tehnicilor şi algoritmilor utilizaţi.

Atingerea obiectivului propus deschide calea pentru crearea unui aparat de minimizare a sistemelor secvenţiale şi a dezvoltării de metodologii în domeniile proiectării logice şi sintezei hardware [2][3] şi software.

1.3 Prezentarea conţinutului

Lucrarea este structurată în 10 capitole. Elementul central este aplicaţia **COMIN** (**CO**mbinational **MIN**imization) implementată pentru minimizarea sistemelor decizionale.

În **Capitolul 1** sunt prezentate: un scurt istoric al minimizării, problematica acestuia, motivaţia prezentului demers, obiectivul principal şi cele secundare propuse şi etapele cercetării. În finalul primului capitol este prezentată structura tezei.

Capitolul 2 conţine prezentarea succintă domeniului sistemelor logice şi problemele legate de interpretarea specificărilor tabelare. Tot aici se regăseşte terminologia utilizată pe parcursul lucrării, structura formală a reprezentării tabelare a funcţiilor şi a librăriilor de funcţii, notaţiile şi semantica notaţională utilizată. În finalul capitolului prezint o analiză

succintă care plasează procesul de minimizare în problematica mai generală a optimizării sistemelor decizionale.

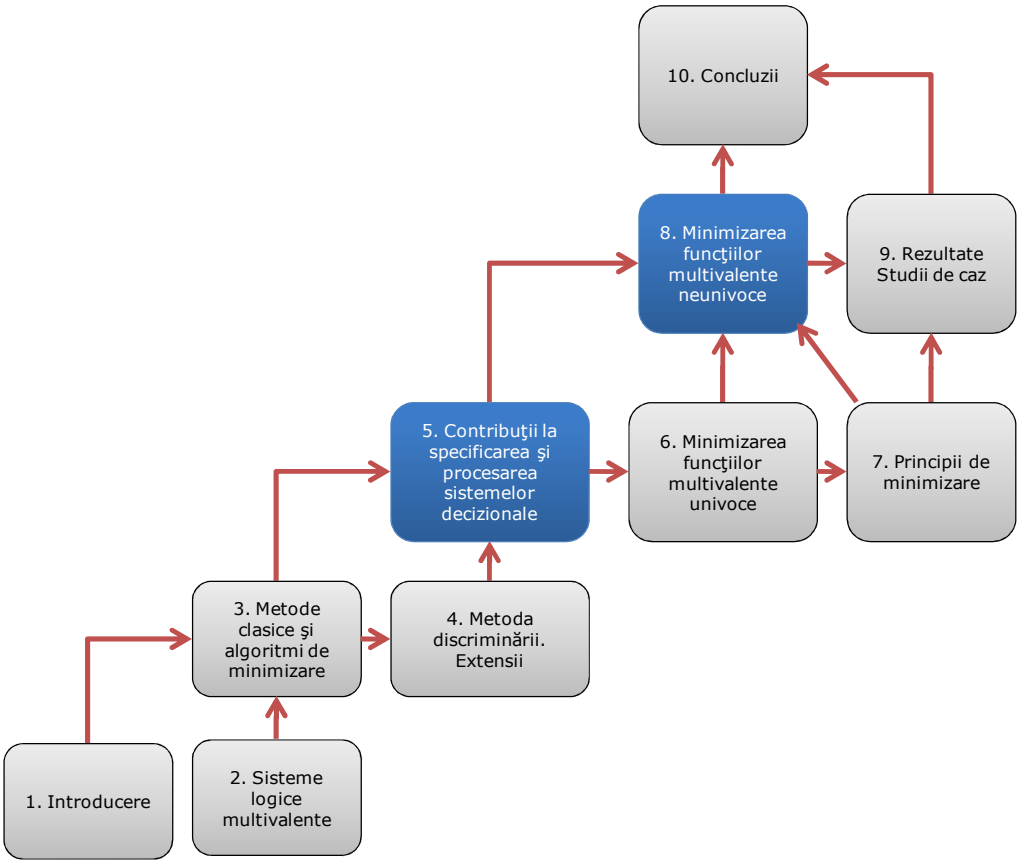


Fig. 1-1 Structura lucrării

Pentru a studia metoda discriminării am aprofundat tehnicile deja existente în teoria sintezei de circuite combinaţionale. În **capitolul 3** am prezentat o analiză a metodelor utilizate pentru minimizarea sistemelor decizionale binare şi multivalente:

- sistemele grafice (diagrame Veitch-Karnaugh) [4];
- metoda Quine McCluskey [5][6];
- aplicaţia Espresso [7][8] pentru sistemele binare,;
- aplicaţia SIS [9] care permite abordarea sistemelor multivalente prin codarea în binar;
- aplicaţia MVSIS [10][11] pentru minimizarea sistemelor multivalente;
- aplicaţiile BOOM-I [12] şi BOOM-II [13] care realizează o implementare euristică performantă din punct de vedere al timpului de execuţie şi cu rezultate deosebite în minimizarea sistemelor binare;

- metoda Discriminării [14], ultima cunoscută în acest domeniu.

În **capitolul 4** am prezentat metoda discriminării [15], la a cărei dezvoltare şi implementare am participat. Prima implementare a facilitat rularea unui număr mare de teste şi a condus, implicit, la evidenţierea unor aspecte teoretice noi, specifice problematicii minimizării. Observaţiile care au adus elemente noi au fost utilizate pentru a rafina metoda. Forma definitivă a metodei discriminării, prezentată în acest capitol, a fost obţinută după mai multe iteraţii. În finalul capitolului am propus câteva extensii pentru metoda discriminării:

- extinderea metodei pentru a accepta specificări incomplete, ambigue sau semi-optimizate;
- determinarea automată a ambiguităţii unei specificări;
- tehnici de eliminare automată a ambiguităţilor;
- determinarea, clasificarea, eliminarea acolo unde este posibil şi anunţarea hazardului neeliminat.

În **capitolul 5** am definit un aparat matematic care asigură suport de operare asupra logicii multivalente. Algebra multivalentă prezentată asigură un set de operatori care acoperă metoda discriminării şi extensiile propuse. Am modelat cu ajutorul operatorilor definiţi procesările specifice fiecărei etape parcurse în cadrul metodei discriminării, ceea ce a permis analiza metodei din punct de vedere al complexităţii calculului. Algebra elaborată, fiind o generalizare a algebrelor binare, acoperă şi cazul particular al specificaţiilor binare. Pe parcursul dezvoltării modelului matematic am urmărit definirea unui set complet de operatori care să acopere toate cazurile de utilizare ale modelului apărute în lucrare.

Metodele preliminare implementate pentru a verifica şi valida modelul matematic dezvoltat pe baza principiului discriminării sunt prezentate în **capitolul 6**. Am descris 3 metode de minimizare complet diferite din punct de vedere al abordării problemei:

- în prima metodă am utilizat proprietăţile vectorilor consens pentru determinarea mulţimii vectorilor implicanţi.
- în a doua metodă am utilizat operaţia de complementare pentru a determina mulţimea vectorilor implicanţi maximali.
- în a treia metodă am utilizat o strategie de căutare de tip A* şi am eliminat o parte din dezavantajele primelor două metode.

Analiza avantajelor şi dezavantajelor acestor metode, a condus la definirea unui set de principii de minimizare şi a unui pachet de strategii ce pot fi utilizate pentru procesarea funcţiilor multivalente. Principiile şi strategiile de minimizare deduse sunt prezentate în **capitolul 7**.

În **capitolul 8** am prezentat aplicaţia finală, **COMIN**, care permite minimizarea funcţiilor multivalente deterministe sau nedeterministe, complet sau incomplet specificate cu ieşiri corelate sau necorelate. Metoda utilizează forma de reprezentare tabelară a datelor şi o tehnică de abordare proprie, bazată pe modelul matematic elaborat. Forma finală a fost obţinută după mai multe iteraţii deoarece nu exista o metodă matură de referinţă de minimizare care să utilizeze reprezentarea tabelară. După fiecare iteraţie au fost eliminate neajunsurile observate în pasul anterior sau a fost crescut

gradul de generalitate al metodei. Aplicaţia de minimizare obţinută acoperă atât obiectivul principal cât şi obiectivele secundare propuse. Metoda prezentată a fost implementată ca un sistem deschis care poate fi actualizat astfel încât să integreze simultan diverse strategii de minimizare.

Calitatea rezultatelor minimizărilor funcţiilor multivalente obţinute cu aplicaţia **COMIN** sunt prezentate în **capitolul 9**. Sunt evidenţiate avantajele metodei dezvoltate exprimate atât cantitativ cât şi calitativ în raport cu rezultatele obţinute cu metoda **MVSIS**, singura metodă multivalentă disponibilă la ora actuală.

Capitolul 10 conţine concluziile desprinse pe parcursul studierii şi elaborării metodelor şi implementărilor prezentate în această lucrare, rezumatul contribuţiilor şi o serie de direcţii de dezvoltare ulterioară a cercetărilor în domeniul minimizării sistemelor multivalente utilizând principiile prezentate în **capitolul 7**.

2 Sistemele logice multivalente

- terminologie, specificare şi reprezentare-

În acest capitol realizez o clasificare a sistemelor logice, prezint interpretările ce pot fi atribuite sistemelor multivalente, modul tabelar de reprezentare al sistemelor utilizat pe parcursul lucrării şi valorile speciale utilizate în cadrul acestuia, semantica ce se asociază acestor reprezentări şi, în finalul capitolului, fac o analiză succintă a ciclului proiectare-implementare care plasează procesul de minimizare în problematica mai generală a optimizării sistemelor decizionale.

Din punctul de vedere al modelării sistemelor decizionale, logica multivalentă permite abordarea naturală a mai multor situaţii decât logica bivalentă. Logica bivalentă este eficientă, de exemplu, pentru proiectarea circuitelor integrate din calculatoare, dar este ineficientă în lumea reală, deoarece necesită un nivel intermediar de codare a valorilor reale cu ajutorul valorilor binare. Într-adevăr, orice logică multivalentă poate fi codată binar, dar o astfel de tratare se poate dovedi uneori nepractică.

2.1 Clasificarea sistemelor logice multivalente

Criteriile care stau la baza clasificării sistemelor logice sunt [16][17]:

- caracteristicile domeniilor pe care sunt definite variabilele – (continue sau discrete, omogene sau heterogene);
- cardinalitatea domeniilor de valenţe - bivalente sau multivalente;
- reprezentarea valenţelor – binară sau codată binar (algoritmică);
- structura variabilelor – scalară sau vectorială;
- tipul ieşirii - singulară sau multiplă (după numărul variabilelor de ieşire).

Clasificarea după domeniile de definiţie

Sistemele logice multivalente clasificate după cardinalitatea domeniilor de valenţe ale variabilelor se împart în:

- sisteme cu logică continuă (ex. sisteme fuzzy [18]);
- sisteme cu logică multivalentă (sisteme de logică discretă)[19].

Prin „**sistem de logică continuă**” se înţelege un sistem de logică multivalentă cu un număr infinit de valenţe ($n \to \infty$, n fiind numărul de

valenţe). Variabilele utilizate într-o logică continuă se numesc variabile continue.

În cazul sistemelor digitale, cu variabile definite pe o mulţime discretă şi finită, sistemul logic se va numi - doar acolo unde este necesar să-l deosebim de logica continuă - „**sistem de logică multivalentă discretă**” (sau, simplu, „**logică multivalentă discretă**”). În restul cazurilor, specificarea de „discret” nu este necesară.

Sistemele logice pot fi, prin prisma domeniilor variabilelor utilizate omogene sau heterogene. Sistemele în care variabilele sunt definite pe domenii identice de valenţe se numesc **omogene**. În sistemele **heterogene** sunt utilizate mai multe domenii de valenţe pentru reprezentarea variabilelor. Mai mult, sistemele pot fi omogene avand in vedere numai variabilele de intrare sau numai variabilele de ieşire. Sistemele omogene pe intrare şi pe ieşire dar heterogene pe ansamblu sunt utilizate pentru descrierea transaltoarelor.

Clasificarea după cardinalitatea domeniilor de valenţe

Prin „**sistem de logică bivalentă (binară)**” se înţelege un sistem formal cu ajutorul căruia se pot exprima funcţii ale căror variabile de intrare, interne sau de ieşire au două valori (valenţe).

Prin „**sistem de logică multivalentă**” se înţelege un sistem formal cu ajutorul căruia se pot exprima funcţii ale căror variabile de intrare, interne sau de ieşire pot avea două sau mai multe valori (valenţe).

Clasificarea după modul de reprezentare al valenţelor

Valenţele domeniilor binare se reprezintă utilizând simbolurile pentru adevărt şi fals. O modalitate de reprezentare a valenţelor domeniilor multivalente o reprezintă utilizarea unor secvenţe unice de valenţe binare pentru fiecare valoare multivalentă. Secvenţele se obţin cu algoritmi de codare.

Prin „**sistem de logică algoritmică**” se înţelege un sistem de logică ce utilizează o algebră multivalentă, ale cărei valenţe sunt codate cu ajutorul a două (**binar**) sau mai multor valenţe (**n-ar**). Logica algoritmică permite transpunerea algoritmilor în funcţii logice (funcţii decizionale) în vederea optimizării algoritmilor şi/sau implementării acestora pe sisteme hardware reconfigurabile. Funcţiile logice pot fi implementate astfel încât să permită prelucrarea intrărilor într-o manieră paralelă, specifică operării cu funcţiile combinatoriale şi tehnicile de sinteză a circuitelor digitale.

Logica algoritmică deschide, în principiu, calea creşterii sensibile a vitezei de prelucrare a informaţiei în calculatoare structurate corespunzător tehnicii sugerate mai sus, dar folosind tehnologia existentă. Perspectiva folosirii unor asemenea sisteme se întrevede în tehnologiile viitorului, folosind de exemplu modularea optică a luminii, tehnologii biologice, tehnologia cristalelor, dar şi în implementări multivalente din tehnologia semiconductorilor, bazate pe nivele logice multiple de tensiune sau curent.

Clasificare sistemelor de logică				Caracteristicile domeniilor de definiţie						
Corespondenţa I/O			Notaţie	Logică discretă						Logică continuă
Intrări	Ieşire			număr valenţe						
				=2		>2				
reprezentare	reprezentare	tip		BiValentă	Algoritmi că binară	MultiValentă	Algoritmi că n-ară			
[s]/[v]	[s]/[v]	[1]/[n]		[Di]	[Bi]	[A2]	[MV]	[An]	[Co]	
scalară [s]	scalară [s]	singulară [1]	[SS1]	Di SS1	Bi SS1	-	MV SS1	-	Co SS1	
		multiplă [n]	[SSn]	Di SSn	Bi SSn	-	MV SSn	-	Co SSn	
scalară [s]	vectorială [v]	singulară [1]	[SV1]	Di SV1	Bi SV1	-	MV SV1	-	Co SV1	
		multiplă [n]	[SVn]	Di SVn	Bi SVn	-	MV SVn	-	Co SVn	
vectorială [v]	scalară [s]	singulară [1]	[VS1]	DI VS1	BI VS1	A2 VS1	MV VS1	An VS1	CO VS1	
		multiplă [n]	[VSn]	Di VSn	Bi vsn	A2 vsn	Mv vsn	An vsn	Co VSn	
vectorială [v]	Vectorială [v]	singulară [1]	[VV1]	Di VV1	Bi vv1	A2 vv1	Mv vv1	An vv1	Co VV1	
		multiplă [n]	[VVn]	DI vvn	BI vvn	A2 vvn	MV vvn	An vvn	CO VVn	

Tab. 2-1. Clasificarea sistemelor logice

Clasificarea după structura variabilelor

În funcţie de structura variabilelor – scalare sau vectoriale – sistemul logic va fi numit:

- „**sistem de logică multivalentă scalar**" sau „logică multivalentă scalară";
- „**sistem de logică multivalentă vectorial**" sau „logică multivalentă vectorială".

Clasificarea sistemelor după numărul de ieşiri

Logica multivalentă singulară este destinată specificării sistemelor cu o singură ieşire. Ea se mai numeşte şi „**logica sistemelor cu o singură ieşire**". În funcţie de caracteristicile domeniului asociat variabilei de ieşire această logică poate fi clasificată după cum urmează:

1. **Logică multivalentă singulară discretă**
   - „logică binară" - sistem logic cu două valenţe pe ieşire;
   - „logică n-ară" - sistem logic cu n valenţe pe ieşire, unde $n > 2$;
2. **Logică multivalentă singulară continuă** („**logică singulară continuă**"), este utilizată în sisteme care iau decizii logice pe un set de variabile cu un număr infinit de valenţe („variabile continue"). Un caz particular al logicii continue este „**logica simplă fuzzy**", aplicată părţii decizionale a unui sistem fuzzy cu o singură ieşire.

Logica multivalentă multiplă este destinată specificării sistemelor cu mai multe ieşiri. Ea se mai numeşte şi „**logica sistemelor cu ieşiri multiple**". În funcţie de caracteristicile domeniului de valenţe această logică poate fi clasificată după cum urmează:

1. **Logică multivalentă multiplă discretă**
   - „logică binară" - un sistem logic cu două valenţe pe fiecare ieşire;
   - „logică n-ară" - sistem logic cu n valenţe pe fiecare ieşire, unde $n > 2$;
2. **Logică multivalentă multiplă continuă** („**logică multiplă continuă**"), este utilizată în sisteme care iau simultan mai multe decizii logice pe un set de variabile de ieşire cu un număr infinit de valenţe („variabile continue"). Un caz particular al logicii continue este „**logica multiplă fuzzy**", aplicată părţii decizionale a unui sistem fuzzy cu mai multe ieşiri.

Logica multivalentă vectorială mai este numită şi „**logica sistemelor cu mai multe ieşiri**", pentru că sistemul logic formal permite o tratare globală optimizată a ieşirilor multiple ale sistemului analizat sau supus sintezei. Toate ieşirile sistemului sunt incorporate într-o variabilă de tip vector. La rândul lor, fiecare variabilă de intrare sau ieşire poate fi un vector.

2.2 Specificarea sistemelor logice

Documentul specificativ (sau **specificarea**), descrie în mod explicit funcţionalitatea sistemului – scopul exact al acestuia – şi prezintă toate constrângerile pe care trebuie să le satisfacă. Specificarea va include domeniile de intrare şi ieşire ale sistemului precum şi corespondenţa dintre acestea.

Documentul reprezintă un contract. Se va considera că implementatorii au realizat contractul în momentul în care vor oferi o soluţie care să satisfacă criteriile din specificare.

Specificarea unui sistem logic se face cu ajutorul reprezentărilor tabelare interpretate în mod determinist sau nedeterminist.

Def. 2-1. **Tabelul** este structura de date utilizată pentru reprezentarea unui sistem decizional prin specificarea tuturor corespondenţelor intrare/ieşire (rândurile tabelului). Tabelele sunt formate din **rânduri** care reprezintă o corespondenţă între o combinaţie de intrare a sistemului şi o combinaţie corespunzătoare ce trebuie să se regăsească la ieşirea sistemului.

Def. 2-2. Un tabel este **complet specificat**, dacă toate combinaţiile de intrare se regăsesc în tabelul de valori; în caz contrar este **incomplet specificat**. Un tabel complet specificat poate fi reprezentat prin câte un rând pentru fiecare combinaţie de intrare sau prin gruparea de combinaţii de intrare cu aceeaşi ieşire cu scopul de a reduce dimensiunea specificării.

Valorile ataşate variabilelor de ieşire pentru un rând pot fi singulare (o singură valoare) sau pot fi submulţimi ale domeniilor corespunzătoare fiecărei variabile de ieşire în parte.

Def. 2-3. O corespondenţă intrare/ieşire (un rând al tabelului de valori) se numeşte **univocă** dacă fiecărei ieşiri îi corespunde o singură valoare. Dacă cel puţin unei variabile de ieşire îi corespunde mai mult de o valoare atunci corespondenţa se numeşte **neunivocă**.

Def. 2-4. Un tabel se numeşte **univoc** dacă toate corespondenţele intrare/ieşire (toate rândurile) sunt univoce. În caz contrar tabelul se numeşte **neunivoc**.

Def. 2-5. Un tabel este **nedeterminist** dacă o combinaţie de intrare nu constrânge sistemul să determine la fiecare ieşire în parte toate valorile specificate. Sistemul trebuie să prezinte cel puţin o valoare la fiecare ieşire.

Def. 2-6. Un tabel este **determinist** dacă o combinaţie de intrare constrânge sistemul să determine la fiecare ieşire în parte toate valorile specificate.

Ca o consecinţă, un tabel reprezintă o corespondenţă **deterministă**, dacă corespondenţa intrare/ieşire este **univocă.** O corespondenţă neunivocă nu reprezintă neapărat o corespondenţă nedeterministă. Un tabel

este **determinist** dacă pentru orice combinaţie de intrare la ieşirea sistemului trebuie să fie prezente toate valorile de ieşire specificate, indiferent dacă este univoc sau neunivoc.

Este evident că un tabel univoc este determinist şi un tabel nu poate fi nedeterminist decât dacă este neunivoc. Tabele neunivoce reprezintă doar o subclasă a tabelelor deterministe. Fixarea unei singure valori de ieşire pentru tabele nedeterministe reprezintă o constrângere care reduce clasa de soluţii pentru minimizarea acestora. Interpretarea cea mai naturală o reprezintă selectarea unui subset optim din punct de vedere al minimizării pentru valorile de ieşire corespunzătoare rândurilor neunivoce.

Din cele de mai sus se deduce că o specificare tabelară poate avea ataşată o semantică determinstă dacă este neunivoc.

Probleme de specificare

O specificare poate fi deficitară prin ambiguitate sau incompletitudine. Ambiguitatea se manifestă când secţiuni sau propoziţii din documentul de specificare permit mai multe interpretări. În cazul sistemelor logice ambiguitatea nu este neapărat o deficienţă, ea fiind proprie sistemelor neunivoce, indiferent de semantica determinstă sau nedeterministă.

O clasă distinctă este cea a specificărilor incomplete [20], în care nu sunt menţionate combinaţiile de intrare ce nu pot să apară în timpul funcţionării sistemului. O astfel de specificare este acceptabilă numai când se cunosc cu certitudine aceste combinaţii. Nu este necesară specificarea combinaţiilor de intrare inaccesibile. Pe de altă parte, aceste combinaţii reprezintă una din sursele minimizării, aşa cum se va vedea mai târziu. Specificările incomplete apar frecvent în sistemele descrise prin intermediul tabelelor.

Simboluri speciale utilizate în specificare

Formalismul utilizat în specificarea tabelelor include utilizarea unor simboluri speciale: „~” (**ALL**) şi „ –” (**DC**). Simbolul **ALL** reprezintă toate valorile domeniului iar simbolul **DC** reprezintă orice submulţime a domeniului. Oricare din cele două simboluri pot să apară atât în partea de intrare cât şi în partea de ieşire a unui rând.

Simbolul ALL

O poziţie dintr-un rând corespunzătoare uneia dintre variabilele de intrare este marcată prin **„~”** (**ALL**), dacă ea reprezintă întreg domeniul de valori al variabilei. Valoarea **ALL** se poate specifica şi prin enumerarea întregului spectru de valorii al domeniului.

Un rând cu specificarea **ALL** pe cel puţin o poziţie a unei variabile de ieşire pune în evidenţă o corespondenţă **neunivocă**.

Simbolul DC

O poziţie a unei variabile este marcată prin **„–” (DC)**, atunci când variabila nu este specificată pentru acea poziţie din tabel. Prin urmare „nu

interesează", considerându-se că poate reprezenta orice submulţime a domeniului de valori al variabilei, inclusiv mulţimea totală sau mulţimea vidă. Mulţimea vidă poate fi utilizată numai pentru ieşire. Într-un asemenea caz, avem de-a face cu un adevărat „**don't care**".

Relativ la o valenţă a domeniului, utilizarea valorii **DC** implică faptul că valenţa poate să fie sau nu prezentă –"**don't care**"– sau, cu alte cuvinte, nu este obligatoriu ca toate valenţele domeniului să fie prezente pe ieşirea sistemului. O consecinţă directă este aceea că atunci când o funcţie este minimizată, nu are rost luarea în calcul în procesul de minimizare a rândurilor care au **DC** („**don't care**") pe ieşire, întrucât un asemenea rând nu interesează şi nu trebuie să implice procesări specifice. De asemenea, un rând care are **DC („–")** pe toate poziţiile de ieşire poate fi eliminat complet din tabel înainte de a începe procesul de minimizare.

O specificare **DC** pe o poziţie a unei variabile de intrare (**„input DC"**) poate reprezenta – ca şi în cazul unei variabile de ieşire – orice submulţime a domeniului de valori. Într-un asemenea caz, rândul nu poate fi eliminat din tabel şi, mai mult, trebuie luat în considerare cu întreg domeniul de valori. Spre deosebire de ieşire unde nu interesa, valoarea **DC** intră drept condiţie impusă. Eliminarea rândului ar compromite corespondenţa intrare/ieşire. În timpul minimizării o poziţie **DC** nu poate discrimina doi vectori. Principiul discriminării implicat în minimizare nu poate fi aplicat deoarece nici o valoare a unei variabile nu se poate diferenţia în raport cu **DC** (şi evident nici în raport cu **ALL**).

În cazul tabelelor neunivoce deterministe procesul de minimizare trebuie să urmărească generarea obligatorie a tuturor valenţelor de ieşire ce pot fi prezente conform specificaţiei iniţiale.

Diferenţa dintre ALL şi DC

O remarcă importantă referitoare la specificările **DC** şi **ALL** constă în faptul că o corespondenţă intrare – ieşire este complet definită numai de către valorile variabilelor de intrare diferite de **DC** sau **ALL**. Valorile **DC** şi **ALL** intervin în procesul minimizării, efectul lor în poziţiile de intrare şi ieşire fiind precizat anterior.

De aici rezultă că o funcţie este cu atât mai simplu reprezentată (mai apropiată de o formă minimizată), cu cât sunt mai puţine rânduri în tabel şi mai puţine poziţii specificate (literale), adică poziţii diferite de **ALL** şi **DC**.

Observaţiile de mai sus definesc rolul minimizării ca fiind reducerea numărului de literale şi de rânduri. Limita inferioară a minimizării este dată de un volum suficient de informaţie înglobat în literale, astfel încât, atunci când sunt aplicate la intrarea tabelului minimizat combinaţii care în tabelul iniţial aveau valori de ieşire diferite să se obţină rezultatele corespunzătoare, diferite, la ieşire.

Influenţa simbolurilor ALL si DC asupra minimizării

Este important de remarcat că o specificare **ALL** nu reprezintă un „**don't care**" (**DC**), deoarece ea desemnează o mulţime clar specificată: toată mulţimea de valori a variabilei de intrare sau de ieşire.

Rezumând:

- valorile **DC** şi **ALL** pe partea de intrare a unui tabel sunt tratate unitar, ca având semantica simbolului **ALL**;
- pe partea de ieşire, poziţiile specificate cu simbolul **DC** nu sunt luate în considerare – este eliminat la minimizare rândul care conţine numai **DC p**e ieşire;
- pe partea de ieşire, poziţiile specificate cu simbolul **ALL** sunt considerate ca specificaţie neunivocă – evidenţiază o corespondenţă neunivocă.

În cazul tabelelor incomplet specificate valoare implicită de ieşire pentru combinaţiile nespecificate este **DC**. Cu ajutorul opţiunii **„în rest"** se modifică valoarea de ieşire pentru aceste combinaţii.

Sintagma „**în rest**" reprezintă generic toate combinaţiile de intrare nespecificate, iar „**default**" reprezintă valoarea de ieşire atribuită acestora. Opţiunea „**în rest**" poate fi utilizată o singură dată într-o specificaţie. Tabelele incomplet specificate se transfromă în complet specificate utilizând opţiunea „**în rest**". În cazul tabelelor complet specificate utilizarea acestei opţiuni nu se justifică.

Sintaxa acestei linii de tabel este

```
în rest / default
```

sau:

```
/ default
```

De exemplu, un rând

```
/ 5
```

reprezintă o specificaţie de tip „în rest" unde 5 este dat ca valoare „**default**".

Pentru valoarea „**default**" pot fi utilizate şi submulţimi ale domeniului de valori:

```
/ {2, 3}
```

În cazul tabelelor care nu au specificaţia „**în rest**" se consideră implicit că pentru combinaţiile de intrare nespecificate este admisă orice valoare de ieşire, adică:

```
/ -
```

2.3 Minimizarea în contextul optimizării sistemelor logice

Minimizare unei specificări logice este procesul în care printr-un set de deducţii logice bazate pe afirmaţiile iniţiale se poate obţine o reprezentare echivalentă redusă din punct de vedere al numărului total de componente sau al complexităţii acestora. Minimizarea are rolul de a pregăti un sistem logic pentru a fi implementat.

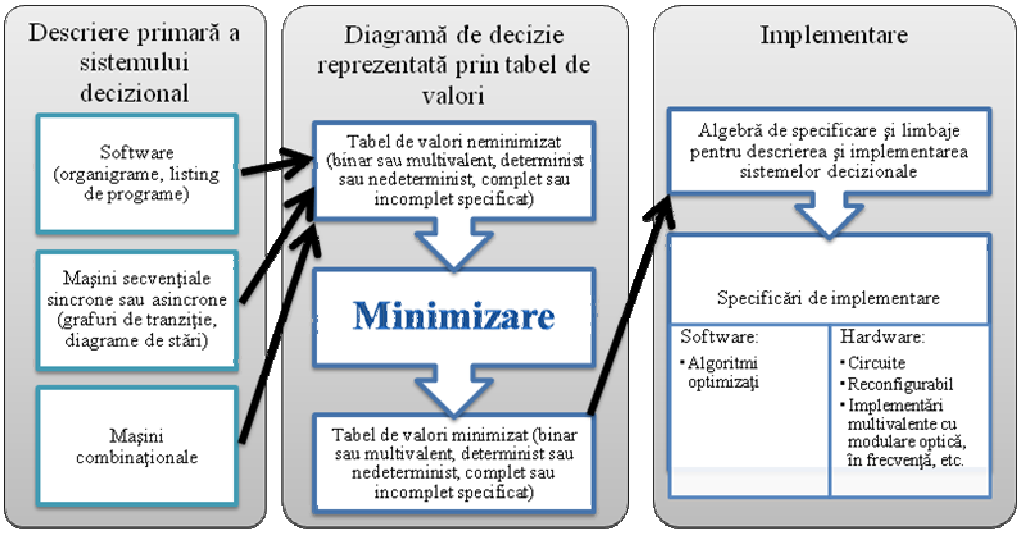


Fig. 2-1 Minimizarea în contextul optimizării sistemelor decizionale

Minimizarea este plasată ca proces între descrierea primară a unui sistem decizional şi faza de implementare a sistemului. Descrierea primară, în funcţie de provenienţa sistemului decizional, are diferite forme:

- pentru software organigrame sau cod sursă,
- pentru maşini secvenţiale sincrone sau asincrone - grafuri de tranziţie sau diagrame de stări,
- pentru maşini combinaţionale - tabele de valori.

Specificările de implementare ale sistemului optimizat se obţin cu ajutorul algebrelor de specificare sau a limbajelor pentru descrierea şi implementarea sistemelor decizionale. Implementarea propriu-zisă poate fi software sau hardware [21] - circuite, hard reconfigurabil sau implementări multivalente cu modulare optică, în frecvenţă etc.

Diagrama din Fig. 2-1 prezintă localizarea minimizării în contextul general al optimizării şi implementării sistemelor decizionale.

Minimizarea se aplică specificărilor:

- complete sau incomplete;
- cu variabile de intrare şi ieşire binare şi/sau multivalente;
- cu una sau mai multe ieşiri;
- deterministe sau nedeterministe.

Minimizarea sistemelor cu ieşiri multiple se face prin tratarea fiecărei ieşiri în mod independent (minimizare paralelă) sau prin interconectarea în reţea a variabilelor de ieşire prin utilizarea unora dintre acestea ca variabile de intrare pentru determinarea celorlalte ieşiri (minimizare în reţea). Minimizarea în reţea se poate face cu sau fără corelarea valorilor variabilelor de ieşire. De exemplu, pentru un sistem cu două ieşiri, valorile variabilelor de ieşire pot fi numai în anumite combinaţii (*(00)*, *(01)* şi *(10)*) şi nu *(11)*.

Tehnica de minimizare poate fi adaptată în funcţie de natura şi destinaţia tabelului de valori prin considerarea unor opţiuni cum ar fi:

- cu sau fără păstrarea integrităţii specificaţiei iniţiale;
- cu sau fără păstrarea corelaţiilor dintre ieşiri;
- cu metrică bazată numai pe numărul de literale;
- cu metrică bazată pe numărul de literale şi pe numărul de rânduri.

2.4 Exemplu de minimizare multivalentă

Logica multivalentă este un instrument de exprimare a corespondenţelor intrare/ieşire utilizate pentru descrierea părţilor decizionale ale algoritmilor şi sistemelor digitale. Operarea cu logica multivalentă începe să capete o importanţă majoră în domeniul minimizării software şi în noile dezvoltări ale circuitelor digitale ce utilizează codificarea multivalentă în curent, în frecvenţă sau optic.

O cale naturală de exprimare a funcţiei unui sistem decizional este un tabel binar sau multivalent. Tabelul este nucleul unui sistem decizional ce înglobează structura logică a unui algoritm sau structura logică a unui circuit digital. Optimizarea nucleului este reprezentată de minimizarea corespondenţei intrare/ieşire transpusă prin tabelul de valori.

Interesul crescând pentru minimizare este justificat de faptul că optimizarea algoritmilor în fazele incipiente (de diagramă de flux de date) conduce la implementări software ulterioare mai bune [22] [235][240][243]. Optimizarea necesită un instrument eficient şi o strategie corespunzătoare.

În programul de mai jos predomină componenta decizională faţă de componenta operaţională:

```
int function(int x1,int x2,int x3)
{
        if ( x1==1 && x2==3 && x3==1 ) return 0;
        if ( x1==2 && x2==1 && x3==4 ) return 3;
        if ( x1==3) if( x2==5 && x3==7 ) return 9;
        else if( x2==7 && x3==2 ) return 7;
        if ( x1==4 && x2==2 && x3==2 ) return 0;
        if ( x1==5)     if( x2==7)      if( x3==2 ) return 7;
        else if ( x3==3 ) return 0;
}
```

Algoritmul poate fi scris sub forma unei corespondenţe multivalente intrare/ieşire exprimată tabelar astfel:

```
x1 x2  x3  / function
1   3   1  /  0
2   1   4  /  3
3   5   7  /  9
3   7   2  /  7
4   2   2  /  0
5   7   2  /  7
5   7   3  /  0
```

Aplicând o metodă de minimizare tabelul devine:

```
x1  x2  x3 /  function
3   -   - /  9
4   -   - /  3
-   7   2 /  7
în rest   /  0
```

Prin transpunerea tabelului minimizat în algoritm, se obţine următorul rezultat:

```
int function(int x1,int x2,int x3)
{
        if( x1==3 ) return  9;
        if( x1==4 ) return 3;
        if( x2==7 && x3==2 ) return 7;
        return 0;
}
```

Se observă că utilizarea unei metode de minimizare conduce la o micşorarea a numărului de linii din aplicaţia software (de la 7 linii la 4 linii).

2.5 Concluzii

În acest capitol am realizat o clasificare a sistemelor decizionale (Fig. 2-1) în funcţie de caracteristicile domeniilor pe care sunt definite variabilele, cardinalitatea domeniilor de valenţe, modul de reprezentare a valenţelor, structura variabilelor şi tipul ieşirii.

În cadrul subcapitolului dedicat specificării sistemelor decizionale am introdus conceptele:

- specificare univocă;
- specificare neunivocă cu semantică deterministă;
- specificare neunivocă cu semantică nedeterministă.

Utlizarea reprezentărilor tabelare clasică presupunea sistemele neunivoce ca fiind implicit nedetreministe. Semantica deterministă se putea obţine numai prin modificarea corespunzătoare a structurii variabilelor de ieşire. Algoritmii clasici nu permit minimizarea tabelelor neunivoce cu semantică deterministă.

Am introdus un simbol nou (**ALL**) în specificarea tabelară a sistemelor decizionale care, spre deosebire de **DC**, impune generarea pe ieşire a tuturor valenţelor domeniului.

3 Metode clasice, algoritmi şi metode auxiliare utilizate în minimizarea funcţiilor logice

În evoluţia metodelor şi algoritmilor de minimizare se disting două etape. În prima etapă, până în 1986, s-au evidenţiat metoda grafo-analitică dezvoltată de Maurice Karnaugh [4] şi cea analitică dezvoltată de Willard Van Orman Quine şi Edward J. McCluskey [5][6]. În 1986 R. E. Bryton, pornind de la studiul diagramelor orientate aciclice, a găsit o nouă manieră de operare asupra funcţiilor boolene cu ajutorul diagramelor de decizie [23][24]. În perioada următoare, eforturile depuse în domeniul minimizării s-au concentrat în cea mai mare parte pe îmbunătăţirea metodelor bazate pe diagrame de decizie sau pe analiza proprietăţilor diagramelor de decizie utilizate pentru reprezentarea sistemelor logice [25].

Reprezentarea cu ajutorul diagramelor orientate aciclice a facilitat evoluţia rapidă a domeniului având ca implicaţie directă extinderea domeniului de utilizare al sistemelor logice complexe. Teoria dezvoltată pe baza diagramelor de decizie nu este acoperitoare, deoarece domeniul de aplicare este limitat de complexitatea calculului şi de forma de reprezentare. În acest sens au fost dezvoltate metode anexe care să trateze sisteme logice binare complexe reprezentate cu ajutorul diagramelor de decizie sau au fost create noi metode bazate pe reprezentări tabelare ale sistemelor.

Date fiind complexitatea domeniului [26] şi numărul mare de abordări posibile, studiile din domeniu au condus la dezvoltarea unor metode specifice de minimizare [27][28][29] ce nu pot fi clasificate în mod absolut. Fiecare dintre acestea acoperă o anumită subclasă din problematica sistemelor logice. O comparaţie viabilă între metode rezultă din analiza tipurilor de sisteme logice admise şi a facilităţilor de prelucrare oferite.

În continuare sunt prezentate metode şi algoritmi dezvoltaţi pentru minimizare şi câteva tehnici auxiliare utilizate în cadrul aplicaţiilor de minimizare.

3.1 Structuri de date utilizate în minimizare

Un algoritm pentru procesarea funcţiilor logice trebuie să fie susţinut de o structură de date care, pe de o parte să permită memorarea cu un cost minim a funcţiei şi, pe de altă parte, să faciliteze operaţiile uzuale cu

funcţiile logice: evaluare, comparare, compunere etc. [30][31][32][33][34] S-au făcut numeroase cercetări în vederea modelării structurilor de date şi a operaţiilor asupra acestora legate atât de proiectarea logică cât şi de domeniul inteligenţei artificiale [35][36][37][3][38][39][40][41][42].

În cadrul aplicaţiilor de minimizare sunt utilizate două tipuri de structuri de date: structura tabelară (care include şi formatul mai evoluat - notaţia poziţională) şi diagramele de decizie [43][44][45][46][47][48].

3.1.1 Notaţia poziţională

Utilizarea notaţiei poziţionale în specificarea sistemelor logice implică utilizarea unei codări binare pentru reprezentarea valenţelor unei variabile. Fiecare vector reprezentat printr-o combinaţie de n valori booleene este proiectat într-un spaţiu n dimensional, fiecare dimensiune fiind atribuită uneia dintre variabile. Un punct din acest spaţiu reprezintă un mintermen. Mai mulţi mintermeni alăturaţi formează un cub. Conform definiţiei lui J. P. Roth [49] un cub este definit ca un vector de n elemente formate din 0, 1 sau X. 1 înseamnă valoarea adevărat, 0 înseamnă complementul valorii 1 şi X reprezintă 0 sau 1 sau ambele valori simultan.

Un mecanism de reprezentare mai performant se poate obţine prin utilizarea cuburilor codate binar. Fiecare valoare este reprezentată printr-o pereche de doi biţi, primul pentru valoarea 0 şi al doilea pentru valoarea 1. 10 este utilizat pentru reprezentarea valorii 0, 01 pentru reprezentarea valorii 1 şi 11 pentru reprezentarea lui X. Semantica acestei reprezentări este următoarea: fiecare bit indică existenţa sau inexistenţa valorii corespunzătoare lui. Operarea asupra vectorilor exprimaţi în notaţie poziţională poate conduce la rezultate care conţin perechea 00. Indiferent pe ce poziţie din vector se află perechea 00 ea indică un cub nul (care nu acoperă nici un mintermen). De exemplu, un vector reprezentat de patru variabile booleene A, B, C şi D se va reprezenta prin patru perechi de cate 2 biţi. Vectorul $AC\overline{D}$ se reprezintă prin cubul 1X10 care se codifică în 01 11 01 10.

Până la utilizarea structurii mai elaborate a diagramelor de decizie, notaţia poziţională era cea mai avansată formă de reprezentare a funcţiilor logice. Prima aplicaţie care defineşte un aparat matematic cuprinzător pentru diagramele de decizie este MINI [50].

În cadrul acestei metode, peste notaţia poziţională au fost definiţi operatorii logici de bază (AND, OR, NOT), proprietăţile cuburilor şi relaţiile dintre ei.

Mărimea unui cub - numărul de mintermeni acoperiţi - este dată de produsul numărului de 1 existenţi în reprezentarea fiecărei valori.

Rezultatul operaţiei AND se obţine prin aplicarea operaţiei AND bit cu bit între cuburile codate ataşate operanzilor.

Rezultatul operaţiei OR este o listă formată din cuburile codate. În anumite condiţii aceste cuburi se pot unifica într-unul singur.

Rezultatul operaţiei NOT este o listă de cuburi obţinută prin negarea pe rând a fiecărei poziţii.

Distanţa dintre două cuburi este reprezentată de numărul de valori din intersecţia cuburilor care nu conţin nici un 1.

Un cub C1 este acoperit de un cub C2 dacă rezultatul operaţiei C1 AND NOT C2 nu conţine numai 0.

Două cuburi pot fi reprezentate printr-unul singur dacă diferă printr-o singură poziţie.

Plecând de la definiţiile de mai sus se pot descrie procesările necesare în aplicaţiile de minimizare care utilizează specificaţia tabelară.

3.1.2 Diagramele de decizie binare (BDD)

O altă structură utilizată pentru reprezentarea sistemelor este diagrama de decizie binară (BDD). BDD se bazează pe descompunerea funcţiilor logice [51][52][53][54][55]. Mai precis, o expresie logică complexă poate fi mai uşor stocată dacă este folosită o formă de reprezentare ce implică factorizarea termenilor. Funcţiile complexe apar ca rezultat al produsului cartezian dintre mai multe subfuncţii. Cheia utilizării diagramelor de decizie este dată de factorizarea expresiilor logice. Factorii produsului cartezian pot fi determinaţi pentru o expresie prin descompunerea acesteia în subfuncţii. Descompunerea utilizează operatorul definit de Claude Shannon [56][57] (cofactori).

Conceptul de BDD a fost introdus de C.Y. Lee [51] şi S.B. Akers [52]. Randal E. Bryant a propus un pachet de funcţii pentru BDD [23][24] care rezolvă în mod eficient operarea cu ajutorul acestui tip de structură de date. Diagramele sunt aplicate în diverse domenii cum ar fi:

- alocarea resurselor [58][59],
- compararea funcţiilor booleene [60][61],
- demonstrarea automată a teoremelor [62][63][64][65],
- descompunerea funcţiilor booleene [66][67],
- implementarea tehnicilor implicite de manipulare a funcţiilor booleene [24][68][69][70][71],
- înmulţirea întregilor [72],
- inteligenţa artificială,
- manipularea automatelor finite [73][74][75],
- manipularea polinoamelor [76],
- optimizarea logică a circuitelor combinaţionale [77][78][79][80][81][82][83][84][85][86],
- programarea 0 – 1 [87],
- simularea simbolică [88][89][90],
- sinteza multinivel [91][92][93][94][95][96][97][98][99],
- sinteza şi alocarea resurselor pentru circuitele de tip FPGA [100][101][102][103][104],

- testarea circuitelor numerice şi generarea vectorilor de test [105][106][107][108][109][110][111],
- testarea echivalenţei automatelor finite [112][113][114][115],
- testarea echivalenţei funcţiilor booleene [116][117][118][119],
- verificarea circuitelor şi structurilor numerice combinaţionale [120][121][122][123][124][125].

Descrierea diagramelor de decizie

Diagramele de decizie reprezintă structura de date cea mai utilizată pentru reprezentarea şi manipularea funcţiilor binare. Cea mai importantă proprietate a diagramelor de decizie este aceea că permit manipularea eficientă a sistemelor deoarece sunt canonice [24][126] [127][128][129][130][131].

Diagramele de decizie sunt stocate în grafuri orientate aciclice.

Într-o diagramă de decizie pot fi înglobate simultan mai multe funcţii având acelaşi domeniu de definiţie [132]. Fiecare funcţie este un pointer către un nod din graf.

Nodurile terminale sunt etichetate cu valori din domeniile de ieşire ale funcţiilor reprezentate.

Toate nodurile neterminale se numesc noduri interne şi sunt etichetate cu o variabilă din domeniul de definiţie, iar arcele care pornesc din aceste noduri sunt au asociate una sau mai multe valori din domeniului variabilei nodului.

Pe orice cale din diagramele de decizie, de la rădăcină până la o frunză, fiecare variabilă apare o singură dată.

Diagramele de decizie ordonate au proprietatea că există o aranjare a variabilelor de intrare astfel încât pe orice cale de la rădăcină la o frunză variabilele apar corespunzător acestei ordonări. Aceasta înseamnă că dacă o variabilă x_i apare în ordinea stabilită înaintea variabilei x_j, atunci în orice cale din diagrama de decizie niciodată x_j nu apare înaintea lui x_i.

Reprezentarea funcţiilor cu ajutorul diagramelor de decizie ordonate este canonică. Două funcţii sunt identice dacă reprezentările lor cu ajutorul diagramelor de decizie – utilizând aceeaşi aranjare pentru variabile – sunt identice. Reducerea diagramelor de decizie se bazează doar pe două reguli:

- regula de ştergere
- regula de unificare

Scopul reducerilor este de a micşora numărul de noduri din diagramele de decizie prin eliminarea nodurilor echivalente. Două noduri sunt echivalente dacă numărul de arce care pornesc din acestea sunt egale, corespund aceloraşi perechi (variabilă/valoare) şi conduc către aceleaşi noduri. Eliminarea se face prin alegerea unuia dintre cele două noduri şi mutarea tuturor arcelor care ajungeau în celălalt spre acesta. Nodul în care nu mai intră nici un arc este eliminat împreună cu toate arcele care pornesc din el. Conform teoriei automatelor acesta a devenit nod inaccesibil. Operaţia de eliminarea se face de jos în sus (bottom-up). Ea continuă până când nu mai există noduri echivalente.

Mărimea diagramelor de decizie rezultate depinde de ordinea în care sunt testate variabilele [133][134][135]. Alegerea ordinii variabilelor determină un număr liniar sau exponenţial de noduri. Un exemplu clasic în acest sens este reprezentarea funcţiei $f = x_1x_2 + x_3x_4 + x_5x_6$, considerând următoarele ordonări: $(x_1, x_2, x_3, x_4, x_5, x_6)$ şi $(x_1, x_3, x_5, x_2, x_4, x_6)$.

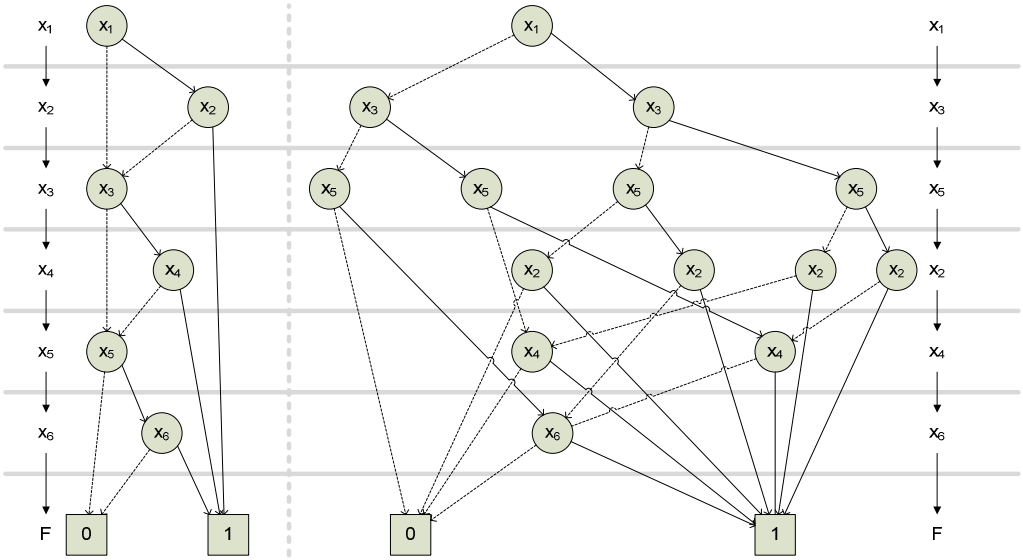


Fig. 3-1 Influenţa ordinei variabilelor asupra mărimii diagramelor de decizie

Din acest exemplu reiese clar că ordinea variabilelor este foarte importantă. În primul caz diagrama are 6 noduri interne, iar în al doilea 14.

Algoritmul lui Bryant [136][137] aplică regulile prezentate pentru reducerea diagramelor de decizie pe parcurgerea de tip bottom-up. Complexitatea algoritmului este $O(|G| log(|G|))$, unde $|G|$ reprezintă numărul de noduri ale diagramei de decizie.

În tabelul de mai jos sunt enumerate o serie de pachete de programe dezvoltate pentru implementarea structurii BDD şi a algoritmilor de manipulare pentru diagramele de decizie.

Pachet/Autor	**Descriere**
BDD Universitatea Berkeley	Bibliotecă de funcţii care utilizează tehnica parcurgerii în lăţime pentru gestiunea diagramelor de decizie binare.

ABCD Armin Biere [136]	Pachetul ABCD a fost dezvoltat şi utilizat cu succes pentru evaluarea diagramelor de decizie utilizate la verificarea modelelor de către Bwolen Yang. ABCD este o librărie bazată pe parcurgerea în adâncime. Funcţiile notabile sunt: marchează şi şterge (mark-and-sweep) bazată pe garbage collection şi integrarea nodurilor BDD cu tabele unice. BDD foloseşte adresarea deschisă, referinţe indexate (în loc de pointeri) către nodurile BDD. Această tehnică reduce mărimea unui nod BDD la jumătate (2 cuvinte maşină în loc de 4).
CAL Rajeev Ranjan şi Jagesh Sanghavi [138][139][140]	CAL este o librărie publică bazată pe parcurgerea în lăţime pentru a beneficia de localizarea în memorie. Algoritmul pentru garbage collection este bazat pe gestiunea referinţelor cu compactare de memorie. Pentru a creşte localizarea referinţelor fiecare nod BDD conţine referinţele cofactorilor. Pentru a păstra mărimea de 4 cuvinte maşină este folosită gruparea biţilor pentru a folosi şi salva referinţele unui nod. Operaţia „relational product" se bazează pe parcurgerea în adâncime cu o cuantificare a fiecărui pas bazată pe parcurgerea în lăţime.
CUDD Fabio Somenzi [8], [24]	Biblioteca de funcţii pentru manipularea diagramelor de decizie binare, algebrice şi reduse (prin eliminarea ramurilor care conduc numai la valoarea 0 - ZBDD). CUDD este o librărie BDD publică bazată pe parcurgere în adâncime. În CUDD referinţele nodurilor sunt memorate în timp real pe parcursul calculelor. Pentru a înlătura efectele actualizărilor asupra performanţei (când un număr mare de noduri este modificat), CUDD le adaugă într-o listă şi le actualizează doar când sunt scoase, cu condiţia ca modificarea să mai fie necesară. Adăugarea în tabela CUDD utilizează o politică de premiere. Memoria tampon - „cache" - creşte dacă numărul de potriviri creşte. CUDD sortează parţial lista rămasă în procesul de garbage collection pentru a îmbunătăţi localizarea în memorie. O altă funcţionalitate distinctă a CUDD este utilizarea unei serii de tehnici euristice pentru rearanjarea dinamică a variabilelor.

EHV Geert Janssen	EHV este o librărie BDD publică bazată pe parcurgere în adâncime. Principala diferenţă faţă de implementările uzuale este că asigură suport pentru interschimbarea intrărilor şi permite ataşarea de atribute nodurilor BDD. Cele mai recente versiuni permit ca rezultate intermediare să fie memorate în nodurile BDD, care la rândul lor, elimină necesitatea existenţei unei memorii tampon pentru anumite operaţii specifice BDD. Această facilitate încarcă memoria cu 2 cuvinte maşină pe nod.
PBF Bwolen Yang şi Ying-An Chen	PBF este o librărie experimentală bazată pe parcurgere parţială în lăţime. Parcurgerea parţială în lăţime împreună cu managerul de memorie pe variabilă şi cu mecanismul „garbage collector" care compactează memoria utilizând o tehnică de tip „marchează şi şterge" sunt folosite pentru a îmbunătăţi localizarea în memorie. Parcurgerea parţială limitează expansiunea în lăţime şi evită supraîncărcarea memoriei, specifică acestui tip de parcurgere.
TiGeR Olivier Coudert, Jean C. Madre şi Herve Touati	TiGeR este o librărie comercială bazată pe parcurgerea în adâncime. Facilităţi interesante sunt segmentarea cache-ului şi algoritmul garbage collection. În TiGeR fiecare tip de operaţie are propriul cache. Acest lucru permite optimizarea independentă a cache-urilor. Cache-ul pentru operaţii nepolinomiale cum ar fi „relational product" este configurat să fie de patru ori mai mic decât cache-ul unei operaţi polinomiale. Algoritmul TiGeR pentru garbage collection este modificat astfel: lista rămasă este sortată pentru a menţine localizarea în memorie şi compactarea memoriei este făcută numai atunci când resursele de memorie devin critice.
BuDDy Jørn Lind-Nielsen [141]	O librărie pentru BDD care implementează operaţii vectoriale, reordonarea dinamică a variabilelor, un mecanism „garbage collector" eficient, managementul automat al referinţelor. Această bibliotecă a fost dezvoltată pentru a demonstra că BDD pot fi utilizate pentru dezvoltarea unui sistem de analiză adaptabil programelor mari, uşor de implementat şi care se comportă satisfăcător pentru o cantitate mare de date de intrare a căror procesare necesită un volum mare de operaţii.

Aplicaţii de minimizare bazate pe diagrame de decizie

Una dintre şcolile recunoscute în domeniul minimizării care utilizează diagramele de decizie este Universitatea din Berkeley [142][143][144][145][146][147][148]. Acolo au fost dezvoltate aplicaţii de minimizare: ESPRESSO[149][150], SIS[9], MVSIS[151], şi BALM [148] (ultima dintre acestea abordează domeniul minimizării automatelor). Atât cât este cunoscut până în acest moment, principala referinţă în domeniul minimizării sistemelor decizionale reprezentate prin tabele multivalente este aplicaţia MVSIS_3.0.

MVSIS este versiunea pentru domeniul multivalent a programelor de minimizare implementate iniţial la Berkeley (ESPRESSO şi SIS). Principiile şi tehnicile de minimizare implementate în MVSIS nu reprezintă un optim [147] mai ales pentru minimizarea în reţea [152] şi pentru cazurile de corespondenţă intrare/ieşire pur multivalente cu ieşire singulară.

Concluzii

Datorită avantajelor pe care le prezintă utilizarea diagramelor de decizie binară în manipularea funcţiilor booleene, aceste reprezentări au fost implementate în diverse medii de proiectare asistată de calculator, fiind utilizate în diverse etape ale proiectării. Dintre acestea pot fi amintite:

- pachetele de programe MVSIS, SIS, HSIS şi VIS de la Universitatea Berkeley din California,
- pachetul de programe Alliance dezvoltat în cadrul laboratorului MASI al Universităţii Pierre et Marie Curie din Paris,
- pachetul ASYL dezvoltat în cadrul laboratorului CSI al INP Grenoble.

Avantajul major al diagramelor de decizie este acela că sunt canonice: pentru o funcţie pentru care a fost selectată o ordine de considerare a variabilelor diagrama de decizie este unică. Proprietatea de canonicitate vine în sprijinul operaţiilor de comparare dintre funcţii reprezentate cu diagrame de decizie.

Diagramele de decizie au un mare dezavantaj legat de dimensiunea acestora, fiind dramatic influenţată de ordinea în care sunt considerate variabilele [153].

3.2 Diagrama KARNAUGH

Prima metodă de minimizare a expresiilor booleene a fost propusă de Maurice Karnaugh în 1950 [4].

Metoda se bazează pe reprezentarea funcţiei cu ajutorul unor diagrame bidimensionale având drept coordonate pentru rânduri şi coloane secvenţe de valori booleene obţinute utilizând codul Gray. Codul Gray este o metodă de enumerare a tuturor combinaţiilor de n cifre binare, descoperită de Frank Gray în 1947, care are proprietatea că două coduri

succesive diferă printr-o singură cifră binară. Perechile de coduri succesive sunt numite şi coduri adiacente.

Diagramele Karnaugh sunt simplu şi facil de utilizat numai până la 6 variabile de intrare. Pentru un număr de intrări mai mare de 6 nu mai pot fi observate uşor soluţiile cu ajutorul acestei metode.

Metoda de simplificare folosind diagrama Karnaugh, aplicabilă numai funcţiilor binare cu o singură ieşire are două variante:

1. metoda grafică propriu-zisă, caracterizată prin faptul că numărul de variabile ale funcţiei care se simplifică este egal cu numărul de variabile care definesc diagrama folosită;
2. metoda grafo-analitică, în cazul căreia numărul de variabile al funcţiei este mai mare decât numărul de variabile care definesc diagrama.

Metoda Karnaugh grafică

Funcţiile binare sunt reprezentate prin specificarea corespondenţelor intrare/ieşire. Specificarea poate fi făcută:

a. **tabelar** - printr-un „**tabel de valori**" („**tabel de adevăr**", în cazul binar). Tabelul de valori este o reprezentare liniară de tipul „valoare intrare – valoare ieşire" (valoarea funcţiei).
b. **matricial** - corespondenţa intrare – ieşire este reprezentată bidimensional, în trei sau mai multe dimensiuni, printr-un cub multidimensional numit **hipercub**. Coordonatele elementului matricei sau hipercubului sunt date de combinaţii de valori ale variabilelor de intrare. Variabilele de intrare sunt partiţionate în două (cazul matricial), în trei sau mai multe părţi, combinaţiile de valori ale unui grup reprezentând una dintre coordonatele elementului matricei sau hipercubului, iar valoarea înscrisă în acest element reprezintă valoarea funcţiei.

Reprezentările matriceale sunt o împachetare în mai multe dimensiuni a tabelului (liniar) de valori al funcţiei.

Diagrama Karnaugh este reprezentarea matriceală, bidimensională a tabelului de adevăr, având specifică ordonarea combinaţiilor de valori ale intrărilor – atât pe orizontală, cât şi pe verticală – astfel încât poziţiile alăturate să difere doar prin valoarea unei singure variabile (cu alte cuvinte să fie „adiacente"). Coordonatele poziţiilor alăturate se constituie în vectori adiacenţi. Într-o diagramă Karnaugh combinaţiile capetelor unei linii sau coloane sunt adiacente.

Comasarea zonelor adiacente cu valori comune de ieşire se face urmărind coordonatele verticale şi orizontale comune (o grupare de combinaţii comună rândurilor şi una comună coloanelor) care definesc zona. Rezultatul este un hipercub reprezentat printr-un set de coordonate din care lipsesc variabilele ale căror valori diferă. Într-o specificare tabelară evidenţierea adiacenţei rândurilor se poate face analitic prin compunerea mai multor termeni într-unul singur eliminând variabilele care diferenţiază, prin valoarea lor, rândurile comasate (**simplificarea prin adiacenţă**).

Dat fiind că adiacenţa este produsă de variabile binare, rezultă că de-a lungul fiecărei dimensiuni a hipercubului numărul de zone din cadrul adiacenţei este o putere a lui 2.

Dacă grupările adiacente considerate în simplificare au valori „1" şi „**–**", atunci funcţia se scrie sub prima formă canonică. Dacă grupările considerate în simplificare au valori „0" sau „-", atunci funcţia se scrie sub a doua formă canonică.

Considerând cazul primei forme canonice, într-o diagramă Karnaugh fiecare grupare adiacentă de „1" şi „**–**" constituie un factor în produsul dual al funcţiei. Expresia funcţiei în prima formă canonică acoperă toate valorile „1" din matrice.

Dintr-o diagramă pot rezulta, în funcţie zonele selectate, mai multe expresii echivalente ale funcţiei, cu complexităţi diferite sau echivalente.

Similar se procedează şi în cazul celei de a doua forme canonice, în acest caz selectându-se în grupări adiacente valorile „0" şi, unde este necesar, valori nespecificate („**–**"). Valorile nespecificate ale funcţiei („–") se iau în considerare doar pentru completarea unei zone adiacente. Nu este necesară determinarea adiacenţelor pentru a cuprinde şi poziţiile nespecificate.

Zonele adiacente se pot întrepătrunde deoarece atunci când se repetă mintermeni în produs nu se modifică valoarea funcţiei.

Cu cât zonele comasate au o suprafaţă mai mare, cu atât expresia termenului rezultat este mai simplă deoarece prin adiacenţă se elimină mai multe variabile.

Din motive de compactare a reprezentării, se urmăreşte ca diagrama să fie cât mai pătrată. Cu alte cuvinte, cele n variabile ale funcţiei se împart în două grupe de $\left\lfloor \frac{n}{2} \right\rfloor$ şi $\left(n - \left\lfloor \frac{n}{2} \right\rfloor \right)$ variabile.

Se pot imagina reprezentări tip diagramă Karnaugh cu mai mult de patru variabile. O diagramă cu cinci variabile poate fi alcătuită din două diagrame de patru variabile, iar una de şase variabile poate fi alcătuită din două diagrame de cinci variabile ş.a.m.d.), dar cel mai comod (şi frecvent) mod de operare se obţine când aceste diagrame au cel mult patru variabile.

Reprezentarea unor funcţii cu mai mult de patru variabile se face cu diagrame Karnaugh care au cel mult patru variabile, dar aceste diagrame au valori de ieşire constituite din funcţii descrise pe baza variabilelor care nu intră în componenţa coordonatelor diagramei.

Metoda Karnaugh grafo-analitică

Această metodă este utilizată pentru simplificare folosind o diagramă Karnaugh cu mai puţine variabile decât funcţia

Dacă funcţia F are mai multe variabile decât diagrama Karnaugh prin care este reprezentată - funcţia are n variabile**,** iar diagrama este constituită din m variabile, unde $n = m + k, k > 0$**, -** atunci fiecare careu al diagramei va fi o funcţie f_p, specifică acestui careu, depinzând de restul de k variabile care nu intervin în coordonatele diagramei Karnaugh. Funcţia

rezultă din tabelul general de valori al funcţiei F. Pentru fiecare poziţie din diagramă este extrasă funcţia în cele k variabile rămase, considerând primele m variabile fixate.

Dacă, de exemplu, $x_1, x_2, \ldots, x_m$ sunt variabilele care definesc diagrama Karnaugh, iar $x_1, x_2, \ldots, x_n$ sunt variabilele funcţiei F şi dacă se notează prin $p \in \{0, 1, \ldots, 2^m - 1\}$ combinaţiile de valori ale celor m variabile care definesc diagrama, atunci funcţiile specifice f_p depind de variabilele $x_{m+1}, x_{m+2}, \ldots, x_n$.

Funcţiile f_p pot avea următoarele valori:

1. 0, 1(constante);
2. „ – ” (funcţia F este nespecificată);
3. $f_p(x_{m+1}, x_{m+2}, \ldots, x_n)$.

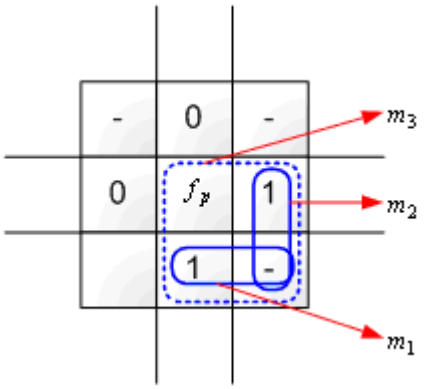


Fig. 3-2 Subdiagramă Karnaugh

Calculul fiecărei funcţii f_p se face folosind câte o subdiagramă pentru fiecare combinaţie p a variabilelor $x_1, x_2, \ldots, x_m$. Subdiagrama are coordonatele definite pe variabilele $x_{m+1}, x_{m+2}, \ldots, x_n$. Valorile din diagramă sunt extrase din funcţia iniţială prin filtrare după valorile fixate pentru variabilele $x_1, x_2, \ldots, x_n$.

Fig. 3-2 arată cum pot fi identificaţi factorii pentru prima formă canonică, rezultând următoarea expresie parţială a funcţiei: $F = \begin{vmatrix} \ldots \\ m_1 \\ m_2 \\ m_3 f_p \\ \ldots \end{vmatrix}$.

Regulile de minimizare ale funcţiei F, utilizând acest tip de diagrame sunt:

1. se minimizează funcţia, prin adiacenţele din diagramă, considerând toate valorile 1, ţinând cont şi de poziţiile „–” (în cazul primei forme canonice), sau toate valorile 0, ţinând cont şi

de „–" (în cazul celei de a doua forme canonice), fără a se considera careurile cu valorile f_p ;

2. se cuprind separat şi valorile f_p, în arii de adiacenţă care pot include şi valori 1 şi „–" în cazul primei forme canonice, sau valori 0 şi „–", în cazul celei de a doua forme canonice.

Raţiunea regulilor de mai sus este dată de faptul că f_p poate lua atât valoarea 0, cât şi 1, în funcţie de propriile variabile. În cadrul primei reguli nu pot fi luate în considerare aceste careuri (ariile trebuie sa aibă valori constante - fie 0, fie 1). În cea de a doua regulă, minimizarea cu f_p poate cuprinde zone cu 1 şi „–" (pentru prima formă canonică, când se consideră $f_p = 1$), sau zone cu 0 şi „–" (pentru a doua formă canonică când se consideră $f_p = 0$).

Concluzii

Trebuie remarcat că metoda grafo-analitică are sens să fie aplicată pentru funcţii cu mai mult de patru variabile. Până la patru variabile, inclusiv, este mai eficient să se folosească direct o diagramă având acelaşi număr de variabile ca şi funcţia. Pe de altă parte, Metoda Karnaugh grafo-analitică nu garantează întotdeauna obţinerea optimului în simplificare, pentru că simplificările se fac în două etape, fiecare dintre ele putând produce mai multe variante. Întrucât prima etapă nu ţine cont de rezultatul global, este posibil ca o formă mai simplă a unei funcţii f_p să producă o versiune mai complexă a funcţiei F decât dacă s-ar folosi o formă mai complexă a lui f_p .

Pentru funcţii cu multe variabile (chiar de la cinci variabile, în sus), se recomandă metodele analitice: Quine-McCluskey, pentru cazul binar şi Espresso, MVSIS sau metoda discriminării, pentru cazul binar sau multivalent.

În practica curentă utilizarea diagramelor Karnaugh reprezintă cea mai simplă cale de exemplificare a funcţiilor logice.

3.3 Metoda Quine-McCluskey

Prima metodă analitică de minimizare a fost propusă de Willard van Orman Quine şi Edwars J. McCluskey [5][6][154]. Algoritmul permite simplificarea funcţiilor binare cu o singură ieşire şi poate fi utilizat indiferent de numărul de variabile de intrare.

Specific acestei metode este stabilirea mulţimii vectorilor implicanţi maximali utilizând proprietatea de adiacenţă.

Paşi metodei sunt:

1. Stabilirea vectorilor implicanţi ai funcţiei $F(x_1, x_2, ..., x_N)$, pentru care $F = 1$, sau $F = -$, într-un tabel împărţit în clase $S_0^0, S_1^0, ..., S_{N-1}^0$, după numărul de valori de 1 din vectorul implicant, astfel încât clasa S_i^0 are i poziţii cu valoarea 1.
2. Clasele care diferă printr-un număr de 1 mai mare sau egal cu 2, nu pot fi combinate prin adiacenţă. Astfel,
   a. Pentru $i = 0, 1, 2, ..., N-1$ se compară fiecare element din S_i^0 cu fiecare element din S_{i+1}^0;
   b. acolo unde diferenţa apare printr-o singură variabilă x_j (vectori adiacenţi), se creează un nou vector implicant care conţine „–" pe poziţia variabilei x_j şi acest implicant se adaugă unei clase notate S_i^1;
   c. se repetă operaţia precedentă pentru toate perechile de clase S_i^0 şi S_{i+1}^0. Se generează astfel clasele $S_0^1, S_1^1, ..., S_{N-1}^1$.
3. Se reia operaţia de la punctul 2 pentru clasele $S_0^1, S_1^1, ..., S_{N-1}^1$, rezultând clasele: $S_0^2, S_1^2, ..., S_{N-1}^2$. Revenirea la punctul 2 se face până când nu mai este posibilă nici o combinaţie prin adiacenţă.
4. Tabelul de valori rezultat - mulţimea vectorilor implicanţi - se ordonează descrescător după numărul de vectori acoperiţi şi se aleg în paşi succesivi implicanţi astfel încât, de fiecare dată, să fie acoperit un număr cât mai mare de rânduri încă neacoperite din tabelul iniţial. Cu alte cuvinte, selectarea unei soluţii şi implementarea sistemului decizional mai necesită următorii paşi:
   a. simplificarea prin acoperire a tabelului care conţine vectorii implicanţi simplificaţi;
   b. sortarea soluţiilor funcţiei, după prima sau a doua formă canonică:
   c. scrierea expresiei funcţiei, corespunzătoare variantei alese;
   d. trasarea schemei logice a funcţiei.

Metoda Quine-McCluskey se aplică uzual funcţiilor reprezentate sub prima formă canonică, dar se poate ajusta astfel încât să se aplice şi funcţiilor reprezentate sub a doua formă canonică. Dacă se doreşte simplificarea funcţiei sub a doua formă canonică, atunci se aplică punctele 1–4 din algoritmul prezentat mai sus, pentru rândurile pentru care funcţia are valoarea 0 şi, dacă este necesar, folosindu-se şi rândurile în care funcţia este nespecificată („–").

Concluzii

Metoda Quine-McCluskey conţine principalele etape parcurse de orice viitoare metodă analitică de minimizare:

- determinarea vectorilor implicanţi maximali;

- selectarea unei submulţimi optime de vectori implicanţi care să acopere funcţia iniţială.

Utilizarea acestei metode pentru optimizarea globală a funcţiilor binare cu ieşiri multiple impune corelarea tuturor rezultatelor obţinute pentru fiecare funcţie de ieşire în parte utilizând alte metode. Folosirea ei poate conduce doar la optimizări separate ale funcţiei binare cu ieşiri multiple, considerând separat fiecare ieşire.

Metoda Quine-McCluskey are dezavantajul că necesită luarea în considerare şi a combinaţiilor de intrare pentru care ieşirea funcţiei *F* este nespecificată.

3.4 Metoda MINI

Metoda MINI [50][155] a fost prezentată prima dată în 1974 de către S.J. Hong, R.G. Cain şi D.L. Ostapko.

Metoda diferă de abordarea cu ajutorul diagramelor Karnaugh prin două aspecte.

c. costul funcţiei este simplificat prin atribuirea de „greutăţi" egale tuturor implicanţilor.
d. soluţia finală se obţine din forma iniţială sau din soluţia precedentă prin îmbunătăţirea iterativă mai mult decât prin generarea şi acoperirea implicanţilor primi.

Limitarea funcţiei de cost la numărul implicanţilor din soluţie are avantajul eliminării unor probleme asociate cu minimul local. Atâta timp cât numai numărul de implicanţi contează structura acestora poate fi alterată cu condiţia ca acoperirea mintermeniilor să rămână conformă cu specificaţia iniţială a funcţiei. Metodele utilizate pentru modificarea implicanţilor sunt similare cu cele folosite în minimizarea unei funcţii utilizând diagrama Karnaugh.

Algoritmul MINI porneşte de la o soluţie iniţială obţinută printr-o operaţie de preprocesare. Aceasta are ca scop descompunerea vectorilor în vectori disjuncţi pentru:

e. evitarea dependenţei de specificaţia iniţială;
f. facilitarea operaţiilor ulterioare de comasare a vectorilor;
g. reducerea posibilităţilor de comasare.

Soluţia astfel obţinută este îmbunătăţită iterativ. Asupra implicanţilor funcţiei sunt efectuate trei modificări de bază:

h. reducerea,
i. restructurarea,
j. expandarea.

Este necesară o bună reprezentare a mintermenilor pentru a uşura procesarea implicanţilor. Autorii au propus utilizarea notaţiei poziţionale care permite reprezentarea unitară a mintermenilor şi a implicanţilor.

Primul tip de modificare este **reducerea** acestora la cea mai mică acoperire posibilă astfel încât funcţia să-şi păstreze acoperirea iniţială.

A doua modificare se obţine prin examinarea implicanţilor în perechi pentru a decide dacă pot fi **restructuraţi** prin reducerea unuia şi extinderea corespunzătoare a celuilalt peste aceeaşi mulţime de mintermeni.

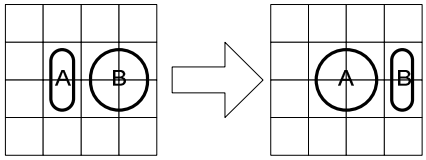


Fig. 3-3 Restructurarea implicanţilor

Cea de a treia modificare, numită **expandare**, înseamnă creşterea acoperirii fiecărui implicant la dimensiunea maximă şi eliminarea celor acoperiţi (redundanţi).

Reducerea şi reconfigurarea pot micşora acoperirea unui implicant până la mulţimea vidă (în raport cu notaţia poziţională utilizată), iar expandarea poate reduce numărul implicanţilor din soluţie prin eliminarea celor redundanţi. Reconfigurarea determină micşorarea soluţiei când este posibil acest lucru (reducerea numărului de implicanţi). Dintre cele trei operaţii, reducerea şi expandarea pot conduce la eliminarea de termeni. Ordinea în care implicanţii sunt micşoraţi, restructuraţi şi măriţi este esenţială pentru succesul procesării. Tehnica abordată impune reiterarea celor trei procesări până când nu mai este posibilă reducerea mărimii soluţiei curente.

Algoritmul este proiectat pentru minimizarea funcţiilor ale căror soluţii minime conţin maxim câteva sute de implicanţi indiferent de numărul de variabile. Majoritatea problemelor practice sunt de această natură deoarece proiectanţii lucrează de obicei cu specificaţii logice ce nu conţin mai mult de câteva sute de condiţii. Proiectantul este capabil să exprime funcţia prin câteva sute de implicanţi deoarece structura problemei ajută la evidenţierea grupării mintermenilor şi implicit la utilizarea unor termeni cu grad mare de acoperire. Metoda de minimizare are ca scop considerarea grupărilor alternative care nu pot fi observate direct din structura problemei.

Secvenţa iterativă constă din trei paşi:

1. reducerea fiecărui termen;
2. reconfigurarea termenilor analizaţi în perechi;
3. expandarea termenilor;

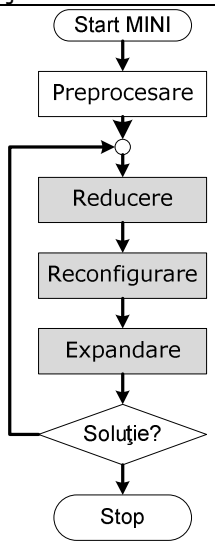


Fig. 3-4 Algoritmul MINI

Secvenţa este reluată dacă soluţia curentă este mai bună decât cea anterioară. Dacă nu s-a găsit o soluţie mai bună atunci algoritmul se opreşte, considerând soluţia actuală ca fiind finală.

Concluzii

În cadrul metodei MINI au fost definite pentru prima dată operaţiile de reducere, restructurare şi expandare a implicanţilor. Definirea şi descrierea acestor operaţii a necesitat introducerea de noi concepte:

1. vectori disjuncţi,
2. mulţimi de vectori care definesc F, $\overline{F}$ şi DC,
3. notaţie poziţională,
4. distanţă dintre doi vectori.

Operaţiile şi conceptele definite în cadrul metodei MINI se regăsesc mai târziu şi în alte metode de minimizare binare sau multivalente. Cea mai importantă observaţie a autorilor este aceea conform căreia ordinea în care sunt luaţi în calcul vectorii dintr-o soluţie temporară şi ordinea în care sunt considerate variabilele în cadrul fiecărei operaţii (de ex: transformarea vectorilor în vectori disjuncţii) influenţează convergenţa algoritmului.

Observaţia de mai sus poate avea ca şi consecinţă ideea de a crea serii de prelucrări în care ordinea considerării vectorilor nu influenţează rezultatul sau definirea de operaţii în care nu contează ordinea de considerare a variabilelor.

3.5 Metoda Espresso

În prezent, în afară de metoda Quine-McCluskey, sunt folosite şi alte metode analitice de minimizare, remarcându-se metoda ESPRESSO

[156][157], cu variantele ESPRESSO, ESPRESSO II, ESPRESSO MV[158]. ESPRESSO este o extensie şi o modernizare a metodei MINI.

Atât metoda MINI, cât şi ESPRESSO au fost dezvoltate la Universitatea Berkeley din California, SUA.

În mod implicit, ESPRESSO determină soluţii euristice, dar permite, opţional, calcularea de soluţii exacte pentru o clasă largă de sisteme decizionale. Nu toate probleme pot fi rezolvate exact deoarece metoda necesită un volum foarte mare de memorie. Se aplică pentru minimizarea funcţiilor binare cu una sau mai multe ieşiri.

Spre deosebire de metodele deja prezentate, acest algoritm [159][160] nu impune vectori complet specificaţi (mintermeni). El operează direct cu cuburi (vectori care conţin şi valori nespecificate), ceea ce conduce la reducerea volumului de date procesat. ESPRESSO şi toate aplicaţiile dezvoltate din acesta utilizează diagramele de decizie ca structuri de date pentru stocarea informaţiilor de intrare şi proprietăţile diagramelor pentru a implementa operaţii asupra funcţiilor stocate. Pornind de la algoritmul ESPRESSO, grupul de la Berkeley a dezvoltat o serie de aplicaţii care integrează progresiv din ce în ce mai multe tipuri de prelucrări legate de minimizare: ESPRESSO euristic, ESPRESSO exact, SIS , MVSIS, BALM.

O caracteristică a acestei serii de aplicaţii este descompunerea tabelelor de intrare pentru funcţii (complet sau incomplet specificate) în trei subfuncţii complet specificate care corespund valorilor de ieşire: *ON* , *DC* şi *OFF* .

Rezultatul minimizării cu ajutorul algoritmului ESPRESSO este o acoperire pentru *ON* sau pentru *OFF* , în funcţie de opţiunile selectate.

Principalele operaţii sunt reducerea, extensia şi eliminarea redundanţelor. Cu ajutorul operaţiilor de reducere şi extensie sunt determinaţi implicanţii esenţiali (care obligatoriu vor fi prezenţi în soluţie) şi implicanţii maximali. Dintre implicanţii maximali este selectat un subset care nu conţine redundanţe.

Algoritmul are posibilitatea de a minimiza independent fiecare funcţie, dar şi să minimizeze global o specificaţie care include mai multe funcţii, astfel încât să găsească implicanţi comuni.

O extensie a metodei ESPRESSO, denumită ESPRESSO MV, poate fi folosită pentru funcţii cu variabile de intrare binare sau multivalente, ieşirea putând fi binară – simplă sau multiplă – sau multivalentă simplă.

Tratarea multivalentă se reduce la o minimizare binară prin expandarea variabilelor multivalente în vectori binari, care codifică fiecare valoare din Z_N într-un vector binar cu N componente, cu un singur element 1 pe poziţia corespunzătoare valorii.

Modul acesta de tratare este impus de minimizarea binară şi orientată pe procesarea separată a fiecărei ieşiri. O ieşire este soluţionată prin folosirea în minimizare a mulţimii *ON* împreună cu mulţimea *DC* (don't care) sau a mulţimii *OFF* împreună cu mulţimea *DC* .

Codificarea 1 din N permite reconstituirea directă, după procesare, a valorii minimizate.

Metoda nu permite o minimizare pur multivalentă deoarece variabilele de intrare sau de ieşire sunt codificate binar (tip 1 din N), ceea ce conduce la creşterea substanţială a numărului de variabile prin expandarea tabelului de valori. Efectul este amplificat în cazul tratării unor funcţii multivalente cu intrări şi ieşiri multiple simple sau vectoriale, situaţie care apare frecvent în proiectarea sistemelor decizionale hardware sau software.

3.6 Aplicaţia MVSIS pentru minimizarea funcţiilor multivalente

Aplicaţia MVSIS este cel mai avansat pachet pentru minimizarea funcţiilor multivalente. Tehnica utilizată este dezvoltată plecând de la ESPRESSO. Diferenţa esenţială între MVSIS şi ESPRESSO este dată de domeniile disponibile pentru variabile[161][162]. În MVSIS pot fi utilizate variabile definite pe domenii cu mai mult de două valenţe (multivalente). MVSIS permite procesarea sistemelor singulare formate dintr-o singură funcţie, numită nod, dar şi a sistemelor complexe, formate din mai multe funcţii, specificate prin intermediul unei reţele cu mai multe noduri. Într-o reţea ieşirea unui nod poate fi utilizată ca intrare pentru alt nod.

3.6.1 Prelucrarea funcţiilor 2-Level multivalente cu MVSIS

În teoria care stă la baza aplicaţiei MVSIS sunt utilizaţi următorii termeni [163][164][165]:

Definiţie: O funcţie f este unate pozitivă în variabila x_i dacă $f_{\overline{x_i}} \subseteq f_{x_i}$.

Definiţie: O funcţie f este unate negativă în variabila x_i dacă $f_{x_i} \subseteq f_{\overline{x_i}}$.

Definiţie: O funcţie f este unate în variabila x_i dacă este unate pozitivă sau unate negativă în variabila x_i.

Definiţie: O funcţie f este unate dacă este unate în toate variabilele ei.

Definiţie: O acoperire f este unate pozitivă în variabila x_i dacă, pentru orice cub c_j din reprezentarea funcţiei $\overline{x_i} \notin c_j$.

Definiţie: O acoperire f este unate negativă în variabila x_i dacă, pentru orice cub c_j din reprezentarea funcţiei $x_i \notin c_j$.

Algoritmii ESPRESSO şi MVSIS se bazează pe exploatarea proprietăţilor funcţiilor unate [10][166][167][168]. Într-o funcţie minimă de acoperire de tip unate toate cuburile sunt esenţiale. Dacă o funcţie este

unate atunci găsirea unei astfel de acoperiri reprezintă minimul. Proprietăţile funcţiilor unate sunt utilizate în toate etapele minimizării:

1. determinarea implicanţilor primi;
2. reducerea implicanţilor;
3. complementarea etc.

Construirea diagramei de decizie

Datele de intrare, prezentate sub formă tabelară sunt, iniţial, memorate într-o diagramă de decizie utilizând pachetul de programe CUDD. Biblioteca pune la dispoziţia utilizatorului două metode de rearanjare a variabilelor: una euristică, rapidă, care urmăreşte minimul local ce se poate obţine prin interschimbarea variabilelor adiacente şi una exactă, mare consumatoare de resurse, care urmăreşte minimul global. O opţiune utilizată este aceea a integrării pe nivelele superioare a variabilelor care nu sunt unate cu scopul de a rezolva exact părţi din sistemul primit spre rezolvare[169][170][171].

Diagrama astfel obţinută este procesată iterativ cu ajutorul unor algoritmi polinomiali în vederea obţinerii rezultatului final. Complexitatea procesării revine în cea mai mare parte algoritmului de construire a diagramei de decizie. Alegerea unei soluţii optime pentru ordinea variabilelor în diagramă este o problemă NP-Hard [153].

Selectarea implicanţilor

Pe baza relaţiei existente între implicanţii determinaţi cu ajutorul diagramei de decizie şi rândurile (cuburile) specificaţiei iniţiale se construieşte o matrice de acoperire ca în metoda lui Thelen. În această matrice sunt utilizate tehnici de determinare a tautologiilor şi de eliminare a rândurilor şi coloanelor dominate.

Definiţie: Dacă un implicant i este singurul care acoperă un rând r atunci i se numeşte esenţial.

Implicanţii esenţiali se reţin în soluţie şi sunt eliminaţi din matrice împreună cu rândurile pe care le acoperă. Un implicant esenţial poate acoperi mai multe rânduri.

Definiţie: Un rând r_1 ai cărui implicanţi sunt conţinuţi în setul de implicanţi ai rândului r_2 domină rândul r_2

Rândul r_2 este dominat şi poate fi eliminat. O situaţie particulară o reprezintă perechile de rânduri acoperite de acelaşi set de implicanţi. În această situaţie este eliminat unul dintre rânduri.

Definiţie: Un implicant i_1 a cărui mulţime de rânduri acoperite reprezintă un superset al mulţimii de rânduri acoperite de implicantul i_2 se numeşte dominant.

Implicantul i_2 este dominat şi poate fi eliminat.

După ce se fac toate eliminările se obţine o matrice numită **nucleu ciclic**. Nucleul ciclic este rezolvat euristic prin selectarea cu prioritate a implicanţilor care acoperă mai multe rânduri din tabelul iniţial.

3.6.2 Prelucrarea funcţiilor multinivel cu MVSIS - reţele

Aplicaţia MVSIS pentru procesarea sistemelor multivalente de tip reţea este o generalizare a aplicaţiei SIS dezvoltată pentru sistemele binare. Este suficientă prezentarea aplicaţiei MVSIS (fără a mai trece în revistă aplicaţia SIS) deoarece include toate conceptele dezvoltate pentru cazul binar.

La baza acestei aplicaţii stă un set de principii, metode şi algoritmi dezvoltaţi de-a lungul timpului la Universitatea din Berkeley. În acest subcapitol este prezentată o sinteză a acestei teorii. Trebuie precizat faptul că o parte din aceste concepte sunt încă în curs de implementare.

Sursa nedeterminismului este dată de flexibilitatea maximă [172] obţinută în cadrul procesului de minimizare a unui nod.

Prin flexibilitate se înţelege mulţimea de valori ce pot fi prezente la ieşirea sistemului pentru o combinaţie de intrare dată. În cazul binar, o mulţime de valori cu mai mult de un element este reprezentată doar de mulţimea $\{0,1\}$, adică de tot domeniul de valori. În cazul multivalent este posibilă existenţa unei mulţimi de valori de ieşire cu cel puţin două valenţe, care să nu fie identică cu tot domeniul.

În cazul binar nedeterminismul se manifestă la funcţiile incomplet specificate. Părţii nespecificate a funcţiei îi corespunde la ieşire valoarea **DC** ($\{0,1\}$) Flexibilitatea este utilizată pentru a crea un echivalent minim determinist sau nedeterminist al nodului.

Funcţiile nedeterministe se pot clasifica astfel:

- funcţii cu mai multe valori de ieşire pentru o intrare dată;
- funcţii incomplet specificate.

Ieşirea poate lua orice valoare dintr-o submulţime strictă, dată, a domeniului de ieşire.

Comportamentul unui sistem este definit prin corespondenţa intrare – ieşire, ce poate apărea la momentul simulării acestuia. Simularea începe cu o evaluare a intrării, continuând cu evaluarea în ordine topologică a întregului sistem. Sistemele nedeterministe permit în timpul evaluării interpretări diferite.

Dezvoltarea modelului pentru această metodă a însemnat:

1. definirea următoarele operaţii logice: descompunere, substituţie, eliminare, comprimare, amestecare şi minimizare.
2. studiul limitelor în care pot fi aplicate aceste operaţii fără a denatura specificaţia iniţială.
3. conceperea de algoritmi pentru calculul flexibilităţii complete.

Flexibilitatea completă depinde de tipul de comportament ales. Determinarea flexibilităţii complete face parte din soluţia adoptată pentru minimizare.

Modelul de minimizare corespunde unei tehnici de tip „divide et impera", permiţând procesări separate a părţilor componente ale sistemului. Rezultatele sunt combinate pentru a se obţine minimizarea întregului sistem. Sistemele sunt definite ca reţele compuse din noduri.

Definiţie: O reţea nedeterministă este un graf aciclic orientat. Un nod reprezintă o relaţie nedeterministă între intrările nodului şi ieşirea singulară multivalentă a acestuia. Un arc este orientat de la nodul i la nodul j dacă relaţia din nodul j depinde sintactic de variabila asociată nodului i.

Definiţie: O relaţie este bine definită dacă pentru fiecare combinaţie de intrare există cel puţin o valoare de ieşire.

Sistemele cu mai multe ieşiri sunt definite ca două relaţii având acelaşi domeniu de intrare şi domenii diferite de valori.

Considerând trei domenii de valori D_0, D_1, D_2, se defineşte o relaţie $F : D_0 \to D_1 \times D_2$. Dacă, pentru o intrare x ieşirea sistemului este $\{(0,0),(0,1),(1,0)\}$ se pune problema existenţei sau inexistenţei valorii de ieşire $(1,1)$.

Definiţie: Fie o funcţie $F : D_0 \to D_1 \times D_2$. Pentru orice intrare $i \in D_0$, mulţimea de valori de ieşire corespunzătoare domeniului D_1 este $S_1^i \subset D_1$ şi pentru D_2 este $S_2^i \subset D_2$. Funcţia F este simetrică dacă $F(i) = S_1^i \times S_2^i, \forall i \in D_0$.

Pentru implementare este convenabil ca funcţia să fie reprezentată ca o mulţime de funcţii deterministe cu ieşiri binare (câte o funcţie pentru fiecare valenţă a fiecărui domeniu de ieşire).

Definiţie: Un sistem este conform cu un alt sistem specificat dacă domeniile de definiţie şi de valori sunt identice cu ale celui specificat, iar mulţimea de valori de ieşire ale acestuia este inclusă în mulţimea de valori de ieşire corespunzătoare din sistemul specificat sau egală cu aceasta.

Prin comportamentul unui sistem se înţelege mulţimea tuturor corespondenţelor intrare – ieşire.

Sunt definite trei modele de comportament: normal (NS), normal-compatibil (NSC) şi mulţime (SS). Ele diferă prin modul în care se fac alegerile în situaţiile nedeterministe.

Definiţie: Toate comportările unui sistem sunt reprezentate de reuniunea celor trei categorii de comportamente.

Definiţia conformităţii este strâns legată de aceste comportamente.

Normal Simulation (NS)

NS este considerat comportamentul cel mai apropiat de realitate. Sistemul, văzut ca o reţea, după ce primeşte valoarea de intrare, evaluează

fiecare nod şi alege în mod aleator una dintre valorile de ieşire, propagând-o în ordine topologică, până când au fost evaluate toate ieşirile.

Comportamentul NS este uşor de simulat prin perechi intrare – ieşire singulare. Este mult mai dificil să se obţină toate perechile de valori intrare – ieşire posibile.

Din toate cele trei metode de simulare aceasta este cea mai complexă din punct de vedere al procesării. Metoda de minimizare aleasă pentru acest comportament modelează un nod printr-o funcţie cu mai multe ieşiri binare deterministe. Rezultatul este un sistem binar monolitic.

Acestei tehnici i-a fost adăugată o metodă simplă, utilă mai ales dintr-un punct de vedere conceptual. Nedeterminismul este înlăturat printr-o aşa-numită determinizare a intrărilor. Pentru fiecare variabilă de ieşire cu caracter nedeterminist este adăugată o variabilă de intrare cu domeniul de valori identic cu al variabilei de ieşire şi care primeşte o valoare identică cu cea a variabilei de ieşire. Fiecare specificaţie nedeterministă este multiplicată pentru fiecare valoare de ieşire în parte. Rezultatul este un sistem determinist incomplet specificat.

NSC – NS Compatibil (NSC)

Acest tip de comportament este identic cu NS, dar fiecare funcţie (ieşire) a sistemului este minimizată separat. Comportamentul întregului sistem este definit ca produs cartezian al tuturor ieşirilor. Rezultatul este un sistem cu ieşiri compatibile deoarece se pierde corelarea dintre valorile de ieşire. Pentru un sistem cu o singură ieşire comportamentele NS şi NSC sunt identice. Raportat la NS, NSC este mai uşor de procesat.

Teoremă: Comportamentul NSC este echivalent cu compactarea reţelei în ordine topologică inversă.

Evaluarea în ordine topologică inversă este echivalentă cu determinizarea reţelei prin introducerea pseudo-intrărilor, urmată de eliminarea acestora.

O altă proprietate determinată pentru comportamentul NSC este că el conduce spre obţinerea celei mai mici reţele cu ieşiri simetrice, care include comportamentul NS. Deoarece comportamentul NSC este simplu de procesat, este utilizat pentru calculul limitei superioare a comportamentului NS.

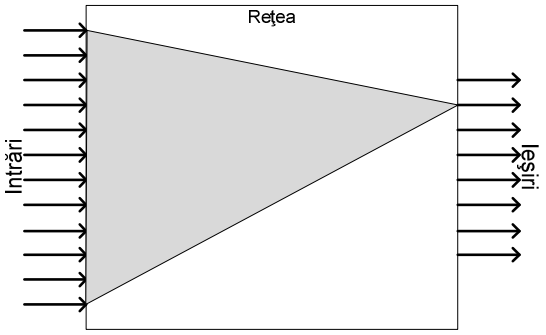


Fig. 3-5 Procesarea independentă a ieşirilor

O cale uşoară pentru procesarea NSC este extragerea fiecărui „con" corespunzător fiecărei ieşiri urmată de procesarea independentă a acestuia. Diferenţa dintre NS şi NSC devine astfel evidentă, deoarece noduri interne ale grafului pot avea valori de ieşire distincte pentru procesarea de ieşiri distincte ale sistemului.

Teoremă: Comportamentele NS şi NSC nu se schimbă dacă sunt eliminate noduri deterministe. Consecinţa acestei teoreme este că eliminarea pseudo-intrărilor adăugate pentru determinizarea sistemului nu schimbă comportamentul acestuia.

Comportamentul SS – Set Simulation

În cadrul acestui comportament fiecărui semnal îi este atribuită o mulţime de valori în locul unei singure valori. Parcurgerea se face în ordine topologică. Ieşirea unui nod este reprezentată de mulţimea tuturor valorilor posibile de ieşire.

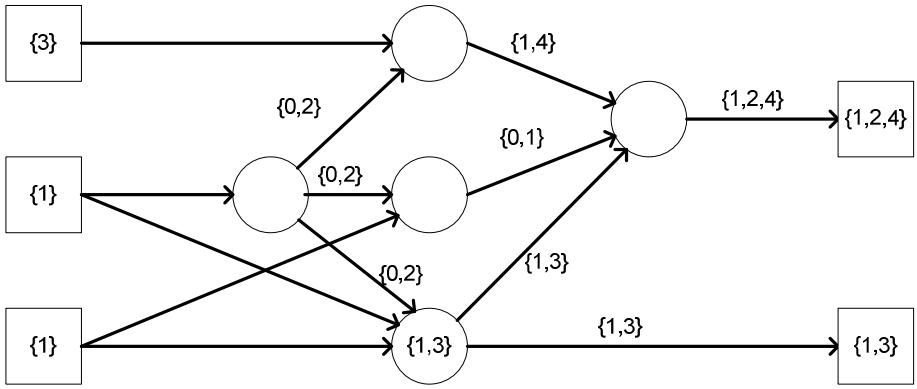


Fig. 3-6 Comportament SS

Tabelul de valori corespunzător intrării 3 1 1 este:

3 1 1	/	1 1
3 1 1	/	1 3
3 1 1	/	2 1
3 1 1	/	2 3
3 1 1	/	4 1
3 1 1	/	4 3

Tab. 3-1 Relaţie nedeterministă corespunzătoare comportamentului SS

O posibilă tratare a acestui comportament este următoarea:

1. Tratarea separată a fiecărei valenţe de ieşire descrisă printr-o funcţie binară ce indică posibilitatea prezenţei valenţei la ieşirea sistemului (1 este posibil, 0 nu este posibil)
2. Minimizarea globală a reţelei
3. Recombinarea funcţiilor binare astfel încât să se obţină variabilele multivalente iniţiale.

Această tratare este considerată foarte eficientă şi este implementată în aplicaţia MVSIS.

Specificare externă

O specificare externă indică mulţimea comportamentelor observabile din afară. Orice comportament sintetizat trebuie să fie bine definit şi să fie inclus în specificarea externă. Aceasta poate fi simetrică (comportament independent pentru fiecare ieşire) sau generală (valorile ieşirilor sunt relaţionate).

Observaţie - Comportamentele NSC şi SS pierd corelarea dintre valorile de ieşire atunci când este construită câte o funcţie pentru fiecare ieşire în parte, dar au avantajul că reprezentarea este mult comprimată. Sunt avantajoase când se ştie că nu există corelare între ieşiri, dar sunt aplicabile unei clase restrânse de probleme, deoarece denaturează comportamentul sistemului.

$$NS \subseteq R^{spec} \subseteq NSC \subseteq SS$$

Pe de altă parte, un comportament este conform dacă reprezintă o submulţime a specificaţiei externe. În aceste condiţii NSC şi SS sunt viabile doar pentru sisteme simetrice.

O tehnică utilizată pentru NSC şi SS este adăugarea de variabile suplimentare care să ajute la simetrizarea sistemului adăugând preudo-intrări.

În cazul sistemelor determinste nu se pune problema necorelării variabilelor de ieşire deoarece nu mai există diferenţă între NS, NSC şi SS.

Minimizarea

Procesul de minimizare constă în două etape:

1. Calculul flexibilităţii
2. Înlocuirea reprezentării logice iniţiale cu una conţinută în flexibilitate.

Cu aplicaţia MVSIS, pornind de la tabelul iniţial şi calcularea flexibilităţii deschide calea către mai multe forme minimizate ale funcţiei date. În tabelele de mai jos este prezentată o posibilă secvenţă de minimizare pentru o funcţie cu două intrări multivalente x şi y.

x/y	0	1	2	3	4
0	2	3	3	1	3
1	6	3	3	1	3
2	6	3	7	5	7
3	4	0	4	4	4

Tab. 3-2 Tabel iniţial

x/y	0	1	2	3	4
0	012345	012345	01234567	012345	135
1	6	135	1357	1357	67
2	6	135	1357	1357	67
3	024	024	0246	4	012345

Tab. 3-3 Determinarea flexibilităţii

x/y	0	1	2	3	4
0	5	5	5	5	5
1	6	5	7	7	7
2	6	5	7	7	7
3	4	4	4	4	4

Tab. 3-4 Posibil rezultat al minimizării

Determinarea flexibilităţii maxime (complete)

Pentru determinarea flexibilităţii complete sunt utilizate două metode: una corespunde comportamentelor NS şi NSC şi alta comportamentului SS.

Flexibilitatea pentru NS şi NSC

Definiţie: O reţea căreia i se taie nodul i este o reţea derivată din cea originală prin înlocuirea ieşirii nodului i cu o variabilă y_i considerată intrare adiţională (pe lângă celelalte intrări).

Calculul flexibilităţii maxime se bazează pe determinarea valorilor variabilei y_i astfel încât sistemul să fie conform cu specificaţia iniţială. Valorile determinate pentru y_i se numesc observabilitate parţială.

Teoremă: Pentru NS şi NSC observabilitatea parţială este maximă. Dacă o funcţie deterministă $y_i = f_i(\textit{intrari primare})$ este utilizată pentru înlocuirea nodului i astfel încât comportamentul lui y_i să nu fie inclus în observabilitatea parţială a lui y_i din reţea, atunci comportamentul total al sistemului utilizând funcţia f_i nu mai corespunde specificaţiei externe.

Flexibilitatea pentru SS

Pentru a determina flexibilitatea în comportamentul SS trebuie determinată mulţimea de valori a fiecărui nod intern pentru toate combinaţiile de intrare. De exemplu, dacă unui nod intern j îi corespunde o mulţime de valori dintr-un domeniu trivalent $\left\{b_0^j, b_1^j, b_2^j\right\}$, pentru a procesa reţeaua sunt adăugate trei semnale de intrare b_0^j, b_1^j şi b_2^j. Pentru mulţimea de valori {0,2} cele trei semnale de intrare vor fi (1,0,1). Între noile intrări şi ieşirea y_j a nodului se stabileşte relaţia $R\left(b_0^j, b_1^j, b_2^j, y_j\right)$. Pentru exemplul dat (1,0,1,**0**) şi (1,0,1,**2**) sunt în relaţie, dar (1,0,1,**1**) nu este în relaţie deoarece valoarea 1 nu aparţine mulţimii de valori.

Flexibilitatea completă este obţinută ca o relaţie booleană cu mai multe ieşiri prin relaţionarea combinaţiilor iniţiale de intrare (termenii funcţiei) cu noile variabile b_i^j introduse.

3.7 Alte aplicaţii care utilizează reprezentări tabelare

Aplicațiile prezentate în acest capitol se aplică numai funcțiilor binare.

3.7.1 Algoritmul FC-Min

Algoritmul FC-Min [173][174] (Find Coverage) a fost creat la Universitatea din Praga pentru minimizarea funcţiilor binare cu mai multe ieşiri. Autorii au urmărit să găsească o metodă care să rezolve euristic, dar rapid, probleme cu un număr foarte mare de intrări (de ordinul sutelor) şi cu număr foarte mare de rânduri (tot de ordinul sutelor). Metoda prezentată propune descompunerea spaţiului de căutare ghidată de cele mai frecvent utilizate valori ale variabilelor. Autorii au reuşit să obţină cu ajutorul acestui algoritm rezultate mai bune decât algoritmul ESPRESSO pentru funcţiile binare.

Iniţial, conform acestei metode, sunt contorizate apariţiile fiecărei valenţe pentru fiecare variabilă în tot tabelul de intrare. Numărul de apariţii este sortat în ordine descrescătoare şi este selectată variabila a cărei valoare apare cel mai frecvent. Spaţiul de intrare este descompus pe baza valorilor acestei variabile. Cum această metodă tratează numai funcţiile binare, se obţin doi vectori cu o singură poziţie specificată, cea a variabilei selectate. Se testează dacă valorile de ieşire ale rândurilor acoperite de fiecare dintre cele două cuburi sunt identice. Dacă, pentru un cub, rândurile acoperite nu au aceeaşi valoare de ieşire, atunci acesta nu este implicant, şi algoritmul continuă cu selectarea unei noi variabile de intrare şi a unei valenţe a acesteia până când rândurile acoperite de fiecare cub în parte au aceeaşi ieşire.

În urma aplicării acestei metode se obţine un set de implicanţi care acoperă funcţia iniţială. Pentru a verifica dacă implicanţii sunt maximali algoritmul continuă cu verificarea fiecărui literal din fiecare implicant. Dacă acel literal poate fi eliminat fără să acopere şi un rând cu ieşire diferită atunci literalul este eliminat permanent şi se continuă verificarea cu următorul literal.

Dintre implicanţii maximali este ales un subset utilizând o metodă Greedy ghidată după numărul de rânduri iniţiale acoperite de fiecare implicant maximal în parte. Se calculează numărul de rânduri acoperite de fiecare implicant maximal, sunt sortaţi implicanţii în ordine descrescătoare după această valoare şi este ales primul implicant. Sunt eliminate dintre rândurile iniţiale rândurile acoperite de implicantul ales şi algoritmul de

selecţie continuă având ca date de intrare implicanţii neselectaţi şi rândurile rămase. Algoritmul se opreşte când au fost eliminate toate rândurile.

Ieşirile multiple afectează direct modul de aplicare al algoritmului. Acesta a fost extins pentru a putea fi aplicat şi funcţiilor cu ieşiri multiple.

3.7.2 Metoda CD (Coverage Directed)

Metoda de minimizare CD a fost dezvoltată la Universitatea din Praga şi vizează minimizarea funcţiilor booleene definite pe o porţiune restrânsă a domeniului de definiţie. Principiul de bază al metodei este reducerea graduală a unor termeni cu grad mare de acoperire până când este generat un implicant. Scopul declarat al acestei metode este de a găsi o soluţie într-un timp mic de procesare pentru funcţii cu un număr de variabile de ordinul sutelor şi un număr de rânduri tot de ordinul sutelor.

Metoda este proiectată numai pentru funcţii booleene care nu sunt definite pentru o parte semnificativă a spaţiului de intrare. Pentru o funcţie cu n intrări binare, pe care sunt definite un număr de u rânduri cu ieşirea 1 şi z rânduri cu ieşirea 0, există relaţia $u + z << 2^n$.

Algoritmul determină literalul utilizat cel mai frecvent pentru rândurile cu ieşirea 1. Dacă vectorul format din acest literal nu este implicant (rezultat obţinut în urma comparării vectorilor cu rândurile cu ieşirea 0) se adaugă un nou literal, şi tot aşa, până când se obţine un implicant. În continuare sunt eliminate rândurile acoperite de implicantul obţinut şi se reia algoritmul până sunt acoperite toate rândurile cu ieşirea 1. Dacă la un moment dat există mai multe literale cu acelaşi număr de apariţii, este selectată în mod aleator unul dintre ele.

Fiecare implicant astfel obţinut este expandat prin teste de eliminare a literalelor. Dacă un literal poate fi eliminat fără a acoperi vreun rând cu ieşirea 0, atunci eliminarea devine permanentă. În caz contrar literalul este pus la loc şi se continuă cu următorul literal.

Soluţia finală utilizează o tehnică GREEDY astfel: sunt selectaţi implicanţii care acoperă rândurile acoperite de cel mai mic număr de implicanţi. Dacă există mai mulţi astfel de implicanţi, sunt selectaţi cei care acoperă un număr mai mare de rânduri.

Metoda este liniară din punct de vedere al numărului de intrări şi proporţională cu pătratul numărului de rânduri. Rezultatele au fost obţinute experimental. Nu a putut fi evaluată exact complexitatea algoritmului.

3.7.3 BOOM

Aplicaţia BOOM [12] pentru minimizarea funcţiilor binare complet sau incomplet specificate a fost dezvoltată de P. Fišer şi J. Hlavička la Universitatea din Praga. Metoda implementată utilizează un sistem de generare de tip top-down prin care, în loc să încerce eliminarea de literale

pentru a obţine implicanţi, sunt reduşi termeni generali progresiv prin adăugarea de literale. Metoda este euristică şi, aşa cum afirmă autorii, obţine rezultate deosebite pentru funcţii cu foarte multe variabile de intrare, dar cu un număr relativ mic de termeni. Principalele diferenţe dintre nucleul aplicaţiilor ESPRESSO şi BOOM sunt:

- modul în care sunt tratate definiţiile funcţiilor. ESPRESSO utilizează secvenţe reducere/expandare pentru a determina forme minimizate cu mai puţini termeni şi BOOM utilizează funcţii *ON* şi *OFF* pentru a determina dacă o soluţie determinată la un moment dat este corectă sau nu.
- modul de abordare. ESPRESSO utilizează operaţia de descompunere disjunctă a termenilor dintr-o soluţie parţială şi expandarea acestora prin eliminarea de literale, iar BOOM porneşte de la o mulţime de supercuburi care acoperă termenii iniţiali ai unei funcţii pe care le reduce prin adăugarea de literale până devin implicanţi.

Algoritmul BOOM–I

Metoda este considerată utilă mai ales pentru funcţiile cu mai multe variabile de intrare (de ordinul miilor) şi cu un număr relativ mic de termeni (rânduri). Această metodă are în vedere funcţiile cu mai multe ieşiri.

Algoritmul propus se utilizează acolo unde alte aplicaţii de minimizare nu dau rezultate datorită timpului mare de procesare. Acest algoritm este o continuare a metodei prezentate anterior.

Metoda de determinare a implicanţilor este aceeaşi (coverage directed): reducerea cuburilor n dimensionale prin adăugarea de literale până când sunt acoperiţi termenii pentru valoarea 1. Faţă de metoda anterioară, dacă există mai multe literale cu aceeaşi frecvenţă de apariţie, sunt testate toate variantele (o căutare care parcurge toate posibilităţile nivelului următor înainte de a alege o variantă).

Implicanţii găsiţi sunt expandaţi, dacă este posibil, prin eliminarea de literale, având în vedere şi eliminarea redundanţelor.

Generarea de implicanţi şi expandarea acestora se face pentru fiecare ieşire în parte. Pentru a obţine o soluţie globală cât mai bună, considerând că este posibil ca cei mai buni implicanţi globali să nu fie neapărat maximali pentru fiecare funcţie, algoritmul îi reduce prin adăugarea de literale astfel încât să se obţină implicanţi comuni funcţiilor date. Determinarea lor se face prin intersecţia mulţimilor de implicanţi obţinuţi pentru fiecare funcţie în parte.

Soluţia globală este determinată utilizând acelaşi algoritm de tip GREEDY ca şi în metoda prezentată anterior.

O variantă a acestui algoritm, în etapa de generare a implicanţilor, utilizează o selectare aleatorie dintre variantele echivalente (literale echipotenţiale). Selecţia aleatorie deschide calea către o extensie a algoritmului, în sensul că acesta poate fi reluat, şi mulţimii de implicanţi determinaţi în cadrul rulării curente i se adaugă noi implicanţi. În acest fel

creşte probabilitatea de a alege implicanţi mai buni în selecţia soluţiei finale. Această metodă este dublată de o serie de tehnici care au ca scop diminuarea redundanţei în calcule.

Algoritmul BOOM–II

În cadrul aplicaţiei BOOM–II [13][175] au fost aduse unele îmbunătăţiri metodei iniţiale BOOM–I. BOOM–II este o continuare a metodelor de minimizare BOOM–I şi FC–Min. Fiecare dintre aceste metode are propriul domeniu de aplicare în care este eficientă. Continuarea studiului şi rafinării celor două metode are ca scop crearea uneia noi, care combină avantajele fiecăreia dintre ele. Rezultatul este o metodă cu un timp bun de procesare şi cu soluţii cât mai apropiate de optim.

Metoda tratează funcţiile binare cu mai multe ieşiri.

Implicanţii generaţi atât de BOOM, cât şi de FC–Min sunt puşi laolaltă şi soluţia finală este dată rezolvând problema acoperirii prin utilizarea tuturor implicanţilor. Suplimentar faţă de cele două metode de la care s-a plecat, BOOM–II este prevăzut cu posibilitatea adăugării de constrângeri definite de utilizatori, care să ghideze algoritmul. Dintre acestea se evidenţiază cele legate de modul de descompunere al funcţiei iniţiale (cu scopul de a fi construiţi termeni comuni pentru ieşirile funcţiei), şi cele legate de sinteza finală (având ca obiectiv obţinerea unor circuite uşor de testat şi o proiectare uniformă (echilibrată) a circuitului). Constrângerile se manifestă prin modificarea corespunzătoare a procedurii de expandare a implicanţilor (partiţionarea uniformă) şi în modul de reducere a acestora (modificarea funcţiei „criteriu” pentru alegerea noului literal).

3.7.4 Concluzii pentru FC-MIN, CD-Search, BOOM–I şi BOOM–II

Această serie de aplicaţii este utilă pentru funcţiile binare definite pe un subspaţiu restrâns al domeniului de definiţie, deterministe, cu număr foarte mare de intrări. Pachetul prevede o serie de opţiuni aplicate pentru ghidarea algoritmului, astfel încât să se obţină rezultate cât mai bune (relativ la domeniul de utilizare al funcţiei minimizate).

3.8 Metoda discriminării

Metoda discriminării a fost propusă de I. Stefănescu [176][177] în 2006. Având în vedere importanţa metodei aceasta va fi prezentată într-un capitol separat.

Metoda este creată pentru minimizarea funcţiilor multivalent cu ieşiri multiple. Principiul pe care se bazează această metodă este cel al discriminării. Metoda urmăreşte identificarea diferenţelor dintre intrările sistemului care generează ieşiri distincte. Scopul urmărit este de a crea un

set minim de teste necesar pentru a stabili ieşirea sistemului pentru o intrare dată.

În cadrul algoritmului vectorii iniţiali sunt grupaţi în funcţie de ieşirile corespunzătoare obţinându-se grupele de ieşiri. Pentru fiecare vector de intrare se creează lista de vectori acoperitori prin înlocuirea valorilor singulare cu valori **DC**. Vectorii acoperitori au un număr de elemente specificate mai mic sau egal cu cel al vectorului iniţial. Se păstrează dintre vectorii acoperitori doar aceia care nu au mintermeni comuni cu vectorii iniţiali din celelalte grupe de ieşire şi care nu sunt acoperiţi de alţi vectori din aceeaşi grupă. Aceştia sunt numiţi vectori discriminanţi.

Vectorii rămaşi sunt implicanţi. Dintre aceştia este ales un subset astfel încât să nu existe redundanţe.

Concluzii

Metoda discriminării, spre deosebire de alte metode, nu determină şi nu analizează în mod explicit submulţimea valorilor intrării pentru care funcţia nu este specificată [178].

Utilizarea metodei discriminării are următoarele avantaje: abordează direct multivalent problema minimizării, nefiind necesare nici un fel de conversii intermediare în binar; permite minimizarea tabelelor de valori care utilizează variabile definite pe domenii multivalente (binare, ternare, etc.); variabilele dintr-un tabel pot fi definite pe domenii diferite (neomogene); permite minimizarea funcţiilor cu intrări simple sau vectoriale şi cu ieşiri singulare sau multiple.

3.9 Tehnici auxiliare folosite în minimizare

Aplicaţiile dezvoltate pentru minimizare conduc la obţinerea unei mulţimi de implicanţi (totală sau parţială). Dintre aceştia o parte sunt redundanţi. Determinarea unui subset fără redundanţe necesită un volum mare de calcul. În sprijinul acestei analize au fost dezvoltate metode care permit alegerea unui subset minim de implicanţi [179][180][181][182].

3.9.1 Metoda lui Petrick pentru selectarea soluţiilor fără redundanţe

Numărul de soluţii care se pot forma utilizând metoda Quine-McCluskey este foarte mare. Aplicarea metodei conduce la obţinerea mulţimii complete de vectori implicanţi. Dintre aceştia trebuie ales un subset suficient care să acopere toţi vectorii iniţiali ai funcţiei. S. R. Petrick a descris o metodă analitică [183] care să vină în completarea metodei Quine-McCluskey astfel încât să determine o submulţime optimă de vectori implicanţi.

Metoda se bazează pe utilizarea unei matrice care înglobează relaţia de acoperire dintre vectorii implicanţi determinaţi şi rândurile tabelului prin care a fost specificată funcţia. Fiecare vector iniţial este exprimat printr-o disjuncţie formată din vectorii implicanţi care acoperă rândul. Produsul acestor disjuncţii conduce prin transformarea lui într-o expresie disjunctivă (SOP - sum of products) la obţinerea tuturor soluţiilor posibile. Fiecare termen reprezintă o soluţie. Sunt selectaţi toţi termenii care au valoarea minimă pentru metrica dată. Dacă nu există o informaţie suplimentară care să ajute la alegerea unuia dintre aceştia ca soluţie, atunci oricare dintre ei reprezintă o soluţie minimă.

Matricea de acoperire reprezintă o corespondenţă binară între vectorii implicanţi şi vectorii iniţiali. Matricea este completată cu valori binare. Vectorii implicanţi sunt notaţi P_j, unde i indică un număr de coloană din matricea de acoperire. Vectorii din tabelul iniţial sunt numerotaţi pornind de la valoarea 1. Fiecărui rând din matrice îi va corespunde un vector iniţial conform cu numerotarea făcută. La intersecţia dintre un implicant şi un rând din tabelul iniţial se memorează valoarea 1 dacă implicatul acoperă vectorul şi 0 în caz contrar.

Metoda determină toate soluţiile utilizând o formă algebrică P de scriere sub formă de produs cartezian al tuturor mulţilor de implicanţi care acoperă fiecare rând iniţial în parte. Dacă, de exemplu, rândul 6 este acoperit de implicanţii P_4 şi P_6 atunci unul dintre factorii expresiei algebrice P este $(P_4 + P_6)$.

În continuare se transformă soluţia din a doua formă canonică în prima formă canonică prin înmulţirea cu prioritate a factorilor care au mai mulţi termeni comuni (evitând astfel o creştere prea rapidă a numărului de termeni din factorii intermediari). În final, după ce sunt eliminaţi termenii redundanţi, sunt selectate toate soluţiile care au metrica stabilită minimă. Metrica, de exemplu, poate fi dată de numărul de implicanţi.

Pentru un tabel format din n rânduri, fiecare acoperit de k_i implicanţi ($i = 1..n$), se obţin $\prod_{i=1}^{n} k_i$ termeni, din care urmează să fie eliminaţi termenii redundanţi.

Metoda lui Petrick permite integrarea metricii definite ca măsură a minimizării pentru evaluarea unei soluţii. În exemplul prezentat soluţiile posibile au fost comparate utilizând ca metrică numărul minim de implicanţi. Pot fi considerate şi alte metrici cum ar fi:

- numărul de literale din implicanţi,
- numărul de literale distincte din implicanţii utilizaţi,
- diverse combinaţii ale criteriilor existente.

Concluzii

Transformarea unei expresii din formă conjunctivă în formă disjunctivă implică o procesare complexă, în sensul că numărul de termeni obţinuţi este egal cu produsul numărului de termeni din fiecare factor al

conjuncţiei. Această metodă trebuie susţinută cu un set de reguli după care să fie efectuate calculele astfel încât să conducă la eliminarea termenilor redundanţi cât mai devreme (urmărind eliminarea redundanţelor de calcul).

Metoda necesită o formă analitică mai evoluată pentru a fi aplicată algoritmic. Aşa cum era de aşteptat, au apărut rapid în domeniul transformării unei expresii din formă canonică în formă disjunctivă variante analitice ale acestei metode, precum şi alte tehnici înrudite, bazate pe principiul metodei lui Petrick.

3.9.2 Teorema lui Nelson pentru transformarea din CNF în DNF

În sprijinul metodei lui Petrick vine teorema lui Raymond J. Nelson [184][185], prin care o expresie algebrică poate fi transformată din forma conjunctivă în forma disjunctivă urmărind eliminarea cât mai timpurie în cadrul procesării a termenilor redundanţi. Transformarea, aşa cum este propusă de Nelson, se bazează pe înmulţirea propriu-zisă a termenilor din factori, eliminând dintre termenii obţinuţi pe cei redundanţi sau pe cei care conţin ambele forme (pozitivă şi negată) ale aceleiaşi variabile.

Această procesare poate fi utilizată atât pentru determinarea soluţiei minime, cât şi pentru simplificarea algebrică a funcţiilor logice binare.

Cele trei reguli de simplificare sunt:

R1: eliminarea termenilor care conţin ambele forme ale unei variabile:

$$a\bar{a} = 0 \tag{3-1}$$

R2: eliminarea din termeni a dublurilor variabilelor:

$$abac = abc \tag{3-2}$$

R3: eliminarea termenilor incluşi în alţi termeni:

$$abac + abc = abc \tag{3-3}$$

Aceste reguli îşi produc efectul începând cu termenii şi disjuncţiile intermediare obţinute în cadrul procesului de calcul al produsului cartezian.

Concluzii

Această procesare poate fi utilizată atât pentru determinarea soluţiei minime cât şi pentru simplificarea algebrică a funcţiilor logice binare. Faţă de metoda lui Petrick, teorema lui Nelson asigură o bază analitică pentru determinarea termenilor redundanţi. Chiar dacă poate fi aplicată în etapele intermediare de calcul pentru eliminarea calculului redundant, totuşi nu prevede un set de reguli care să dirijeze modul în care se desfăşoară calculul. Metodele prezentate până acum asigură suportul teoretic, dar nu oferă o metodă analitică completă pentru reducerea complexităţii procesării.

3.9.3 Algoritmul lui Thelen şi extensii ale acestuia

Algoritmul lui Thelen [186][187] vine în sprijinul teoremei lui Nelson, în sensul că asigură suport pentru memorarea adecvată a soluţiilor obţinute în urma minimizării şi, mai mult de atât, oferă o metodă analitică de aplicare a regulilor pe o diagramă de decizie. Pe măsură ce sunt calculaţi termenii formei disjunctive, aceştia sunt adăugaţi în diagrama de decizie. Algoritmul se poate aplica pentru determinarea celei mai bune soluţii, dar şi pentru determinarea setului de implicanţi maximali, deoarece permite determinarea termenilor redundanţi.

Algoritmul propus de Thelen reprezintă o extensie a teoremei lui Nelson în care cele trei reguli sunt implementate în termenii BDD. Astfel, cele trei reguli se transformă în următoarele reguli de eliminare a arcurilor unui BDD construit pentru înmulţirea factorilor:

R1: Un subarbore este eliminat (nu mai este dezvoltat) dacă rădăcina lui conţine forma conjugată a unui literal ce apare într-un nod predecesor.

$$a\bar{a} = 0 \tag{3-4}$$

R2: O disjuncţie nu mai este luată în considerare dacă ea conţine un literal ce apare într-un nod predecesor.

$$abac = abc \tag{3-5}$$

R3: Un subarbore este eliminat (nu mai este dezvoltat) dacă are în rădăcină un literal care apare pe un nivel superior încă neexplorat.

$$abac + abc = abc \tag{3-6}$$

Mathony H-J a adăugat la cele trei reguli ale lui Thelen alte 3 reguli [233]:

R4: Un subarbore este eliminat dacă există un subarbore deja dezvoltat care porneşte de pe un nivel superior, cu acelaşi literal ca rădăcină, pe care nu a fost aplicată regula R3. Această regulă reduce arborele de căutare cu până la 25%.

Adăugarea regulii R4 complică foarte mult algoritmul, deoarece trebuie memorate căi deja parcurse pentru a aplica această regulă.

R5, **R6**: Celelalte două reguli **R5** şi **R6** se referă la complementarea unei expresii. Ele sunt în relaţie la fel ca şi regulile **R3** şi **R4**. În cadrul operaţiei de complementare a unei expresii algebrice în forma disjunctivă, pentru fiecare pereche de factori care conţin unul forma normală şi unul forma complementată ale unui literal, este ales în anumite condiţii unul dintre factori şi este utilizat numai literalul respectiv, făcând abstracţie de celelalte literale din el.

Concluzii

Algoritmul lui Thelen este utilizat în implementarea bibliotecilor de funcţii pentru prelucrarea funcţiilor logice. Problemele ce pot fi rezolvate cu acesta sunt:

- determinarea implicanţilor esenţiali,
- determinarea acoperirii minime,

- eliminarea redundanţelor.

Principala calitate a algoritmului este oferirea unei metode analitice pentru procesare. În schimb acesta nu este exact, în sensul că rămân termeni redundanţi nedetectaţi. Pentru eliminarea acestora sunt necesare alte metode.

3.10 Concluzii

Metodele şi algoritmii pentru minimizare prezentaţi pot fi clasificaţi în funcţie de diverse criterii:

- tipul metodei;
- domeniul variabilelor;
- forma de reprezentare a datelor;
- tipul de ieşire al funcţiilor logice minimizate;
- tipul soluţiei obţinute.

Toate metodele de minimizare cunoscute, clasice sau moderne, metode precise sau euristice, enumerând aici: metoda analitică Quine-McCluskey, metoda grafică sau grafo-analitică a diagramei Karnaugh, metodele analitice euristice sau precise Espresso I şi II, Espresso MV, metoda SIS, sau MVSIS, indiferent dacă se referă la sisteme binare sau multivalente, cu ieşiri singulare sau multiple, au următoarele caracteristici comune:

1. sunt în esenţă binare;
2. se aplică sistemelor cu o singură ieşire binară (ieşirile multiple binare, sau singulare multivalente se reduc sau se descompun în subsisteme binare cu o singură ieşire);
3. se aplică doar ieşirilor multivalente singulare, după descompunerea sistemelor cu ieşiri multiple în subsisteme cu ieşiri singulare, care, la rândul lor sunt codificate binar, tip 1 din N (adică fiecare din cele N valori multivalente se reprezintă printr-un vector binar cu N poziţii, având doar o valoare 1, restul componentelor vectorului fiind 0), iar ieşirea multiplă rezultată se tratează ca N ieşiri singulare separate;

Metoda de minimizare	Metodă		Domeniu variabile		Reprezentare		Ieşire		Soluţie	
	Grafică	Analitică	Binar	Multivalent	Tabelară	Diagramă de decizie	Singulară	Multiplă	Exactă	Euristică
Karnaugh	x	x	x		x		x			
Quine		x	x		x		x			
MINI		x	x		x			x		
ESPRESSO		x	x			x		x	x	x
MVSIS		x	x	x		x		x	x	x
FCMIN		x	x		x			x		x
CD-COVERAGE		x	x		x		x			x
BOOM I		x	x		x			x		x
BOOM II		x	x		x			x		x
Discriminării		x	x	x	x			x	x	

Tab. 3-5 Clasificarea metodelor şi algoritmilor de minimizare

Principiul de bază al acestor metode de minimizare constă în analizarea valorilor intrării pentru care funcţia are valoarea 1 (sau, alternativ, 0), folosindu-se teoremele de bază ale algebrei booleene (dintre care menţionez adiacenţa, factorul comun, consensul, teoremele lui De Morgan etc.) şi ceea ce este specific acestor metode, optimizarea folosind valorile intrării pentru care funcţia nu este specificată.

Un asemenea principiu implică detalierea şi analizarea tuturor combinaţiilor de intrare nespecificate, operaţie care poate fi destul de laborioasă, în cazul unui mare număr de intrări, sau în cazul sistemelor multivalente.

Întrucât, de fiecare dată, analiza în vederea minimizării se realizează pe două submulţimi ale valorilor intrării:

- submulţimea *ON* , pentru care funcţia are valoarea 1 şi
- submulţimea *DC* (don't care), pentru care funcţia nu este specificată,

sau

- submulţimea *OFF* , pentru care funcţia are valoarea 0 şi
- submulţimea *DC* (don't care), pentru care funcţia nu este specificată,

rezultă o limitare a metodelor doar la optimizarea funcţiilor cu ieşiri binare singulare, urmărindu-se să se reducă celelalte cazuri la acesta, prin descompunerea lor în subsisteme cu o singură ieşire binară.

Metoda discriminării, spre deosebire de celelalte metode, foloseşte pentru minimizare submulţimile valorilor intrării, grupate pe valorile specificate ale funcţiei, neanalizându-se submulţimea valorilor intrării pentru care funcţia nu este specificată.

Ideea de bază a minimizării constă în reţinerea în fiecare vector de intrare aparţinând unei grupe valorice a ieşirii, a numărului minim de componente specificate care-l separă (îl discriminează) faţă de toţi ceilalţi vectori de intrare aparţinând altor grupe valorice ale funcţiei.

Metoda discriminării permite o gamă largă de aplicaţi care probează flexibilitatea, adaptabilitatea şi eficienţa metodei. Structurile de logică vectorială, cu particularizarea ei „logică algoritmică”, optimizabile prin discriminare, deschid perspectiva obţinerii unui salt în viteza de prelucrare a informaţiei, prin „hardificarea” software-lui, sau direct prin optimizarea structurilor decizionale ale algoritmilor.

4 Metoda discriminării

- prezentare şi propuneri de extindere -

Expun această metodă într-un capitol separat datorită avantajelor pe care le prezintă faţă de celelalte metode. Mai mult, fiindcă nu era descris explicit un algoritm, am realizat o implementare, cu scopul de a face o comparaţie între rezultatele obţinute cu aceasta şi cele obţinute cu alte metode cunoscute. Implementarea a fost realizată în colaborare cu autorul metodei şi validată de acesta.

Înainte de a începe prezentarea metodei trebuie să amintesc câteva detalii legate de colaborarea cu autorul acesteia. Scopul iniţial al colaborării a fost implementarea unei metode de minimizare şi optimizare a funcţiilor multivalente specificate tabelar care să aibă la bază metoda discriminării. Pe măsură ce am realizat implementarea, am descoperit o serie de avantaje faţă de metodele clasice, şi faptul că deschide o nouă perspectivă în domeniul scrierii de algoritmi prin interpretările ce pot fi asociate rezultatelor.

Aşa cum a fost prezentat în capitolul dedicat prezentării sistemelor logice multivalente, variabilele dintr-un sistem logic pot fi scalare sau vectoriale. Pentru a simplifica prezentarea metodei nu se va păstra gruparea componentelor variabilelor vectoriale. Ele vor fi tratate fără a face vreo diferenţiere, ca elemente scalare, alături de celelalte elemente scalare ale sistemului logic.

4.1 Descrierea generală a metodei

Metoda discriminării se bazează pe determinarea diferenţelor dintre combinaţiile de intrare ce determină ieşiri distincte ale sistemului logic.

Un sistem logic poate avea ieşire singulară sau multiplă. Pentru sistemele cu ieşire multiplă se **multiplică sistemul pentru fiecare ieşire** în parte, apoi procesarea se face separat pentru fiecare subsistem logic.

Prima etapă propriu-zisă a metodei, numită **crearea grupelor de valori**, constă în gruparea combinaţiilor de intrare specificate care determină ieşiri identice.

Pentru fiecare grupă se urmăreşte aflarea componentelor, care diferenţiază vectorii din fiecare grupă de vectorii din celelalte grupe. Se creează iniţial **listele de vectori acoperitori**, din care vor fi aleşi **vectorii discriminanţi**, sau simplu: **discriminanţii**.

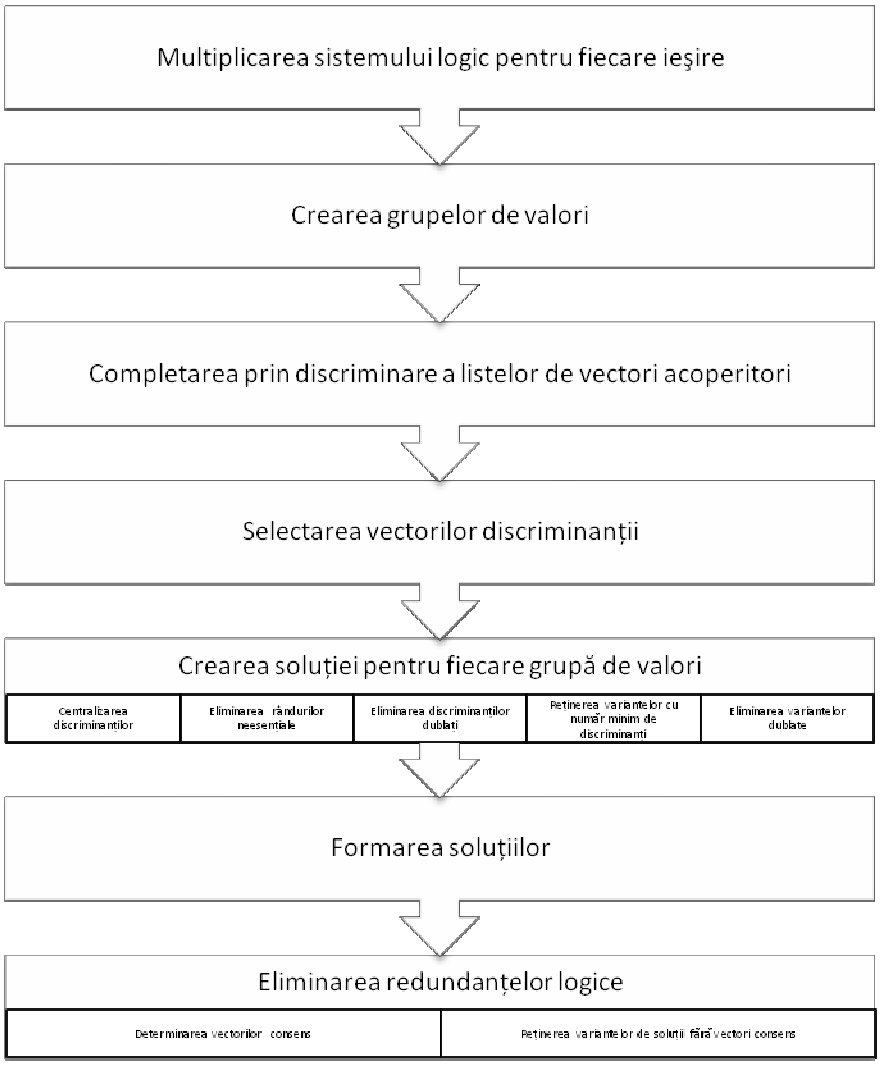


Fig. 4-1 Metoda discriminării

În continuare se face o **centralizare** pe fiecare grupă a **discriminanţilor** şi se determină **variantele de soluţii** pentru fiecare grupă în parte. În cadrul fiecărei soluţii este păstrată o **instanţă unică** pentru fiecare discriminant. În urma acestei operaţii este posibil să apară variante de soluţii cu număr diferit de discriminanţi, şi chiar mai multe instanţe ale aceleiaşi soluţii.

Următoarele etape constau în **reţinerea de variante de soluţii** cu număr minim de discriminanţi şi în **păstrarea instanţelor unice de soluţii**.

Până în această etapă a fost obţinut un pachet de **soluţii temporare** care pot conţine redundanţe. Următoarea operaţie constă în

determinarea vectorilor consens pentru fiecare soluţie. Vectorii consens din cadrul soluţiilor sunt vectori redundanţi. **Soluţiile fără redundanţe logice** se obţin prin eliminarea vectorilor consens acolo unde este posibil.

4.2 Multiplicarea sistemului logic pentru fiecare ieşire

Prima etapă din metoda discriminării constă în multiplicarea tabelului de valori pentru fiecare ieşire în parte. Un tabel de adevăr cu ieşiri multiple este echivalent cu un sistem format din mai multe funcţii (câte una pentru fiecare ieşire), urmând ca fiecare să fie minimizat separat.

Un convertor zecimal DCB / ab definit pe trei variabile $D,C,B \in Z_3 = \{0,1,2\}$ şi cu o ieşire vectorială formată din 2 variabile $a,b \in Z_5 = \{0,1,2,3,4\}$, se defineşte prin tabelul:

Nr. Crt.	Intrare			Ieşire	
	D	B	C	a	b
1	0	0	0	0	0
2	0	0	1	0	1
3	0	0	2	0	2
4	0	1	0	0	3
5	0	1	1	0	4
6	0	1	2	1	0
7	0	2	0	1	1
8	0	2	1	1	2
9	0	2	2	1	3
10	1	0	0	1	4

Tab. 4-1 Convertor „zecimal $Z_3 \rightarrow$ zecimal Z_5”

În urma multiplicării se obţin tabelele de adevăr pentru ieşirile a şi b:

Nr. Crt.	Intrare			Ieşire
	D	B	C	a
1	0	0	0	0
2	0	0	1	0
3	0	0	2	0
4	0	1	0	0
5	0	1	1	0
6	0	1	2	1
7	0	2	0	1
8	0	2	1	1
9	0	2	2	1
10	1	0	0	1

Tabel cu prima ieşire

Nr. Crt.	Intrare			Ieşire
	D	B	C	b
1	0	0	0	0
2	0	0	1	1
3	0	0	2	2
4	0	1	0	3
5	0	1	1	4
6	0	1	2	0
7	0	2	0	1
8	0	2	1	2
9	0	2	2	3
10	1	0	0	4

Tabel cu a doua ieşire

Tab. 4-2 Separarea ieşirilor convertorului „zecimal $Z_3 \rightarrow$ zecimal Z_5”

Procesul de multiplicare necesită o tratare mai atentă în cazul existenţei unor rânduri cu ieşirea DC sau a unei valori de ieşire implicite. Rândurile cu ieşirea DC sunt eliminate. Se consideră exemplul de mai jos pentru o funcţie $F_{F_1,F_2} : Z_2 \times Z_2 \to Z_3 \times Z_2$:

Nr. Crt.	Intrare		Ieşire	
	x	y	F_1	F_2
1	0	0	0	0
2	0	1	0	1
3	2	-	1	-
4	1	2	-	1
5	În rest		2	-

Tabel iniţial

Nr. Crt.	Intrare		Ieşire
	x	y	F_1
1	0	0	0
2	0	1	0
3	2	-	1
4	1	2	-
5	În rest		2

Tabel cu prima ieşire F_1

Nr. Crt.	Intrare		Ieşire
	x	y	F_2
1	0	0	0
2	0	1	1
3	2	-	-
4	1	2	1
5	În rest		-

Tabel cu a doua ieşire F_2

Tab. 4-3 Multiplicarea pentru funcţia $F_{F_1,F_2} : Z_2 \times Z_2 \to Z_3 \times Z_2$

Din tabelul pentru ieşirea F_2 se elimină fără nicio acţiune suplimentară rândurile 3 şi 5 deoarece au ieşirea **DC**. Rândul 4 din sistemul asociat funcţiei F_1 nu poate fi eliminat direct deoarece valoarea de ieşire implicită a acestuia este diferită de **DC**. Se determină mintermenii vizaţi de rândul 5 (ieşirea implicită) şi după aceea poate fi eliminat rândul 4. Rândul 5 va fi înlocuit cu $R_{5.1.}(10/2)$, $R_{5.2.}(11/2)$ şi $R_{5.3.}(02/2)$.

Nr. Crt.	Intrare		Ieşire
	x	y	F_1
1	0	0	0
2	0	1	0
3	2	-	1
5.1.	1	0	2
5.2.	1	1	2
5.3	0	2	2

Tabel cu prima ieşire

Nr. Crt.	Intrare		Ieşire
	x	y	F_2
1	0	0	0
2	0	1	1
4	1	1	1

Tabel cu a doua ieşire

Tab. 4-4 Multiplicarea tabelului de intrare

În concluzie, dacă un sistem obţinut în urma multiplicării tabelului de adevăr conţine un rând cu ieşirea **DC**, atunci acesta este eliminat. Dacă este precizată o valoare de ieşire implicită diferită de **DC**, atunci, după multiplicare trebuie expandat tabelul pentru determinarea ieşirilor nespecificate şi apoi se pot elimina rândurile cu ieşirea **DC**. Complexitatea calculului pentru operaţia de multiplicare este exponenţială, dar depinde foarte puternic de particularităţile sistemului logic, în sensul că două sisteme cu aceeaşi complexitate de specificare din punct de vedere al numărului de rânduri şi literale pot implica un volum de procesare foarte diferit.

4.3 Crearea grupelor de valori

Din acest moment prelucrarea se va face pentru fiecare tabel în parte.

Prin **grupă sau clasă de ieşire** se înţelege mulţimea rândurilor care au aceeaşi ieşire. O grupă G pentru ieşirea V_o este formată din toate rândurile care au aceeaşi ieşire V_o. Grupa asociată unei ieşiri V_o este caracterizată prin valoarea de ieşire şi combinaţiile de intrare cărora le corespunde ieşirea V_o.

Pentru a obţine grupele de ieşire ale unui tabel de adevăr se creează o listă cu ieşirile distincte ale acestuia şi fiecare rând din tabel este ataşat grupei corespunzătoare.

Aplicarea etapei asupra convertorului „zecimal $Z_3 \rightarrow$ zecimal Z_5" conduce la următoarele rezultate:

Nr. Crt.	Intrare			Ieşire
	D	*B*	*C*	*a*
Grupa cu ieşire 0				
1	0	0	0	0
2	0	0	1	0
3	0	0	2	0
4	0	1	0	0
5	0	1	1	0
Grupa cu ieşire 1				
6	0	1	2	1
7	0	2	0	1
8	0	2	1	1
9	0	2	2	1
10	1	0	0	1

Tabel cu prima ieşire

Nr. Crt.	Intrare			Ieşire
	D	*B*	*C*	*b*
Grupa cu ieşire 0				
1	0	0	0	0
6	0	1	2	0
Grupa cu ieşire 1				
2	0	0	1	1
7	0	2	0	1
Grupa cu ieşire 2				
3	0	0	2	2
8	0	2	1	2
Grupa cu ieşire 3				
4	0	1	0	3
9	0	2	2	3
Grupa cu ieşire 4				
5	0	1	1	4
10	1	0	0	4

Tabel cu a doua ieşire

Tab. 4-5 Împărţirea pe grupe de valori

Timpul de calcul maxim este $O(n \log n)$, unde n este numărul de rânduri din tabel.

4.4 Completarea prin discriminare a listelor de vectori discriminanţi

Pentru un vector se poate genera o mulţime de **vectori acoperitori** prin înlocuirea elementelor specificate singulare cu elemente de tip „**don't care**". De exemplu, vectorul $(13-2)$ poate fi acoperit de vectorii: $(13-2)$, $(-3-2)$, $(1--2)$, $(13--)$, $(1---)$, $(-3--)$, $(---2)$ sau $(----)$. În cadrul

acestei etape de prelucrare, pentru fiecare vector de intrare se creează lista C de vectori acoperitori.

Operaţia de generare a vectorilor acoperitori corespunde unei „relaxări" a variabilelor de intrare pe direcţiile restricţionate de valorile specificate în vectorul de intrare.

Aplicarea etapei asupra primei ieşiri (a) din convertorul „zecimal $Z_3 \rightarrow$ zecimal Z_5" conduce la următoarele rezultate:

Grupa (0)	Lista cu vectori acoperitori
(0 0 0)	(0 0 0) (0 0 -) (0 - 0) (0 - -) (- 0 0) (- 0 -) (- - 0) (- - -)
(0 0 1)	(0 0 1) (0 0 -) (0 - 1) (0 - -) (- 0 1) (- 0 -) (- - 1) (- - -)
(0 0 2)	(0 0 2) (0 0 -) (0 - 2) (0 - -) (- 0 2) (- 0 -) (- - 2) (- - -)
(0 1 0)	(0 1 0) (0 1 -) (0 - 0) (0 - -) (- 1 0) (- 1 -) (- - 0) (- - -)
(0 1 1)	(0 1 1) (0 1 -) (0 - 1) (0 - -) (- 1 1) (- 1 -) (- - 1) (- - -)

Grupa (1)	Lista cu vectori acoperitori
(0 1 2)	(0 1 2) (0 1 -) (0 - 2) (0 - -) (- 1 2) (- 1 -) (- - 2) (- - -)
(0 2 0)	(0 2 0) (0 2 -) (0 - 0) (0 - -) (- 2 0) (- 2 -) (- - 0) (- - -)
(0 2 1)	(0 2 1) (0 2 -) (0 - 1) (0 - -) (- 2 1) (- 2 -) (- - 1) (- - -)
(0 2 2)	(0 2 2) (0 2 -) (0 - 2) (0 - -) (- 2 2) (- 2 -) (- - 2) (- - -)
(1 0 0)	(1 0 0) (1 0 -) (1 - 0) (1 - -) (- 0 0) (- 0 -) (- - 0) (- - -)

Tab. 4-6 Listele cu vectori acoperitori pentru ieşirea a

Dacă un vector de lungime k are p valori specificate, $(0 < p \le q)$, atunci putem să înlocuim cu valoarea **DC** 0, 1, … sau p elemente din vectorul iniţial. Numărul de vectori acoperitori distincţi este egal cu $C_p^0 + C_p^1 + ... + C_p^{p-1} + C_p^p = 2^p$. Complexitatea acestei etape este $O\left(2^p\right)$. În cel mai rău caz, dacă p ia valoarea maximă $(p = k)$, atunci complexitatea este $O\left(2^k\right)$. Pentru un tabel cu n rânduri, complexitatea etapei este $O\left(n2^k\right)$. În medie, complexitatea este $O\left(n\left(2^1 + ... + 2^k\right)/k\right) = O\left(n2^{k+1}/k\right) = O\left(n2^{k+1-\log_2 k}\right) \approx O\left(n2^k\right)$.

4.5 Determinarea vectorilor discriminanţi

Vectorii acoperitori ai rândurilor de intrare determinaţi pentru fiecare grupă de valori sunt comparaţi cu vectorii rândurilor de intrare din celelalte grupe de valori. Această operaţie are ca scop determinarea disjuncţiilor dintre rândurile de intrare din grupe diferite.

Un vector acoperitor este eliminat dacă se intersectează cu un rând de intrare din altă grupă de valori deoarece nu evidenţiază nici o disjuncţie. După eliminarea vectorilor acoperitori „nerelevanţi" (care nu se diferenţiază de toate rândurile de intrare din alte grupe de valori) vor rămâne în mulţimile C doar vectorii discriminanţi.

În fiecare listă se vor păstra numai vectorii discriminanţi care au număr minim de poziţii specificate (acestor vectori le corespunde numărul minim de teste ce trebuie efectuate pentru a decide dacă o combinaţie de intrare dată este acoperită sau nu de o specificaţie de intrare V_i din tabelul iniţial al funcţiei). În ceea ce priveşte algoritmul, operaţia se traduce prin eliminarea discriminanţilor care sunt la rândul lor acoperiţi de alţi discriminanţi din aceeaşi listă (nu sunt minimali).

Listele cu vectorii discriminanţi pentru ieşirea a a convertorului „zecimal $Z_3 \rightarrow$ zecimal Z_5" sunt:

Grupa (0)	Lista cu vectori discriminanţi
(0 0 0)	(0 0 -)
(0 0 1)	(0 0 -) (- 0 1)
(0 0 2)	(0 0 -) (- 0 2)
(0 1 0)	(- 1 0)
(0 1 1)	(- 1 1)

Grupa (1)	Lista cu vectori discriminanţi
(0 1 2)	(- 1 2)
(0 2 0)	(- 2 -)
(0 2 1)	(- 2 -)
(0 2 2)	(- 2 -)
(1 0 0)	(1 - -)

Tab. 4-7 Listele cu vectorii discriminanţi pentru fiecare rând în parte

Operaţia de comparare se face element cu element. Pentru doi vectori de lungime k, aceasta are complexitatea $O(k)$. A determina dacă doi vectori se intersectează sau nu implică o procesare de complexitate medie $O(k/2)$. Complexitatea întregii etape este dată de compararea tuturor vectorilor acoperitori cu toţi vectorii de intrare din grupe de valori diferite: $O(nk) \times O\left(n2^k\right) = O\left(kn^2 2^k\right)$.

4.6 Crearea soluţiei pentru fiecare grupă de valori

În continuare trebuie prelucraţi vectorii discriminanţi rămaşi în mulţimile de vectori acoperitori din cadrul fiecărei grupe cu scopul de a determina numărul minim de discriminanţi necesari pentru reprezentarea combinaţiilor de intrare care au stat la baza formării grupei respective. De asemenea, un număr mic de discriminanţi implică un număr mic de teste necesar pentru a decide dacă un vector prezent la intrarea sistemului determină sau nu la ieşirea acestuia a valorii de ieşire asociate grupei de valori.

Crearea soluţiilor pentru grupele de valori implică o procesare independentă pentru fiecare grupă. În primul rând se reunesc toate mulţimile de vectori acoperitori rămaşi, obţinându-se lista cu vectorii discriminanţi ai grupei. Fiecare vector primeşte un nume simbolic. Vectorii

acoperitori din lista asociată fiecărui rând al grupei sunt înlocuiţi cu numele simbolice corespunzătoare. Rolul acestei operaţii este determinarea discriminanţilor care pot acoperi simultan mai multe rânduri din grupă.

Pentru ieşirea a a convertorului „zecimal $Z_3 \rightarrow$ zecimal Z_5" aplicarea operaţiilor corespunzătoare acestei etape conduce la:

Rând	Grupa iniţială		Discriminanţi	Grupa c	
Grupa (0)					
1	(0 0 0)	(0 0 -)	$D_1 = (00-)$	(0 0 0)	$\{D_1\}$
2	(0 0 1)	(0 0 -) (- 0 1)	$D_2 = (-01)$	(0 0 1)	$\{D_1, D_2\}$
3	(0 0 2)	(0 0 -) (- 0 2)	$D_3 = (-02)$	(0 0 2)	$\{D_1, D_3\}$
4	(0 1 0)	(- 1 0)	$D_4 = (-10)$	(0 1 0)	$\{D_4\}$
5	(0 1 1)	(- 1 1)	$D_5 = (-11)$	(0 1 1)	$\{D_5\}$
Grupa (1)					
1	(0 1 2)	(- 1 2)		(0 1 2)	$\{D_1\}$
2	(0 2 0)	(- 2 -)	$D_1 = (-12)$	(0 2 0)	$\{D_2\}$
3	(0 2 1)	(- 2 -)	$D_2 = (-2-)$	(0 2 1)	$\{D_2\}$
4	(0 2 2)	(- 2 -)	$D_3 = (1--)$	(0 2 2)	$\{D_2\}$
5	(1 0 0)	(1 - -)		(1 0 0)	$\{D_3\}$

Tab. 4-8 Listele cu vectorii discriminanţi ai grupei şi redistribuirea lor pe rânduri

În urma operaţiei de determinare a discriminanţilor o parte din rânduri sunt acoperite de un singur discriminant. Aceştia rămân în soluţia finală. După centralizare sunt eliminate din fiecare grupă rândurile care sunt acoperite de discriminanţi, care, la rândul lor, acoperă în mod unic alte rânduri. Pentru exemplul dat sunt eliminate rândurile 2 şi 3 din grupa 0 şi rândurile 3 şi 4 din grupa 1.

Complexitatea acestei procesări depinde de numărul de rânduri din grupă şi de numărul de vectori discriminanţi din fiecare listă asociată unei grupe. În cel mai rău caz, dacă toţi vectorii discriminanţi dintr-o listă asociată unui rând au acelaşi număr de valori specificate, numărul maxim de vectori discriminanţi este $C_k^{k/2} = \frac{k!}{k!^2} \cong \sqrt{\frac{2}{k\pi}} 2^k$ (din formula lui Sterling), unde k este numărul de variabile de intrare. Dacă sistemul este format din n rânduri, vom avea de centralizat câte $d \cong n/g \sqrt{\frac{2}{k\pi}} 2^k$ vectori discriminanţi în fiecare grupă, unde g este numărul de grupe. În total vor fi efectuate $d \log d$ procesări constând din k comparaţii, deci cu un ordin de mărime $O(kd \log d)$. Dacă c reprezintă procentul de rânduri acoperite de un singur discriminant, atunci eliminarea rândurilor care conţin discriminanţi ce acoperă în mod unic alte rânduri se face cu o complexitate proporţională cu

$c\,n/g$ (numărul de rânduri acoperite de un singur discriminant): $O\left(c\,n/g\,k\,(d-1)\right)$.

4.7 Formarea soluţiilor

În continuare, se construiesc soluţiile pentru fiecare grupă de valori, luând din fiecare listă de discriminanţi câte un reprezentant. Se formează o listă de soluţii posibile pentru fiecare grupă de valori. Fiecare soluţie reprezintă de fapt un set de teste care trebuie efectuat pentru a determina dacă un set de date de intrare corespunde sau nu grupei respective de valori.

Până acum nu a fost exclusă posibilitatea apariţiei aceluiaşi vector discriminant în două liste din cadrul aceleiaşi grupe de valori. În timpul formării soluţiilor preliminare, dacă într-o soluţie apare de mai multe ori acelaşi vector discriminant, dublurile sunt eliminate. De aici deducem că este posibil să apară soluţii cu număr diferit de vectori. Se vor păstra numai soluţiile cu lungime minimă. Printre soluţiile cu număr minim de vectori pot să apară mai multe instanţe ale aceleaşi soluţii. Acest fapt se datorează modului în care sunt selectaţi vectorii discriminanţi în soluţiile finale. Se va păstra câte o singură instanţă pentru fiecare soluţie în parte.

Pentru ieşirea a a convertorului „zecimal $Z_3 \rightarrow$ zecimal Z_5" soluţiile pentru cele 2 grupe sunt unice:

Grupa (1)	(0 0 -) (- 1 0) (- 1 1)
Grupa (2)	(- 1 2) (- 2 -) (1 - -)

Tab. 4-9 Tabel cu soluţii

4.8 Eliminarea redundanţelor din soluţii

Următoarea etapă constă în determinarea rândurilor redundante din soluţia problemei, simplificând astfel forma acesteia.

Dacă într-o grupă valorică se identifică un triplet de vectori de forma:

$$va + v'b + ab \qquad (4\text{-}1)$$

unde :

- v şi v' reprezintă, în cele două rânduri, vectori parţiali, ale căror componente specificate diferă de la un vector la celălalt, poziţie cu poziţie,
- a şi b reprezintă restul de componente specificate din vectorii va, respectiv $v'b$,

atunci vectorul ab este posibil să fie redundant.

Vectorul v' nu este complementul lui v. Este o submulţime nevidă din $\bar{v}$ (complementul lui v).

Vectorul produs ab, numit „vector consens potenţial" al vectorilor va şi $v'b$ acoperă logic tranziţiile din v în v', sau din v' în v, conservând în timpul tranziţiei între cei doi vectori valoarea iniţială şi finală (valori egale, pentru că ne referim la aceeaşi grupă valorică).

Acest vector include în mulţimea de vectori şi combinaţiile disjuncte: vab şi $v'ab$, care se pot simplifica datorită acoperirii de către primii doi vectori.

Dacă în tabel se regăsesc şi celelalte combinaţii ale vectorului consens potenţial, atunci acest vector ar reprezenta cu adevărat un vector consens şi ar putea fi eliminat din tabel, reprezentând o redundanţă logică. În caz contrar, el rămâne de drept în tabel, pentru că reprezintă combinaţii specifice distincte ale grupei respective.

În cazul unor vectori binari, vectorul ab este vector consens şi, dacă se găseşte în tabelul de valori minimizat până în această etapă, poate fi eliminat, el reprezentând o redundanţă logică.

Întrucât analiza acestor posibilităţi devine tot mai complexă, cu cât numărul de poziţii nespecificate din vectorul ab este mai mare, complexitatea crescând şi cu gradul de multivalenţă (cu numărul de valenţe), atunci se recomandă să se procedeze în felul următor:

- se generează, prin compararea perechilor de vectori de tipul primilor doi din tripletul analizat, toţi vectorii consens potenţiali;
- tabelul obţinut se păstrează pentru etapa opţională de localizare şi de identificare a hazardului;
- separat, se reţin vectorii consens potenţiali care se regăsesc şi în tabelul grupei respective;
- se discriminează această grupă specială de vectori consens potenţiali (care se regăsesc şi în tabelul grupei) în raport cu celelalte grupe valorice minimizate, urmărind să se stabilească pentru fiecare în parte, dacă se deosebeşte sau nu faţă de vectorii celorlalte grupe valorice;
- vectorii consens potenţiali care se deosebesc de toţi vectorii celorlalte grupe valorice sunt vectori consens, deci redundanţi, şi se elimină din tabelul grupei analizate;
- se procedează la fel în toate grupele;
- în cazul binar, vectorii consens potenţiali, care sunt generaţi din perechi de vectori ce diferă prin valoarea unei singure variabile şi se regăsesc şi în tabel, se elimină direct din acesta; în rest şi în cazul binar se procedează ca mai sus.

De exemplu, pentru grupele de valori din tabelul de mai jos, se pot determina mai mulţi vectori consens:

Grupa	Vectori	Vectori consens posibili	Vectori iniţiali echivalenţi
Grupa (0)	v_1 (0 0 - 0 - -)	(v_1,v_3) (* 0 1 0 - -)	
	v_2 (0 - 0 0 - -)	(v_2,v_4) (* 0 * 0 - -)	
	v_3 (0 - - 0 - 0)	(v_3,v_4) (* 0 1 0 - 0)	
	v_4 (1 0 1 - - -)		
Grupa (1)	v_5 (- 1 1 - - 1)	(v_5,v_9) (- 1 * 1 - 1)	
	v_6 (- 1 - 1 - -)	(v_5,v_{10}) (1 1 * - - 1)	
	v_7 (1 1 - - - -)	(v_7,v_8) (* 1 - 1 - -)	v_6
	v_8 (0 - - 1 - -)	(v_8,v_{10}) (* - 0 1 - -)	v_9
	v_9 (- - 0 1 - -)		
	v_{10} (1 - 0 - - -)		

Tab. 4-10 Tabel cu vectori pentru grupele de ieşiri 0 şi 1

Vectorii consens posibili pentru grupa 0 nu se regăsesc printre vectorii discriminanţi ai grupei, ceea ce înseamnă că aceasta este minimă. Vectorii (v_5,v_9) şi (v_5,v_{10}) nu aparţin grupei 1. În schimb, vectorii (v_7,v_8) şi (v_8,v_{10}) se regăsesc în aceasta. Vectorul (v_7,v_8) este echivalent cu v_6, iar vectorul (v_8,v_{10}) este echivalent cu v_9 deoarece simbolul „$*$” are aceeaşi semnificaţie cu simbolul „–”. Vectorii (v_5,v_9) şi (v_5,v_{10}) nu aparţin grupei 1. Fiind în binar şi cu o singură variabilă (marcată prin „$*$”) care-şi schimbă valoarea între vectorii v_7 şi v_8, rezultă direct, conform argumentelor de mai înainte, că vectorii v_6 şi v_9 sunt redundanţi logic şi pot fi eliminaţi din grupa 1.

Metoda discriminării nu tratează cazul redundanţelor ce nu pot fi deduse direct cu ajutorul vectorilor consens. Să presupunem că într-o grupă de valori formată din următorii vectori v_1, v_2, v_3 şi v_4 avem următoarele relaţii între aceştia:

- $v_5 = (v_1,v_2)$ şi v_5 nu aparţine grupei
- $v_4 = (v_3,v_5)$.

În acest caz vectorul v_5 nu aparţine grupei de valori, dar, chiar dacă ar aparţine, el ar fi eliminat, deoarece este redundant, fiind vector consens pentru perechea (v_1,v_2). Dacă adăugăm acest vector, atunci poate fi eliminat vectorul v_4. Se poate face următoarea deducţie:

$$
\begin{aligned}
& v_1 + v_2 + v_3 + v_4 \equiv \\
& \equiv v_1 + v_2 + (v_1, v_2) + v_3 + v_4 \equiv \\
& \equiv v_1 + v_2 + v_5 + v_3 + v_4 \equiv \\
& \equiv v_1 + v_2 + v_5 + v_3 + (v_3, v_5) \equiv \\
& \equiv v_1 + v_2 + v_5 + v_3 = v_1 + v_2 + (v_1, v_2) + v_3 \equiv \\
& \equiv v_1 + v_2 + v_3
\end{aligned}
\tag{4-2}
$$

Din acest exemplu se poate emite următoarea regulă: pentru eliminarea redundanţei se adaugă toţi vectorii consens posibili: cei formaţi din perechi de vectori iniţiali ai grupei, formaţi dintr-un vector iniţial şi un vector consens şi cei formaţi numai din vectori consens, şi după ce nu se mai poate deduce nici un vector consens sunt eliminaţi toţi vectorii consens. Această tehnică asigură eliminarea tuturor vectorilor din grupă care pot fi deduşi logic din vectorii iniţiali.

4.9 Contribuţii privind extinderea metodei discriminării

Metoda discriminării a fost prezentată de autor ca o secvenţă de procesări ale datelor de intrare. Implementarea algoritmică a metodei a impus o serie de ajustări ale metodei iniţiale şi a condus la completarea acesteia şi cu alte procesări specifice minimizării sistemelor logice.

Metoda poate fi extinsă pentru a accepta diverse interpretări pentru sistemele logice. De exemplu, nedeterminismul prezent în specificaţia iniţială poate fi un nedeterminism impus de proiectantul sistemului sau se manifestă numai pe combinaţii de intrare care nu apar atunci când sistemul este utilizat – sistemul poate fi incomplet specificat, ambiguu sau semioptimizat. În funcţie de modul în care trebuie interpretat sistemul se impun diverse tehnici de preprocesare a tabelului de intrare.

Pentru un sistem logic univoc ambiguitatea tabelului de intrare reprezintă o deficienţă a proiectării care poate fi înlăturată în mod automat prin eliminarea ambiguităţilor sau prin reconsiderarea sistemului ca fiind univoc.

În continuare sunt prezentate:

- o metodă de validare şi preprocesare a datelor iniţiale în funcţie de încadrarea sistemului în:
  - incomplet specificat;
  - ambiguu;
  - semioptimizat;
  - nedeterminist:
- o extensie a etapei de eliminare a redundanţelor;
- o metodă pentru detectarea şi tratarea a hazardului;

Soluţia obţinută în urma minimizării poate prezenta hazard. Pentru o implementare software acesta nu constituie o problemă, dar pentru o

implementare hardware trebuie luat în considerare. Am prezentat soluţii pentru evidenţierea hazardului, clasificarea acestuia în hazard logic sau funcţional, eliminarea hazardului logic şi anunţarea hazardului funcţional pentru a fi eliminat tehnologic.

4.9.1 Interpretarea tabelelor de adevăr pentru informaţii incomplet specificate şi/sau parţial optimizate

În general, un tabel de valori supus minimizării reprezintă o funcţie intrare-ieşire univocă (fără ambiguităţi). Metoda poate fi utilizată şi pentru un tabel de valori specificat neunivoc.

În analiza nedeterminismului manifestat în relaţia intrare-ieşire apar următoarele situaţii:

4. neunivocitate de speţa I (numită şi explicită):

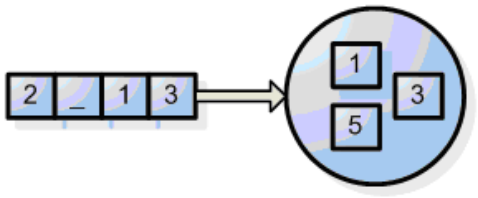


Fig. 4-2 Exemplu de neunivocitate de speţa I

Tabelul conţine cel puţin un vector de intrare căruia îi corespunde o mulţime de valori ale ieşirii. Exemplu: 2 – 1 3 / {1,3,5}

Următoarele cazuri fac parte din categoria neunivocităţilor implicite (care nu sunt evidenţiate în mod explicit în tabelul de intrare, dar se pot deduce prin analiza acestuia).

5. neunivocitate de speţa a II-a:

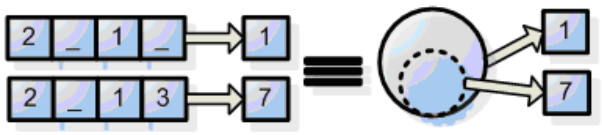


Fig. 4-3 Exemplu de neunivocitate de speţa a II-a

Mulţimea stărilor de intrare reprezentată de un rând din tabel corespunzând unei valori a funcţiei, este inclusă în mulţimea stărilor de intrare reprezentată de un rând care corespunde unei alte valori a funcţiei, de exemplu: 2 – 1 3 / 7 şi 2 – 1 - / 1

6. neunivocitate de speţa a III-a:

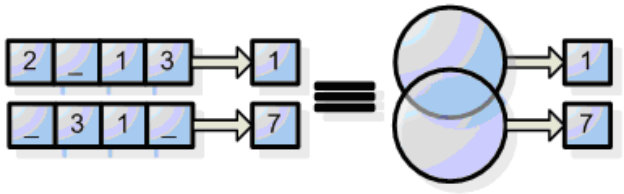


Fig. 4-4 Exemplu de neunivocitate de speţa a III-a

Mulţimile stărilor de intrare reprezentate de două rânduri din tabel aparţinând unor valori diferite ale funcţiei au intersecţia nevidă şi nu se includ una în alta, de exemplu: 2 – 1 3 / 1 şi – 3 1 – / 7.

Neunivocitatea de speţa a III-a nu reprezintă neapărat o nedeterminare în intenţia proiectantului la momentul întocmirii specificării, în schimb pune în evidenţă următoarele:

- tabelul nu are specificată ieşirea pentru orice combinaţie a intrării;
- proiectantul, cunoscând că anumite combinaţii de intrare nu sunt implicate în funcţionare a optimizat parţial specificaţia (introducând „-” în tabel). În exemplul de mai sus el este apriori asigurat că nu vor apărea combinaţii de intrare care să valideze ambele valori ale ieşirii, adică nu vor apărea combinaţii aparţinând intersecţiei submulţimilor combinaţiilor de intrare reprezentate de către cele două rânduri (în cazul exemplului, se consideră că nu va apărea combinaţia (2 3 1 3)), sau, dacă asemenea combinaţii apar, atunci efectul lor nu interesează şi nici nu afectează procesul de minimizare al sistemului.

Rezultatele minimizării, indiferent de metoda utilizată, conţin astfel de neunivocităţi pentru combinaţii care includ partea nespecificată a funcţiei iniţiale.

Situaţiile care implică simultan activarea a două sau a mai multor valori ale ieşirii într-o implementare a unui circuit combinaţional trebuie să producă pe ieşirea circuitului:

- maximul valorilor activate (în cazul primei forme canonice);
- minimul lor (în cazul celei de a doua forme canonice);
- numai una dintre valorile de ieşire activate, folosind semnale suplimentare de selecţie.

În ultimul caz, selecţia poate desemna circuitul drept unul multivaloare sau ca pe un circuit incorporat într-un automat, caz în care memoria automatului este cea care dictează care anume dintre ieşirile activate simultan să apară pe ieşirea circuitului.

Tratarea neunivocităţilor în literatura de specialitate

Problema optimizării sistemelor decizionale nedeterministe este evidenţiată şi tratată în literatura de specialitate. Trebuie menţionat însă că se tratează doar cazul neunivocităţilor de speţa I. Minimizarea se aplică

numai asupra sistemelor cu o singură ieşire multivalentă, sistemele cu ieşiri multiple fiind descompuse în prealabil în subsisteme cu o singură ieşire.

Specificările cu neunivocităţi de speţa a II-a şi mai ales de speţa a III-a, constituie o clasă aparte, cu mare probabilitate de apariţie în definirea sistemelor decizionale, şi anume, în descrierea sistemului – software sau hardware – prin tabele de valori parţial optimizate şi specificate în cadrul etapei de formulare a problemei. Acest mod de specificare se foloseşte pentru toate tipurile de sisteme decizionale (multivalente, cu variabile simple sau vectoriale etc.).

Prezenţa tabelelor de valori incomplet specificate şi parţial optimizate pune în evidenţă existenţa de sisteme decizionale hardware sau software - circuite digitale sau algoritmi – cu complexitate ridicată datorată numărului mare de combinaţii posibile ale variabilelor de intrare. Numărul acestora este dat de cardinalitatea domeniilor pe care sunt definite variabilele de intrare sau de numărul mare de variabile. Dacă spaţiul de intrare este foarte mare, este aproape imposibil ca proiectantul sistemului să clasifice toate situaţiile. Sistemele complexe sunt adesea incomplet specificate, prezentându-se doar stările implicate în aplicaţie şi, mai mult, sistemul este parţial optimizat de către proiectant. Specificarea incompletă se face ţinând cont de anumite configuraţii care nu apar într-o funcţionare normală.

Pregătirea datelor de intrare pentru tratarea neunivocităţilor

Neunivocităţile pot fi tratate într-o etapă suplimentară de preprocesare a specificaţiei iniţiale. Metoda discriminării se suplimentează cu o etapă care determină existenţa neunivocităţilor şi încadrarea acestora conform descrierilor de mai sus urmată de o modificare a tabelului de intrare.

Modificarea constă în „spargerea” vectorilor implicaţi în neunivocităţi de speţa II şi III în subvectori astfel încât să nu mai rămână decât numai neunivocităţi de speţa I.

Neunivocităţile de speţa I adăugate sistemului pot fi eliminate din tabel dacă există informaţia că acestea desemnează combinaţii de intrare care nu pot apărea în funcţionarea normală a sistemului şi au fost introduse de proiectant doar pentru a reprezenta cât mai uşor funcţia logică (semioptimizat).

Adăugarea acestei etape în metoda discriminării permite o abordare directă, în cel mai natural mod (cu o tehnică pur multivalentă) a minimizării sistemelor decizionale. Tehnicile de minimizare cunoscute sunt în esenţă binare şi se aplică tabelelor de valori cu ieşiri singulare multivalente după ce în prealabil sunt codificate în binar, după principiul 1 din N. Metoda discriminării va putea accepta spre optimizare tabele de valori prezentând oricare dintre cele trei tipuri de neunivocităţi, indiferent dacă variabilele de

intrare şi ieşire sunt binare sau multivalente, simple sau vectoriale, dacă tabelul are ieşiri singulare sau multiple.

Acestea evidenţiază disponibilitatea metodei discriminării de a soluţiona practic orice problemă de optimizare, uneori oferind singura cale practică de rezolvare a problemei.

4.9.2 Metodă algoritmică pentru determinarea hazardului

Problema hazardului (a metastabilităţii) este intens studiată [188][189][190] datorită problemelor create de instabilitatea semnalului la tranziţia dintr-o stare în alta. Metoda discriminării include o tehnică de tratare a hazardului.

Vectorii consens determinaţi în cadrul etapei de eliminare a redundanţelor pot fi utilizaţi pentru determinarea, clasificarea şi eliminarea hazardului. În continuare prezint o procesare suplimentară care trebuie adăugată după eliminarea redundanţelor cu scopul de a rezolva, dacă este posibil, sau cel puţin a determina cazurile de hazard. Etapa constă din:

4. determinarea situaţiile de hazard pentru fiecare soluţie în parte;
5. încadrarea fiecărui hazard determinat în clasa de hazard logic sau clasa de hazard funcţional;
6. eliminarea hazardului logic;
7. anunţarea hazardului funcţional pentru a putea fi eliminat tehnologic (dacă este posibil).

Determinarea situaţiilor de hazard

Determinarea situaţiilor de hazard se face pentru fiecare grupă de ieşire în parte. Se determină vectorii consens pentru fiecare pereche de vectori de intrare din soluţie. Determinarea unui vector consens este posibilă dacă cei doi vectori comparaţi sunt de forma $v_1 = x.A$ şi respectiv $v_2 = \overline{x}.B$. De exemplu, pentru două intrări cu componente binare $v_1(-\ 1\ 1\ -\ -\ 1)$ şi $v_2(1\ -\ 0\ -\ -\ -)$ vectorul consens este $(v_1,v_2)(1\ 1\ *\ -\ -\ 1)$. Pentru binar este suficient să existe doi vectori în aceeaşi grupă care să aibă pe una dintre poziţii valori specificate diferite. Apare problema determinării vectorilor consens pentru vectori cu componente multivalente. În acest caz trebuie urmărit să existe un set de vectori care să aibă pe o intrare fiecare dintre valenţele domeniului acelei intrări. De exemplu, pentru următorii vectori din $Z_2 \times Z_3 \times Z_2$: $v_1(1\ 0\ -)$, $v_2(-\ 1\ 1)$ şi $v_3(-\ 2\ 1)$ apare vectorul consens: $(v_1,v_2,v_3)(1\ *\ 1)$. Dacă nu există una dintre valori atunci nu poate fi produs vectorul consens, la fel cum pentru binar trebuie să existe ambele valori pentru a determina vectorul consens.

Totuşi, pentru situaţiile în care nu au fost găsite toate elementele necesare pentru construirea unui vector consens într-o grupă, se va

memora respectivul vector într-o listă de candidaţi deoarece există posibilitatea, aşa cum va fi prezentat mai târziu, ca acel vector consens candidat să fie utilizat la eliminarea hazardului.

Pe măsură ce sunt deduşi vectorii consens sunt ataşaţi grupei prin intermediul căreia au fost determinaţi. O dată cu determinarea vectorilor consens este reţinută şi lista de vectori de intrare pe baza cărora a fost dedus fiecare vector consens sau vector consens candidat. Cu ajutorul acestei informaţii se formează o listă cu vectorii care pot genera hazard.

Clasificarea situaţiilor de hazard determinate şi eliminarea hazardului funcţional

Până acum s-a discutat de construirea vectorilor consens. Există două liste:

1. listă cu vectori consens care au putut fi determinaţi complet dintr-o grupă;
2. listă cu vectori consens candidaţi pentru care s-au regăsit într-o grupă numai o parte din vectorii necesari declarării acestora ca vectori consens propriu-zişi.

Vectorii din prima listă anunţă, cu siguranţă, un hazard de tip logic.

A doua listă trebuie tratată astfel:

- dacă se regăseşte cel puţin un vector de intrare (care a lipsit din grupa din care au fost determinaţi vectorii consens) în altă grupă de ieşire, atunci poate fi anunţat un hazard funcţional.
- dacă nu se regăseşte nici unul din vectorii „lipsă” înseamnă că proiectantul a prevăzut că la intrare nu vor apărea niciodată combinaţiile respective, ceea ce este echivalent cu o valoare de tip „don’t care” pe ieşire. Prin urmare putem adăuga vectorii respectivi (vectorii „lipsă”) în grupa care ne interesează astfel încât să obţinem toate elementele pentru determinarea consensului, ceea ce implică transformarea acestuia din consens candidat în consens propriu-zis. În această situaţie hazardul determinat este de tip logic şi poate fi eliminat prin adăugarea consensului la grupa respectivă.

4.10 Concluzii

Pentru a obţine o reprezentare minimă a unui sistem logic metoda discriminării, faţă de alte metode, urmăreşte determinarea conjuncţiilor dintre vectorii de intrare din cadrul grupelor de valori şi a disjuncţiilor dintre vectorii de intrare aflaţi în grupe diferite. Conjuncţiile reprezintă atributele unei grupe, iar disjuncţiile reprezintă acele atribute care separă o grupă de alte grupe de valori. Din această abordare poate fi dedus modul în care se urmăreşte minimizarea în cadrul metodei: o reprezentare minimă este o reprezentare care printr-un set minim de teste determină apartenenţa unui

vector de intrare la o grupă de valori şi, în consecinţă, determină valoarea de ieşire corespunzătoare.

Abordarea nu este cea mai potrivită din punct de vedere al complexităţii de calcul (complexitatea este foarte mare – exponenţială). Am implementat această metodă pentru a compara rezultatele minimizării efectuate, cu toate metodele analizate în capitolele precedente.

Diferenţele dintre rezultatele obţinute cu aplicaţia realizată şi aplicaţiile corespunzătoare altor metode, prezentate în capitolul precedent, au demonstrat avantajele metodei discriminării din punct de vedere al complexităţii soluţiilor. Pe de altă parte, implementarea metodei a scos în evidenţă o serie de neajunsuri, cum ar fi: lipsa unor strategii de căutare adecvate, lipsa unor limite pentru restricţionarea operaţiilor de căutare, redundanţa calculului şi lipsa unei modelări sau a unui model matematic care să completeze algebra logică.

Punând în balanţă avantajele şi dezavantajele metodei discriminării (calitatea rezultatelor în raport cu neajunsurile metodei) cât şi în raport cu celelalte metode de minimizare, am considerat că este necesară crearea unei metode noi, care să aibă la bază principiul discriminării, să elimine pe cât posibil dezavantajele metodei discriminării şi să înglobeze strategii de minimizare specifice pentru diverse clase de sisteme logice (univoce, neunivoce deterministe sau nedeterministe, cu ieşiri corelate sau necorelate etc.).

Un aspect important ce trebuie remarcat pentru metoda discriminării este acela că poate accepta variabile scalare multivalente – în care o valoarea este reprezentată printr-o submulţime de valori a domeniului, diferită de mulţimea vidă. Compararea unor variabile multivalente se poate face utilizând operatorii din teoria mulţimilor. Acest tip de valori nu necesită ajustări propriu-zise ale metodei sau ale modului de aplicare al acesteia, indiferent dacă valorile se regăsesc pe intrarea sau pe ieşirea sistemului.

Metoda discriminării are următoarele avantaje:

- abordare direct multivalentă, nefiind necesare nici un fel de conversii intermediare în binar sau alte reprezentări echivalente;
- permite minimizarea sistemelor logice care utilizează variabile definite pe domenii multivalente (binare, ternare, etc.), variabilele putând avea domenii diferite;
- tratează unitar sisteme logice definite cu variabile simple şi/sau vectoriale;
- acceptă sisteme logice cu ieşiri singulare sau multiple;
- oferă o soluţionare mai directă (faţă de abordările recente cunoscute) a neunivocităţilor explicite, tratate ca valori multivalente suplimentare ale ieşirii.

Referitor la optimizarea algoritmilor, trebuie menţionat că numai într-o algebră multivalent-vectorială se pot exprima natural algoritmii, iar metoda discriminării oferă instrumentul practic al optimizării într-o astfel de algebră.

Toate aceste aspecte, care evidenţiază disponibilitatea metodei discriminării de a soluţiona practic orice problemă de minimizare mai direct,

şi uneori oferind singura cale practică, se datorează – aşa cum s-a arătat mai înainte – principiului de bază de minimizare al discriminării – principiu multivalent natural – complet diferit de principiul de bază – binar în esenţă – al tuturor metodelor cunoscute.

Utilizarea celor două etape suplimentare (pentru tratarea ambiguităţilor de proiectare a sistemelor logice univoce şi pentru tratarea hazardului) lărgeşte domeniul de aplicare al metodei discriminării atât din punct de vedere al datelor de intrare cât şi din punct de vedere al utilizării.

Acestea pot fi implementate ca opţiuni, lăsând în seama utilizatorului luarea deciziilor acolo unde acest lucru se impune. Utilizatorul poate alege eliminarea ambiguităţilor prin ştergerea rândurilor implicate sau prin reconsiderarea sistemului decizional ca fiind neunivoc. Dacă implementarea sistemului se face hardware atunci acolo unde este cerut metoda va anunţa hazardul funcţional. Mai mult, metoda poate elimina o clasă de hazard (cel logic) şi anunţa sursa hazardului funcţional (pachetele de rânduri din tabelul minimizat care induc un astfel de hazard). În sarcina utilizatorului rămâne determinarea soluţiilor tehnologice pentru eliminarea hazardul funcţional.

5 Contribuţii la extinderea specificării şi procesării sistemelor neunivoce

Prezentare şi modelare matematică

Conceptele prezentate în capitolul dedicat sistemelor logice sunt detaliate şi modelate matematic în acest capitol cu scopul de a crea un aparat matematic necesar pentru descrierea metodelor de minimizare dezvoltate în cadrul acestei lucrări [191].

La ora actuală metodele Espresso şi MVSIS, considerate ca fiind cele mai evoluate din domeniul procesării sistemelor decizionale, sunt metode care utilizează diagramele de decizie binare [192], respectiv multivalente [27][149]. Tehnica utilizată pentru generarea implicanţilor este de tip „bottom-up”. Algoritmul de bază utilizat poate fi împărţit în două faze: o fază de construcţie a diagramei de decizie şi o fază de minimizare a diagramei [193][194][195].

În cadrul metodei MVSIS este utilizată o codare a valorilor din specificare cu ajutorul listelor de valenţe. Operaţiile pentru procesarea sistemului logic sunt descrise ca secvenţe de operaţii elementare asupra diagramelor de decizie. Pentru a utiliza acest pachet este necesară construcţia diagramei de decizie, a cărei dimeniune este influenţată puternic de ordinea de intrare a variabilelor în diagramă [196]. Mai mult, problema determinării unei ordini corespunzătoare a acestor variabile este o problemă NP-complete [153]. În acest capitol dezvolt un pachet de operaţii elementare independent de structura de diagramă de decizie. Forma de reprezentare a valorilor utilizată generalizează reprezentarea din MVSIS [197][198] din punct de vedere al intrepretărilor ce pot fi date specificaţiilor logice. Reprezentarea şi pachetul de operaţii dezvoltate sunt utile şi pentru reprezentarea şi procesarea sistemelor logice neunivoce, urmând a fi utilizate în cadrul dezvoltării metodelor de minimizare prezentate în lucrare.

5.1 Motivaţie

Funcţiile pot fi specificate utilizând o relaţie matematică, cu ajutorul căreia poate fi dedusă valoarea de ieşire în funcţie de valorile de intrare, sau tabelar, sub forma unei corespondenţe între valorile de intrare şi valorile de ieşire[199][200]. Pentru a evita inconvenientele introduse de diagramele de decizie am ales să dezvolt pachetul de operaţii plecând de la specificarea iniţială, tabelară, a sistemelor logice. Tabelul de valori poate fi

interpretat ca o listă de elemente structurate, compuse din lista de valori ce corespund intrărilor şi lista de valori ce corespund ieşirilor. În tabele apar două tipuri de notaţii:

- valori specificate, folosite pentru reprezentarea unui **singur element** din domeniul de valori;
- valori de tip „orice" („don't care"), folosite pentru reprezentarea **întregului domeniu** de valori.

Primul tip de notaţie acoperă o singură valoare din domeniu, iar cel de al doilea tip de notaţie acoperă tot domeniul. Cele două notaţii nu reprezintă acelaşi tip de date. Una este de tip element, iar una este de tip mulţime. De regulă, o listă este o colecţie omogenă. Dacă privim tabelul ca o colecţie omogenă, utilizarea celor două tipuri de valori, cu semantici diferite, creează probleme în descrierea seturilor de operatori necesari procesării.

Un model mai complex de reprezentare al datelor este utilizat în MVSIS. În cadrul acestuia pot fi utilizate şi submulţimi ale domeniilor de valenţe pe lângă valorile singulare folosite în mod uzual. Utilizarea submulţimilor are dublu rol: permite specificarea corespondenţelor nedeterministe şi este utilă pentru reprezentarea compactă a datelor de intrare (de exemplu, doi vectori 0 1 2 şi 1 1 2 pot fi reprezentaţi unificat prin {0,1} 1 2).

În cazul binar, singura mulţime cu mai mult de un element este mulţimea totală. În proiectarea logică binară este utilizată o a treia valenţă X care corespunde stării de înaltă impedanţă – starea în care nu este prezentă nici o valoare logică pe fir. La o analiză mai atentă starea de înaltă impedanţă corespunde mulţimii vide în reprezentarea cu submulţimi.

Pentru eliminarea acestor neconcordanţe dezvolt un sistem de reprezentare şi operare extins asupra specificaţiilor multivalente neunivoce care să permită utilizarea atât a valorilor singulare cât şi a valorilor de tip mulţimi de valenţe. Acesta cuprinde reprezentarea datelor (codarea) şi procesările specifice asupra structurilor utilizate în reprezentare (definirea şi descrierea operatorilor).

Aparatul matematic propus în acest capitol este destinat studiului sistemelor decizionale multivalente bazat pe utilizarea variabilelor de tip multivaloare. El s-a conturat pe măsură ce am dezvoltat metodele preliminare prezentate în capitolul următor. Scopul acestui capitol este crearea unui mecanism formal ce va fi utilizat la descrierea sistemelor decizionale şi la modelarea prin ecuaţii matematice a procesărilor implicate în operaţiile specifice minimizării sistemelor decizionale multivalente:

- domenii multivalente, variabile multivalente şi operatori specifici;
- specificarea multivalentă şi reprezentarea tabelară a datelor de intrare;
- vectori de variabile multivalente şi operatori specifici;
- matrice (liste de vectori) şi operatori specifici;
- operatori neomogeni (operatori cu operanzi vectori şi matrice).

Extinderea specificărilor multivalente

Unificarea modului de reprezentare se rezolvă prin utilizarea submulţimilor cu o singură valenţă în locul valorilor singulare. Se disting cazurile particulare ale mulţimilor vide, mulţimilor formate dintr-un singur element şi mulţimilor egale cu tot domeniul de definiţie. Celelalte cazuri sunt reprezentate de submulţimi diferite de întregul domeniu.

Utilizarea acestui mod de reprezentare are următoarele avantaje:

1) permite specificări neunivoce explicite: o valoare reprezentată de o mulţime cu un element reprezintă o specificare univocă şi valorile reprezentate prin mulţuimi cu mai multe elemente sunt neunivoce;
2) reprezentare valorii de înaltă impedanţă este desemnată de mulţimea vidă, nemaifiind necesară – pentru sisteme binare - utilizarea de variabile trivalente 0,1 şi X sau 0, 1 şi 2;
3) crează un cadru în care tratarea nedeterminismul nu înseamnă numai selectarea unei singure valori din submulţimea de valori posibile. Nedeterminismul poate fi soluţionat prin selectarea unei submulţimi nevide a mulţimii de valori posibile. Există cazuri în care selectarea unei submulţimi conduce la soluţii mai bune decât atunci când este selectată o singură valoare. Aceast mod de tratare al nedeterminismului nu este implementat în metodele de minimizare actuale cu toate că el nu contravine regulilor de specificare a sistemelor decizionale.

Singurul dezavantaj rămas din punct de vedere al specificaţiilor neunivoice este că nu poate fi făcută din reprezentare diferenţa dintre o specificaţie neunivocă deterministă şi una nedeterministă. Soluţia este introducerea unei opţiuni care să stabilească modul de tratare a sistemului: determinist sau nedeterminist.

5.2 Analiza şi selectarea tehnicii de codare a domeniilor multivalente

Printr-un **domeniu multivalent** M se înţelege o mulţime de valori (numite valenţe). Un domeniul multivalent este o mulţime finită. Numărul de valenţe este o caracteristică a unui domeniu multivalent numită **cardinalitate**. Domeniul binar $M_2 = \{false, true\}$ este un domeniu multivalent cu două valenţe $false$ şi $true$. Utilizarea domeniilor multivalente ca atare în prelucrarea tabelelor de adevăr îngreunează înţelegerea metodelor.

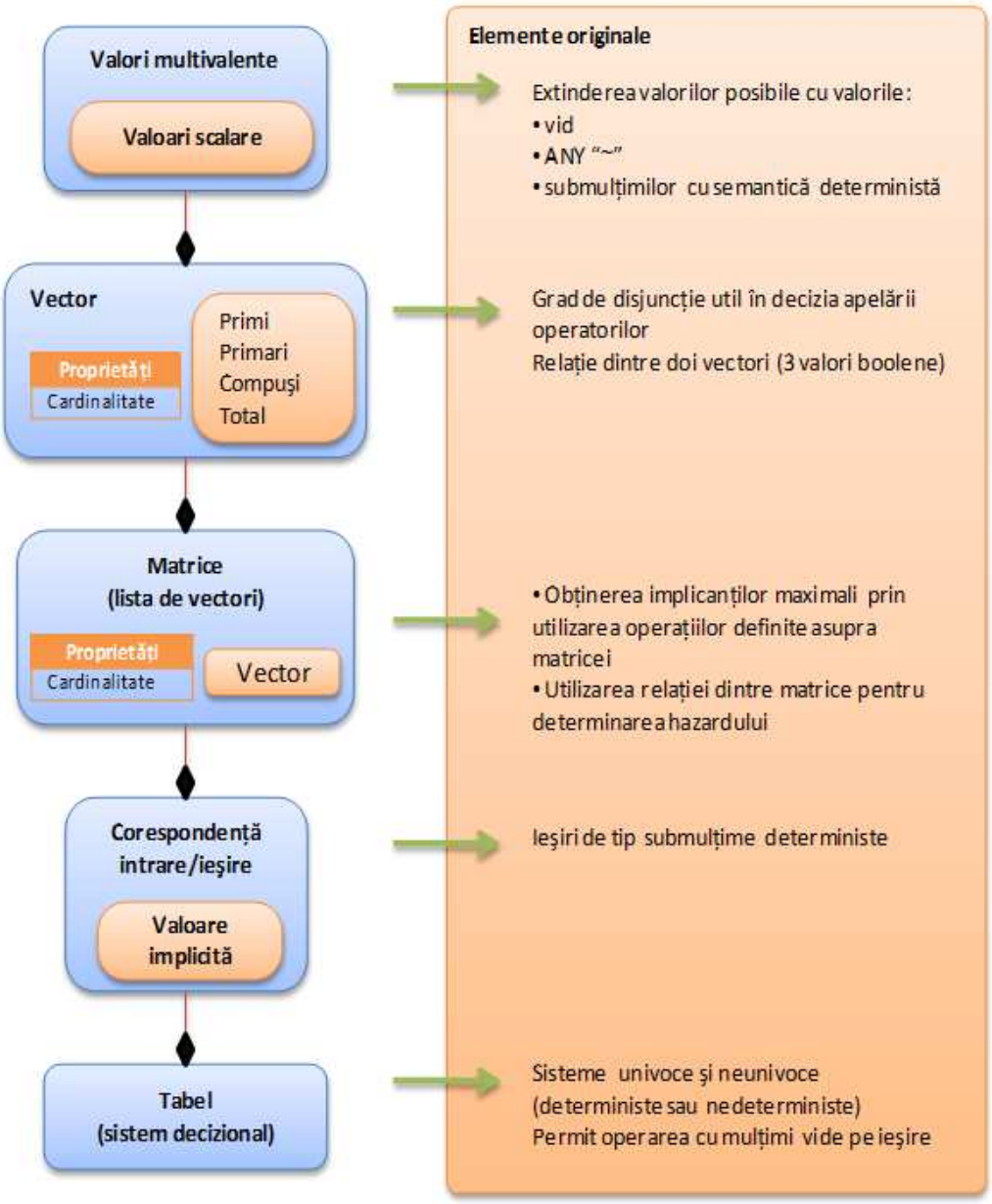


Fig. 5-1 Structuri cu valori multivalente dezvoltate pentru specificarea sistemelor logice

Am introdus notaţia Z_n pentru mulţimea formată din primele n numere naturale: $\{0,1,...,(n-1)\}$. În locul unui domeniu $M_3 = \{oprit, liber, ocupat\}$ este de preferat utilizarea domeniului $Z_3 = \{0,1,2\}$. Se construieşte o funcţie de codare bijectivă definită pe domeniul

multivalent cu valori în mulţimea Z_n, prin asocierea arbitrară a câte unei singure valori din domeniul Z_n fiecărei valenţe.

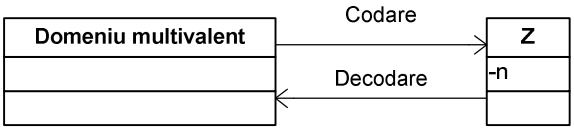


Fig. 5-2 Codarea în Z_n

Exemplul 1:

Pentru domeniul binar codarea se face în Z_2 astfel:

$$f : M_2 \to Z_2 \tag{5-1}$$

$$f(false) = 0 \tag{5-2}$$

$$f(true) = 1 \tag{5-3}$$

Exemplul 2:

Pentru un domeniu multivalent asociat stărilor unui aparat de taxat de pe un taximetru: $M_3 = \{oprit, liber, ocupat\}$ o codare în Z_3 poate fi : $f : M_3 \to Z_3$, unde $\begin{cases} f(oprit) = 0 \\ f(liber) = 1 \\ f(ocupat) = 2 \end{cases}$.

Observaţie: nu trebuie asociate elementele mulţimii $Z_n = \{0, 1, 2, ..., n-1\}$ cu secvenţa de numere $0, 1, 2, ..., n-1$. Pentru valenţele unui domeniu de valori, în cel mai general caz, nu este definită o ordine parţială sau totală. Ele sunt simboluri, elemente ale unei mulţimi, fără nici o legătură cu numerele naturale. Există şi sisteme care în cadrul codării stabilesc un element minim şi un element maxim, dar considerentele care conduc la o astfel de tratare nu sunt relevante în minimizarea sistemelor logice.

Avantajele codării în Z_n

Utilizarea unei codări binare pentru reprezentarea unui domeniu multivalent a cărui cardinalitate nu este o putere a lui 2 introduce valenţe suplimentare inexistente în problema iniţială. Pentru exemplul precedent, se poate utiliza şi o codare cu ajutorul a două variabile binare, una care să stabilească dacă aparatul de taxat este pornit sau nu şi a doua să stabilească dacă este disponibil sau nu. Corespondenţa este prezentată în tabelul de mai jos:

Nr. Crt.	Codare binară		Codare în M_3	Codare Z_3
	Pornit	Liber		
1	Nu	Nu	Oprit	0

2	Nu	Da		
3	Da	Nu	Ocupat	1
4	Da	Da	Liber	2

Tab. 5-1 Codare binară şi codare multivalentă

După cum se observă codarea binară implică patru stări - la două dintre acestea le corespunde aceeaşi valoare în Z_3.

5.3 Valori multivalente (de tip submulţime)

Utilizarea conceptului de mulţime pentru reprezentarea valorilor şi variabilelor multivalente necesită redefinirea elementelor care stau la baza specificărilor sistemelor logice. Printr-o valoare multivalentă (denumită în continuare valenţă) din Z_n se înţelege un subdomeniu al domeniului Z_n. Pentru intrările sistemului, o valoare multivalentă trebuie să aibă cel puţin un element pentru a fi validă. O valoare de intrare multivalentă trebuie să ofere cel puţin o variantă. Pentru Z_n pot fi enumerate maxim 2^n valori distincte pentru ieşire şi maxim $2^n - 1$ valori distincte pentru intrare. Astfel, valorile posibile pentru domeniul Z_3 sunt $\{\varphi,\{0\},\{1\},\{2\},\{0,1\},\{0,2\},\{1,2\},\{0,1,2\}\}$. Pentru a simplifica utilizarea acestor submulţimi în loc de $\{0\}$, de exemplu, se va utiliza 0 şi pentru valoarea care conţine toate elementele mulţimii se va utiliza „don't care" - $\{\varphi,0,1,2,\{0,1\},\{0,2\},\{1,2\},-\}$.

Valoarea are o proprietate nouă, acoperirea, legată de noţiunea de mulţime. Prin acoperirea unei valori multivalente se înţelege cardinalitatea mulţimii care reprezintă valoarea. Deoarece o valoare de intrare trebuie să aibă cel puţin un element , înseamnă că acoperirea are o valoare mai mare sau egală cu 1.

Variabilele multivalente sunt variabile care pot primi ca valori subdomenii ale unui domeniu de valori multivalent Z_n.

Utilizarea specificării propuse în reprezentarea corespondenţelor neunivoce deterministe şi nedeterministe

Interpretările valorilor de tip mulţime din specificare sunt diferite pentru intrare şi pentru ieşire. Specificarea unei valori cu cardinalitate mai mare decât 1 pe intrare semnifică faptul că pentru fiecare valenţă cuprinsă în ea trebuie să se activeze ieşirea dată în specificarea corespunzătoare. Interpretarea valorilor de ieşire se împarte în trei categorii în funcţie de cardinalitatea valorii.

1) O valoare cu cardinalitatea 0 înseamnă că sistemul nu va activa nici o valoare de ieşire (o extindere a noţiunii de înaltă impedanţă folosită la sistemele binare). În acest caz sistemul este univoc.

2) O valoare cu cardinalitatea 1 corespunde tot unei specificări univoce în care este desemnată explicit valoarea de ieşire.
3) Valorile cu cardinalitate mai mare decât 1 corespund specificărilor neunivoce. Acestea, la rândul lor se împart în deterministe şi nedeterministe.
    a. Specificările nedeterministe semnifică faptul că orice valoare nevidă, submulţime a valorii iniţiale este admisă pentru ieşirea sistemului.
    b. În cazul neunivocităţii deterministe sistemul va trebui să activeze toate valenţele valorii de ieşire.

Cazul particular al domeniului cu două valenţe Z_2

În cazul binar există doar patru submulţimi (din Z_2 se pot genera submulţimile: $S(Z_2) = \{\varphi, 0, 1, -\}$). Pentru ieşirea sistemului pot fi utilizate oricare dintre aceste submulţimi şi numai submulţimile $\{0, 1, -\}$ pentru intrare. Nu există valori formate din două valenţe în afară de mulţimea totală.

Reprezentarea în memorie a specificărilor multivalente

Descrierea mulţimilor de valenţe asociate valorilor se realizează cu ajutorul funcţiei caracteristice. Valoarea v atribuită unei variabile V definită pe domeniul Z_n este o funcţie $v : Z_n \to Z_2$ definită astfel:

- $v(x) = 0$ dacă variabila V poate lua valoarea x;
- $v(x) = 1$ dacă variabila V nu poate lua valoarea x.

Reprezentarea în memorie a valorilor se poate face în mai multe moduri:

1) prin memorarea funcţiei caracteristice – pentru fiecare valenţă din domeniu este utilizat un bit a cărui valoare true sau false indică dacă valenţa respectivă face parte sau nu din valoare;
2) prin reprezentarea mulţimii ON – lista de valenţe care fac parte din valoare. Valenţele care nu fac parte din valoare pot fi deduse prin operaţia de complementare;
3) prin reprezentarea mulţimii OFF – lista de valenţe care nu fac parte din valoare. Valenţele care fac parte din valoare pot fi deduse prin operaţia de complementare.

Reprezentarea unei valori dintr-un domeniu cu n valenţe prin funcţia caracteristică implică utilizarea a n biţi iar prin utilizarea listelor de valenţe implică în medie $ns/2$ biţi, unde s reprezintă spaţiul ocupat de o valenţă. Reprezentarea prin listă de biţi este avantajoasă din punct de vedere al memoriei utilizate decât utilizarea reprezentării prin liste de valori.

5.4 Structuri cu valori multivalente

În descrierea procesărilor sistemelor logice utilizez structuri de valori multivalente (vectori şi matrice) pentru care descriu un pachet de operaţii elementare. Procesarea propriu-zisă constă din secvenţe de operaţii elementare.

Vectori de valori multivalente

Pentru reprezentarea secvenţelor de valori multivalente am utilizat o structură de tip vector. O variabilă vectorială multivalentă V se reprezintă ca o listă de valori: $V=(v_1v_2v_3...v_c)$, fiecare valoare v_i fiind reprezentată la rândul ei prin şirul de biţi corespunzător funcţiei caracteristice $v_i=\left(v_i^0v_i^1v_i^2...v_i^{n_i-1}\right)$:

$$V=\left(\left(v_1^0v_1^1v_1^2...v_1^{n_1-1}\right)\left(v_2^0v_2^1v_2^2...v_2^{n_2-1}\right)\left(v_3^0v_3^1v_3^2...v_3^{n_3-1}\right)...\left(v_c^0v_c^1v_c^2...v_c^{n_c-1}\right)\right), \quad (5\text{-}4)$$

Fiecare valoare multivalentă reprezintă variantele pentru poziţia respectivă. Un vector de valori multivalente poate fi interpretat ca un produs de sume (SOP): „o variabilă vectorială poate lua o valoare de tip vector, compusă din c elemente, primul element fiind din Z_{n_1}, al doilea element din Z_{n_2} până la ultimul element, care poate avea o valoare din Z_{n_c}”. Valorile posibile pentru elementul din poziţia i (cu domeniul de valori Z_{n_i}) sunt date de valorile 1 conţinute de variabila multivalentă de pe poziţia i.

Semnificaţia acoperirii unui vector

Un vector, fiind o listă compusă din c submulţimi aparţinând unor domenii diferite, defineşte o relaţie între elementele a c domenii multivalente Z_{n_i}, unde $i\in\{1,2,3,..,c\}$: $V=(v_1v_2v_3...v_c)\in Z_{n_1}\times Z_{n_2}\times Z_{n_3}\times...\times Z_{n_c}$ Pentru fiecare poziţie i a unui vector sunt posibile $|v_i|$ variante. În total, numărul de variante pentru un vector este egal cu produsul numărului de variante pentru fiecare poziţie în parte.

În Fig. 5-3 este reprezentat un vector $V: Z_{n_1}\times Z_{n_2}\times Z_{n_3}$, $V=(\{0,1,3\}\ \{0,1,2,3,4,5\}\ \{0,1\})$. Numărul de vectori formaţi numai din valori cu acoperire 1, numit acoperirea vectorului V sau grad de acoperire, este $|V|=3\times6\times2=36$.

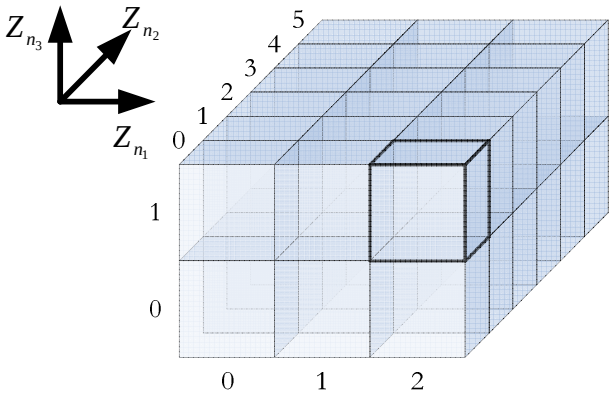


Fig. 5-3 Reprezentarea vectorului V

Acoperirea unui vector este dată de relaţia $|V| = \prod_{i=0}^{c-1} |v_i|$.

Dacă oricare dintre variabilele care compun vectorul are cardinalitatea 0, atunci acoperirea vectorului este 0. Un vector care conţine o variabilă cu cardinalitatea 0 este mulţimea vidă φ (situaţie imposibilă pe intrare).

Clasificarea vectorilor

Am introdus următoarea clasificare a vectorilor în funcţie de tipul de valori utilizat în reprezentarea acestora:

1. **vector prim** - vector format numai din variabile cu cardinalitatea 1. Acoperirea unui vector prim este egală cu produsul cardinalităţilor valorilor şi este 1.
2. **vector primar** - vector format numai din valori de cardinalitate 1 şi valori „don't care".
3. **vector compus** - vector care conţine cel puţin o valoare de cardinalitate mai mare decât 1 dar mai mică decât cardinalitatea domeniului căreia aparţine (diferită de „don't care").
4. **vector total** - vector format numai din valori „don't care". Acoperă toţi vectorii primi.

Numărul de vectori primi în care se descompune un vector reprezintă acoperirea vectorului.

În Fig. 5-4 sunt reprezentaţi grafic un vector prim V_1, un vector primar V_2, un vector compus V_3 şi vectorul total V_4 din $Z_3 \times Z_3 \times Z_2$. Vectorul prim V_1 este inclus în vectorul primar V_2 şi vectorul primar este inclus în vectorul compus V_3.

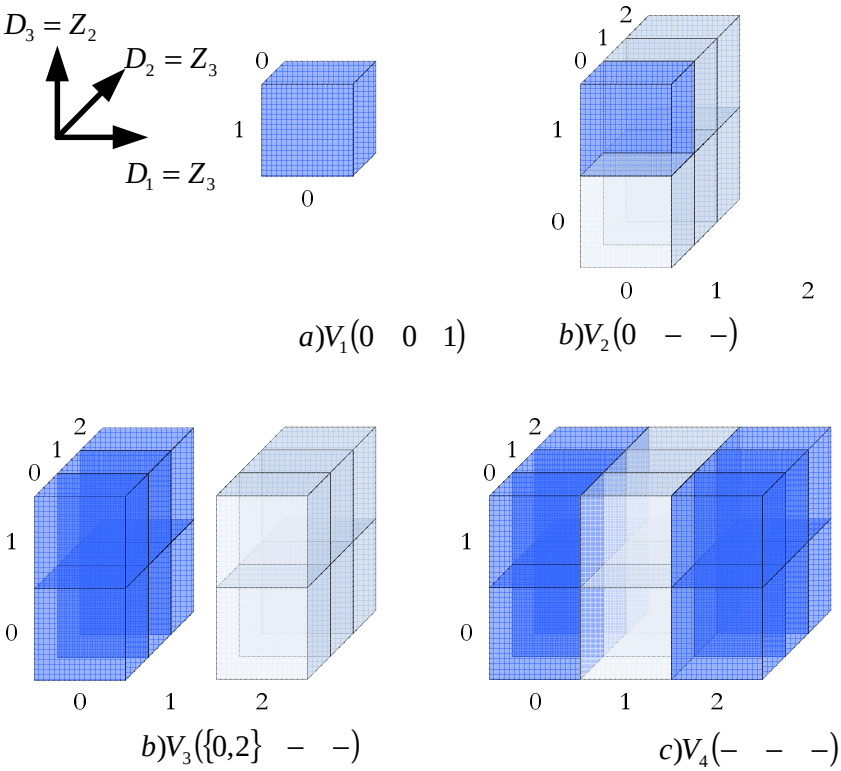


Fig. 5-4 Reprezentarea grafică a unui vector
prim (a), primar (b), compus (c) şi total (d)

Matrice de valori multivalente

Mulţimile mai puţin structurate de vectori primi (cum este cea din Fig. 5-5) nu pot fi întotdeauna reprezentate printr-un singur vector.

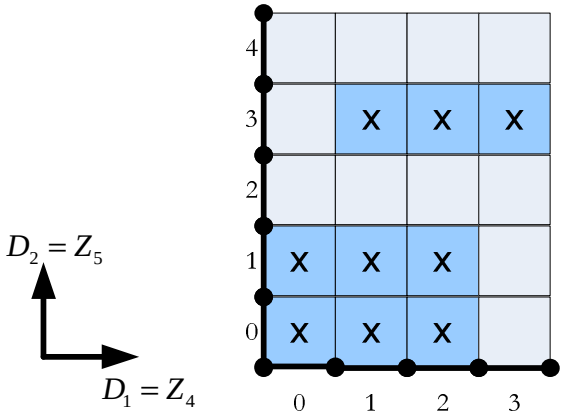


Fig. 5-5 Submulţime din $Z_4 \times Z_5$
nereprezentabilă printr-un singur vector

Mulţimea din figură poate fi reprezentată prin doi vectori $V_1 = (\{0,1,2\} \ \{0,1\})$ şi $V_2 = (\{1,2,3\} \ -)$ ($M = \{V_1, V_2\}$). Este necesară introducerea unei structuri de tip matrice pentru reprezentarea mulţimilor omogene de valori vectoriale. Toţi vectorii dintr-o matrice au acelaşi domeniu de valori $Z = Z_{n_1} \times Z_{n_2} \times ... \times Z_{n_c}$, domeniu pe care este definită şi matricea: $M \subset Z_{n_1} \times Z_{n_2} \times Z_{n_3} \times ... \times Z_{n_c}$.

Mulţimea de vectori primi a unei matrice se obţine prin reuniunea mulţimilor de vectori primi acoperiţi de vectorii care compun matricea. **Acoperirea** unei matrice este dată de numărul de vectori primi distincţi acoperiţi de vectorii matricei.

Matricea M din Fig. 5-5 are cardinalitatea 9 deoarece acoperă vectorii primari: $\{(00),(01),(10),(11),(13),(20),(21),(22),(32)\}$.

O categorie aparte a matricelor este formată din cele care conţin vectori disjuncţi. Aceste matrice au proprietatea că orice pereche de vectori care intră în componenţa ei au intersecţia vidă.

5.5 Specificarea sistemului logic cu valori multivalente

Corespondenţe şi tabele

Tabelele de valori sunt structuri care permit descrierea valorilor unei funcţii pentru toate (tabele complet specificate) sau numai o parte (tabele incomplet specificate) din combinaţiile posibile ale datelor de intrare.

Utilizarea reprezentării propuse

În cadrul reprezentării extinse a valorilor cu mulţimi se disting următoarele tipuri de valori, în funcţie de acoperire:

- 0 - valoarea vidă;
- 1 - valoarea singulară;
- între 1 şi n - valoarea multiplă;
- n - tot domeniul;

Pe intrarea unui sistem logic nu poate fi folosită valoarea vidă. Pentru ieşirea sistemului se poate utiliza orice tip de valoare.

Nr.Crt.	Tip	Cardinalitate c	Specificaţie
a)	Intrare	c=1	… 2 … / … … …
b)		1<c<n	… {1,3} … / … … …
c)		n	… - … / … … …
d)	Ieşire	0	… … … / … {} …
e)		1	… … … / … 2 …
f)		1<c<n	… … … / … {1,3} …
g)		n	… … … / … - …

Tab. 5-2 Utilizarea valorilor multivalente

Interpretarea dată unei valori cu acoperirea mai mare decât 1 pe intrare (b,c) este că se va activa valoarea de ieşire corespunzătoare pentru oricare din valenţele valorii.

Interpretarea dată unei valori de ieşire cu acoperirea 0 (d) este că nu se va activa nici o valenţă de ieşire (ieşirea se va afla în starea de „înaltă impedanţă"). Dacă ieşirea are acoperirea mai mare decât 1 (f,g) atunci interpretarea este dată de semantica specificaţiei. Dacă are semantică nedeterministă atunci orice submulţime nevidă a valorii de ieşire poate să reprezinte setul de valenţe activate pe ieşire. În cazul semanticii deterministe tot setul de valenţe al valorii trebuie activat.

5.6 Relaţionarea structurilor propuse

Operaţiile între multivalori se definesc pe baza operaţiilor elementare între biţii aflaţi pe aceleaşi poziţii din reprezentările acestora: reuniune, intersecţie, diferenţă, diferenţă simetrică, complementare:

Bit poziţie		Complement	Reuniune	Intersecţie	Diferenţă	Diferenţă simetrică
op_1	op_2	$\overline{op_1}$	$op_1 \oplus op_2$	$op_1 \otimes op_2$	$op_1 \setminus op_2$	$op_1 \Delta op_2$
0	0	1	0	0	0	0
0	1	1	1	0	0	1
1	0	0	1	0	1	1
1	1	0	1	1	0	0

Tab. 5-3 Operaţii binare care se extind direct asupra valorilor

Tabelul conţine o sinteză a operatorilor definiţi pentru structurile multivalente în care prin **V** s-au reprezentat operanzii sau rezultatele de tip vector, prim **M** operanzii sau rezultatele de tip matrice, prin **N** rezultatele de tip numeric (numere întregi nenegative), prin **L** rezultatele de tip boolean şi prin **R** rezultatele de tip relaţie (structură formată din trei valori boolene):

Categorie	Operator	Operanzi		Rezultat
Vectori	Intersecţie	V	V	V
	Diferenţă specifică	V	V	V
	Adiacenţă	V	V	N
	Grad de disjuncţie	V	V	N
	Relaţie	V	V	R
	Acoperire	V	V	L
	Complementare	V		M
	Complementare cu diferenţă specifică	V		M
	Diferenţă	V	V	M
Matrice şi vectori	Relaţie cu vector prim	M	V	R
	Reuniune	M	V	M
	Diferenţă	M	V	M

Matrice	Diferenţă	M	M	M
	Diferenţă simetrică	M	M	M
	Reuniune	M	M	M
	Intersecţie	M	M	M
	Complementare	M		M
	Complementare cu diferenţă specifică	M		M
	Relaţii	M	M	R

Tab. 5-4 Operatori definiţi pe structuri multivalente

Relaţionarea structurilor de tip vector introduse

Deoarece vectorii reprezintă mulţimi de vectori primi, operaţiile care se pot efectua asupra vectorilor sunt operaţiile asupra mulţimilor. Operatorii unari sau binari pot fi reprezentţi printr-un singur vector sau prin mai mulţi vectori.

5.6.1 Diferenţa specifică dintre doi vectori

Pentru doi vectori daţi $X=(X_1X_2X_3...X_c)$ şi $Y=(Y_1Y_2Y_3...Y_c)$ din domeniul $Z=Z_{n_1}\times Z_{n_2}\times...\times Z_{n_c}$, vom defini o structură notată $d(V_1,V_2)$ cu format de tip, vector care poate conţine valori vide, numită diferenţă specifică.

Prin diferenţă specifică a doi vectori $X=(X_1X_2X_3...X_c)$ şi $Y=(Y_1Y_2Y_3...Y_c)$ din domeniul $Z=Z_{n_1}\times Z_{n_2}\times...\times Z_{n_c}$ se înţelege structura $d(V_1,V_2)=(d_1d_2d_3...d_c)\in Z_{n_1}\times Z_{n_2}\times Z_{n_3}\times Z_{n_c}$, ale cărei elemente au valoarea $d_i=X_i\setminus Y_i$. Diferenţa specifică conţine elementele distincte ale vectorului X pe fiecare dimensiune a domeniului de definiţie.

Diferenţa specifică este utilizată pentru modelarea operaţiilor de complementare (vectori şi matrice) şi diferenţă (între vectori, vectori şi matrici, matrici). Rezultatul este o structură de date auxiliară din care urmează a fi utilizate elemente în implementarea altor operatori. Rezultatul diferenţei specifice se caracterizează prin numărul de elemente nevide şi prin numărul total de valenţe existent în aceasta. Pentru a putea reprezenta grafic diferenţa specifică valorile vide sunt înlocuite cu „don't care". Utilizarea reprezentării grafice este utilă pentru analiza rezultatului acestei operaţii.

Exemplu: Considerăm doi vectori $V_1,V_2\subset Z_3\times Z_3\times Z_2$, $V_1=(\{1,2\}\ -\ 0)$, $V_2=(1\ \ 2\ \ 1)$. Diferenţele specifice calculate pentru cei doi vectori sunt:

a) $d(V_1,V_2)=(2\ \ \{0,1\}\ \ 0)=\{(2\ -\ -),(-\ \{0,1\}\ -),(-\ -\ 0)\}$ şi

b) $d(V_2,V_1)=(\varphi\ \ \varphi\ \ 1)=\{(-\ -\ 1)\}$.

Reprezentarea grafică a celor doi vectori şi a diferenţelor specifice este redată în figura de mai jos.

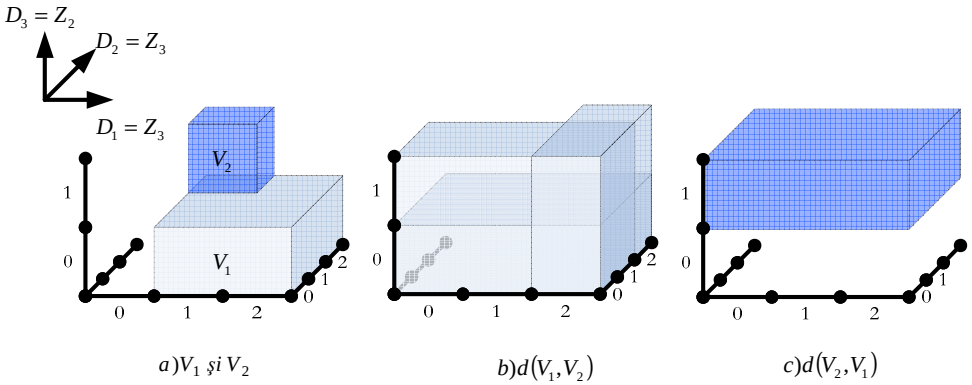


Fig. 5-6 Reprezentarea grafică a diferenţei specifice dintre vectorii V_1 şi V_2

5.6.2 Adiacenţa dintre doi vectori

Pentru doi vectori daţi $X = (X_1X_2X_3...X_c)$ şi $Y = (Y_1Y_2Y_3...Y_c)$ din domeniul $Z = Z_{n_1} \times Z_{n_2} \times ... \times Z_{n_c}$, vom defini o structură numită adiacenţă, notată $a(V_1, V_2)$ cu structură de tip vector care poate conţine valori vide. Pentru doi vectori $X = (X_1X_2X_3...X_c)$ şi $Y = (Y_1Y_2Y_3...Y_c)$ din domeniul $Z = Z_{n_1} \times Z_{n_2} \times ... \times Z_{n_c}$ adiacenţa este o structură $a(V_1, V_2) = (a_1a_2a_3...a_c) \in Z_{n_1} \times Z_{n_2} \times Z_{n_3} \times Z_{n_c}$ ale cărei elemente au valoarea $a_i = X_i \otimes Y_i$. Adiacenţa conţine elementele comune pentru fiecare domeniu Z_{n_i} care apar în produsul cartezian al domeniului de definiţie.

Rezultatul operatorului de adiacenţă este utilizat pentru modelarea rezultatelor operaţiilor dintre vectori şi matrice.

5.6.3 Gradul de disjuncţie dintre doi vectori

Numărul de valori vide din adiacenţa $a(X,Y)$ a doi vectori este numit grad de disjunţie între cei doi vectori şi este notat cu $\langle a(X,Y) \rangle$.

În funcţie de valoarea gradului de disjuncţie dintre doi vectori X şi Y se pot deduce următoarele:

- $\langle a(X,Y) \rangle = 0$ - cei doi vectori se intersectează. În acest caz $a(X,Y)$ este vector (are toate valorile diferite de mulţimea vidă) şi este egal cu $X \otimes Y$;

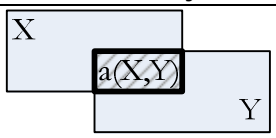


Fig. 5-7 Vectori cu grad de disjuncţie 0

- $\langle a(X,Y)\rangle = 1$ - cei doi vectori nu se intersectează pe un singur domeniu Z_i. Când gradul de disjuncţie este 1 spunem că între vectorii X şi Y există adiacenţă naturală. În acest caz $a(X,Y)$ are acoperirea 0;

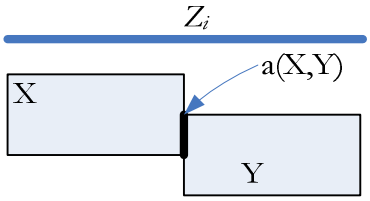


Fig. 5-8 Vectori cu adiacenţă naturală

- $\langle a(X,Y)\rangle > 1$ - cei doi vectori nu se intersectează pe domeniile Z_{n_i}. Când gradul de disjuncţie este mai mare decât 1 spunem că între vectorii X şi Y există adiacenţă într-un punct. În acest caz $a(X,Y)$ are acoperirea 0;

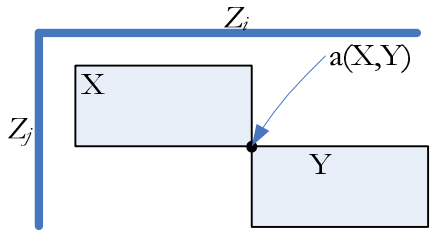


Fig. 5-9 Vectori cu adiacenţă într-un punct

Gradul de disjuncţie este utilizat ca element de decizie în modelarea procesărilor complexe asupra specificărilor logice.

Exemplu: Fie doi vectori $V_1, V_2 \subset Z_4 \times Z_5$, $V_1 = (0 \quad 1)$ şi $V_2(2 \quad 3)$. Deoarece adiacenţa celor doi vectori este $a(V_1,V_2) = (\phi \quad \phi)$ şi gradul de disjuncţie este $\langle a(V_1,V_2)\rangle = 2$ înseamnă că cei doi vectori sunt adiacenţi într-un punct. Într-adevăr, prin schimbarea ordinii elementelor mulţimilor Z_4 şi Z_5 se poate obţine o reprezentare în care vectorii sunt adiacenţi într-un punct.

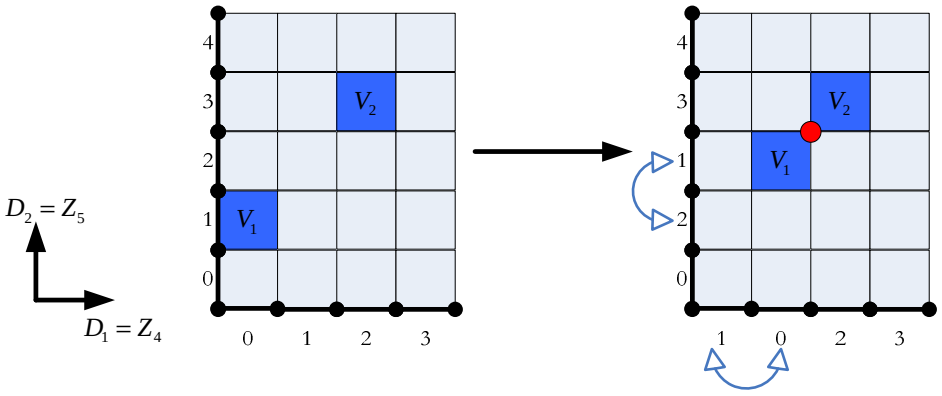


Fig. 5-10 Evidenţierea adiacenţei prin modificarea reprezentării

Ca şi la grafuri, nu există o reprezentare grafică unică a unei mulţimi, dar anumite reprezentări evidenţiază mai bine proprietăţile mulţimii sau structurilor în componenţa cărora intră mulţimea respectivă.

5.6.4 Relaţia dintre doi vectori

Doi vectori X şi Y din spaţiul $Z = Z_1 \times Z_2 \times ... \times Z_c$ pot fi analizaţi pentru a determina în ce relaţie se află. Determinarea relaţiei dintre doi vectori se poate face prin determinarea a trei mulţimi $X \otimes Y$, $X \setminus Y$ şi $Y \setminus X$. Se disting următoarele cazuri:

Caz	r_{left}	r_{middle}	r_{right}	Observaţii		
	$X \setminus Y$	$X \otimes Y$	$Y \setminus X$	X	Y	
a)	$= \phi$	$= \phi$	$= \phi$	$= \phi$	$= \phi$	Imposibil
b)	$= \phi$	$= \phi$	$\neq \phi$	$= \phi$	$\neq \phi$	Imposibil
c)	$= \phi$	$\neq \phi$	$= \phi$	$\neq \phi$	$\neq \phi$	$X = Y$
d)	$= \phi$	$\neq \phi$	$\neq \phi$	$\neq \phi$	$\neq \phi$	$X \subset Y$
e)	$\neq \phi$	$= \phi$	$= \phi$	$\neq \phi$	$= \phi$	Imposibil
f)	$\neq \phi$	$= \phi$	$\neq \phi$	$\neq \phi$	$\neq \phi$	$X \cap Y = \phi$
g)	$\neq \phi$	$\neq \phi$	$= \phi$	$\neq \phi$	$\neq \phi$	$Y \subset X$
h)	$\neq \phi$	$\neq \phi$	$\neq \phi$	$\neq \phi$	$\neq \phi$	

Tab. 5-5 Relaţia dintre doi vectori

Tabelul de mai sus evidenţiază următoarele situaţii:

a) cei doi vectori pot fi eliminaţi deoarece reprezintă vectori vizi;
b) vectorul X poate fi eliminat deoarece este vid;
c) poate fi eliminat unul dintre vectorii X sau Y deoarece reprezintă acelaşi set de valori;
d) vectorul Y poate fi eliminat deoarece este vid;
e) vectorul X poate fi eliminat deoarece este acoperit de vectorul Y;

f) vectorii X şi Y sunt dijuncţi şi diferiţi de mulţimea vidă;
g) vectorul Y poate fi eliminat deoarece este acoperit de vectorul X;

Dacă $X \cap Y = \phi$, atunci se pot determina variabilele care trebuie modificate pentru a trece din spaţiul acoperit de vectorul X în spaţiul acoperit de vectorul Y. Această operaţie este utilă pentru a evidenţia hazardul.

Definiţie 5-1: Relaţia dintre doi vectori X şi Y definiţi peste Z este o funcţie $r : Z \times Z \to \{true, false\}^3$ şi se notează $r(X,Y)$. Rezultatul este un triplet $(r_{left}, r_{middle}, r_{rigth})$ format din trei valori boolene definit astfel:

$$r_{left} = \begin{cases} true & X \setminus Y \neq \varphi \\ false & X \setminus Y = \varphi \end{cases}, \tag{5-5}$$

$$r_{middle} = \begin{cases} true & X \cap Y \neq \varphi \\ false & X \cap Y = \varphi \end{cases}, \tag{5-6}$$

$$r_{right} = \begin{cases} true & Y \setminus X \neq \varphi \\ false & Y \setminus X = \varphi \end{cases}. \tag{5-7}$$

Relaţia dintre doi vectori $X(x_1, x_2, ..., x_k)$ şi $Y(y_1, y_2, ..., y_k)$ se determină din relaţiile $r_i(x_i, y_i)$ determinate pentru fiecare poziţie $i = 1..k$ între cei doi vectori:

$$r_{left} = \bigoplus_{i=1}^{k} r_{i\,left}, \tag{5-8}$$

$$r_{middle} = \bigoplus_{i=1}^{k} r_{i\,middle}, \tag{5-9}$$

$$r_{right} = \bigoplus_{i=1}^{k} r_{i\,right}. \tag{5-10}$$

Tripletul $(r_{left}, r_{middle}, r_{rigth})$ poate fi aflat şi din adiacenţa şi diferenţele specifice dintre cei doi vectori:

$$r_{left} = \left(\sum_{i=1}^{k} \overline{\overline{d_i(X,Y)}} \right) <> 0, \tag{5-11}$$

$$r_{middle} = \overline{\overline{a(X,Y)}} <> 0, \tag{5-12}$$

$$r_{right} = \left(\sum_{i=1}^{k} \overline{\overline{d_i(Y,X)}} \right) <> 0. \tag{5-13}$$

Relaţia $r(X,Y)$ dintre doi vectori X şi Y împreună cu adiacenţa $a(X,Y)$ dintre aceştia şi diferenţele specifice $d(X,Y)$ şi $d(Y,X)$ reprezintă un pachet de operatori puternic cu ajutorul căruia pot fi definite secvenţe logico-matematice pentru implementarea unor operatori mai complecşi. Relaţia $r(X,Y)$ este utilizată pentru a verifica dacă o operaţie conduce sau nu la un rezultat (de exemplu, dacă valoarea r_{left} este falsă atunci operaţia de diferenţă $X \setminus Y$ nu are sens să fie efectuată), iar structurile $a(X,Y)$, $d(X,Y)$ şi $d(Y,X)$ conţin datele implicate în construirea rapidă a rezultatului.

5.7 Utilizarea adiacenţei şi diferenţei specifice

Utilizarea adiacenţei şi a diferenţei specifice se referă la descrierea cu ajutorul acestora a operaţiilor asupra structurilor de valori multivalente utilizate în minimizare.

Plecând de la observaţia că un vector este o matrice particulară cu un singur rând, pentru a simplifica descrierea operaţiilor se va considera că toate operaţiile asupra vectorilor şi matricelor au ca rezultat o matrice. Dacă în urma unei operaţii nu se obţine nici un vector, atunci rezultatul va fi o matrice fără nici un element. Operatorii pentru matrice sunt operatori asupra mulţimilor. Aceştia pot fi unari: $op: Z_D \to Z_D$ sau binari: $op: Z_D \times Z_D \to Z_D$, unde $Z_D = Z_{n_1} \times Z_{n_2} \times Z_{n_3} \times ... Z_{n_c}$.

5.7.1 Operaţia de diferenţă dintre doi vectori

Operaţia de diferenţă dintre doi vectori X şi Y: $X \setminus Y$ trebuie să determine vectorii primari acoperiţi de vectorul X care nu se află printre vectorii primi acoperiţi de vectorul Y.

Teoremă 5-1: Diferenţa dintre cei doi vectori poate fi exprimată prin matricea:

$$X \setminus Y = M = \begin{Bmatrix} d_1 & X_2 & X_3 & ... & X_c \\ X_1 & d_2 & X_3 & ... & X_c \\ X_1 & X_2 & d_3 & ... & X_c \\ ... & ... & ... & ... & ... \\ X_1 & X_2 & X_3 & ... & d_c \end{Bmatrix}, \qquad (5\text{-}14)$$

unde d_i reprezintă valoarea de pe poziţia i din diferenţa specifică calculată pentru cei doi vectori $d = d(X,Y)$.

Demonstraţie:

Pentru a valida această construcţie, este suficient să demonstrăm că:

1) orice vector din M nu se intersectează cu vectorul Y şi este inclus în vectorul X;
2) reuniunea dintre mulţimea $X \otimes Y$ şi matricea M este X.

Orice vector M_i din M este inclus în X deoarece toate elementele lui sunt formate din submulţimi ale elementelor lui X ($d_i \subset X_i$ unde $k = \overline{1,c}$, $k \neq i$).

Să considerăm un rând oarecare i din matricea M, astfel încât $d_i \neq \phi$. Intersecţia vectorilor $M_i = (X_1, X_2, X_3, ..., X_{i-1}, d_i, X_{i+1}, ..., X_c)$ cu vectorul $(Y_1, Y_2, Y_3, ..., Y_{i-1}, Y_i, Y_{i+1}, ..., Y_c)$ va avea pe poziţia i valoarea $d_i \otimes Y_i = (X_i \setminus Y_i) \otimes Y_i = \phi$, ceea ce înseamnă că $M_i \otimes Y = \phi$.

Astfel am demonstrat că

$$M \otimes Y = \phi, \qquad (5\text{-}15).$$

Fie un vector primar $x \in X$. Sunt posibile două situaţii:

- dacă acest vector aparţine mulţimii $X \otimes Y$, atunci aparţine şi mulţimii $M \oplus (X \otimes Y)$.
- dacă acest vector nu aparţine mulţimii $X \otimes Y$, înseamnă că pe cel puţin o poziţie i are o valoare din d_i. Vectorul $x = \left(x_1, x_2, x_3, ..., x_{i-1}, x_i^d, x_{i+1}, ..., x_c\right)$ este inclus în vectorul $M_i = (X_1, X_2, X_3, ..., X_{i-1}, d_i, X_{i+1}, ..., X_c)$ deoarece $\forall k = \overline{1,c}$, dacă $k \neq i$ atunci $x_k \in X_k$, iar dacă $k = i$ atunci $x_k^d \in d_k$ (adică $x \otimes M_i = x$).

Deci, $\forall x \in X$ atunci $x \in X \otimes Y$ sau $x \in M$ de unde rezultă că

$$X = (X \otimes Y) \oplus M, \qquad (5\text{-}16)$$

Din $M \otimes Y = \phi$ şi $X = (X \otimes Y) \oplus M$ se deduce că

$$X \setminus Y = M, \qquad (5\text{-}17)$$

Observaţie: Dacă toate elementele diferenţei specifice $d(X,Y)$ sunt mulţimi nevide, atunci matricea diferenţă are c rânduri. În caz contrar, rândurile i, $i = \overline{1,c}$ pentru care $d_i = \phi$ sunt eliminate din matrice deoarece ele nu pot forma un vector. Numărul de rânduri al matricei diferenţă este egal cu numărul de elemente diferite de mulţimea vidă al diferenţei specifice $d(X,Y)$.

Complementarea unui vector cu ajutorul diferenţei specifice

Operaţia de diferenţă este utilizată pentru a construi matricea complementară a unui vector. Aplicând construcţia de mai sus pentru a calcula diferenţa dintre vectorul total şi un vector $X=(X_1,X_2,X_3,...,X_c)$ se obţine rezultatul:

$$\overline{X}=(-,-,-,...,-)\setminus(X_1,X_2,X_3,...,X_c)=\begin{Bmatrix}\overline{X_1} & - & - & ... & - \\ - & \overline{X_2} & - & ... & - \\ - & - & \overline{X_3} & ... & - \\ ... & ... & ... & ... & ... \\ - & - & - & ... & \overline{X_c}\end{Bmatrix}. \quad (5\text{-}18)$$

Dacă o valoare X_i este DC atunci vectorul $\left(-,-,-,...,-,\overline{X_i},-,...,-\right)$ are cardinalitatea 0, deoarece $\overline{DC}=\phi$ şi este eliminat din listă. Matricea M are maxim c rânduri. Fiecare rând are o singură poziţie diferită de DC care corespunde negării valorii existente în vectorul iniţial pe poziţia respectivă. Vectorul de pe poziţia j pe coloana i va avea valoarea:

$$v_j^i=\begin{cases}- & j\neq i\\ \overline{v_j} & j=i\end{cases}, \quad (5\text{-}19)$$

5.7.2 Operaţii cu rezultat exprimat prin matrici de vectori disjuncţi

Rezultatul diferenţei $X\setminus Y$ dintre doi vectori $X(x_1x_2x_3...x_c)$ şi $Y(y_1y_2y_3...y_c)$ este matricea $X\setminus Y=M=\begin{Bmatrix}d_1 & X_2 & X_3 & ... & X_c\\ X_1 & d_2 & X_3 & ... & X_c\\ X_1 & X_2 & d_3 & ... & X_c\\ ... & ... & ... & ... & ...\\ X_1 & X_2 & X_3 & ... & d_c\end{Bmatrix}$. Considerăm doi vectori consecutivi i şi $i+1$ din această matrice: $V_i(x_1x_2x_3...d_ix_{i+1}...x_c)$ şi $V_{i+1}(x_1x_2x_3...x_id_{i+1}...x_c)$. Între valorile d_i şi x_i există relaţia $x_i=d_i\oplus(x_i\otimes y_i)$. Înlocuim vectorul $V_{i+1}(x_1x_2x_3...x_id_{i+1}...x_c)$ cu vectorii $V_{i+1}^1(x_1x_2x_3...d_id_{i+1}...x_c)$ şi $V_{i+1}^2(x_1x_2x_3...(x_i\otimes y_i)d_{i+1}...x_c)$. Vectorul V_{i+1}^2 este inclus în vectorul V_i ($V_{i+1}^2\subset V_i$) deoarece au toate componentele identice în afară de poziţia i pentru care $d_i\subset x_i$. Deci putem înlocui vectorul V_{i+1} cu vectorul V_{i+1}^2 fără a afecta acoperirea matricei diferenţă. Mai mult, vectorii V_i şi V_{i+1}^2 nu se intersectează deoarece au valori disjuncte pe poziţia i ($d_i\cap(x_i\otimes y_i)=\phi$).

Teoremă 5-2: Se consideră doi vectori X şi Y. Vectorii matricii M sunt disjuncţi şi conţin toţi vectorii primari din $X \setminus Y$.

$$M = \begin{Bmatrix} d_1 & x_2 & x_3 & \dots & x_c \\ x_1 \otimes y_1 & d_2 & X_3 & \dots & x_c \\ x_1 \otimes y_1 & x_2 \otimes y_2 & d_3 & \dots & x_c \\ \dots & \dots & \dots & \dots & \dots \\ x_1 \otimes y_1 & x_2 \otimes y_2 & x_3 \otimes y_3 & \dots & d_c \end{Bmatrix}, \qquad (5\text{-}20)$$

Demonstraţie: Pentru a demonstra că această construcţie exprimă diferenţa prin vectori disjuncţi demonstratez următoarele:

1) $M \otimes Y = \phi$;
2) $M \oplus (X \otimes Y) = X$ şi
3) pentru oricare doi indecşi distincţi i şi j de vectori din matrice este adevărată relaţia $V_i \otimes V_j = \phi$.

Pentru a demonstra punctul 1 este suficient să considerăm un vector oarecare i din matrice şi să arătăm că $V_i \otimes Y = \phi$. Vectorul V_i pe poziţia i are valoare d_i iar vectorul Y are po poziţia i are valoarea y_i. Cum $d_i = x_i \setminus y_i$ înseamnă că $d_i \otimes y_i = \phi$ de unde rezultă că $V_i \otimes Y = \phi$. Cum acestă relaţie este adevărată pentru orice vector din matricea diferenţă se deduce că $M \otimes Y = \phi$.

Pentru a demonstra punctul 2 reunim matricea M cu vectorul $X \otimes Y$ exprimat prin intersecţia celor doi vectori poziţie cu poziţie. Rezultatul este:

$$M \oplus (X \otimes Y) =$$

$$= \begin{Bmatrix} d_1 & x_2 & x_3 & \dots & x_c \\ x_1 \otimes y_1 & d_2 & x_3 & \dots & x_c \\ x_1 \otimes y_1 & x_2 \otimes y_2 & d_3 & \dots & x_c \\ \dots & \dots & \dots & \dots & \dots \\ x_1 \otimes y_1 & x_2 \otimes y_2 & x_3 \otimes y_3 & \dots & d_c \end{Bmatrix} \oplus (x_1 \otimes y_1 \quad x_2 \otimes y_2 \quad x_3 \otimes y_3 \quad \dots \quad x_c \otimes y_c) =$$

$$= \begin{Bmatrix} d_1 & x_2 & x_3 & \dots & x_c \\ x_1 \otimes y_1 & d_2 & x_3 & \dots & x_c \\ x_1 \otimes y_1 & x_2 \otimes y_2 & d_3 & \dots & x_c \\ \dots & \dots & \dots & \dots & \dots \\ x_1 \otimes y_1 & x_2 \otimes y_2 & x_3 \otimes y_3 & \dots & d_c \\ x_1 \otimes y_1 & x_2 \otimes y_2 & x_3 \otimes y_3 & \dots & x_c \otimes y_c \end{Bmatrix} = \begin{Bmatrix} d_1 & x_2 & x_3 & \dots & x_c \\ x_1 \otimes y_1 & d_2 & x_3 & \dots & x_c \\ x_1 \otimes y_1 & x_2 \otimes y_2 & d_3 & \dots & x_c \\ \dots & \dots & \dots & \dots & \dots \\ x_1 \otimes y_1 & x_2 \otimes y_2 & x_3 \otimes y_3 & \dots & x_c \end{Bmatrix} =$$

$$= \dots = \begin{Bmatrix} d_1 & x_2 & x_3 & \dots & x_c \\ x_1 \otimes y_1 & d_2 & x_3 & \dots & x_c \\ x_1 \otimes y_1 & x_2 \otimes y_2 & x_3 & \dots & x_c \end{Bmatrix} = \begin{Bmatrix} d_1 & x_2 & x_3 & \dots & x_c \\ x_1 \otimes y_1 & x_2 & x_3 & \dots & x_c \end{Bmatrix} =$$

$$= \{x_1 \quad x_2 \quad x_3 \quad \dots \quad x_c\} = X$$

La fiecare pas i am folosit relaţia $d_i \oplus (x_i \otimes y_i) = (x_i \setminus y_i) \oplus (x_i \otimes y_i) = x_i$.

Pentru a demonstra punctul 3 este suficient să considerăm valorile celor doi vectori pe poziţai i. Dacă $i < j$ acestea sunt d_i şi respectiv $x_i \otimes y_i$. Deoarece ele sunt disjuncte înseamnă că cei doi vectori (V_i şi V_j)sunt disjuncţi.

Complementarea unui vector $X(x_1x_2x_3...x_c)$ este

$$\overline{X} = (-,-,-,...,-) \setminus (X_1, X_2, X_3, ..., X_c) = \begin{Bmatrix} \overline{X_1} & - & - & ... & - \\ x_1 & \overline{X_2} & - & ... & - \\ x_1 & x_2 & \overline{X_3} & ... & - \\ ... & ... & ... & ... & ... \\ x_1 & x_2 & x_3 & ... & \overline{X_c} \end{Bmatrix}. \quad (5\text{-}21)$$

Rezultatul cu componente disjuncte al diferenţei dintre o matrice M_1 cu componente disjuncte şi un vector V se obţine prin cumularea matricelor obţinute prin scăderea vectorului V din fiecare vector al lui M.

Diferenţa $M_1 \setminus M_2$ dintre două matrice cu componente disjuncte se determină prin eliminarea din M_1 a componentelor lui M_2 utilizând diferenţa dintre o matrice şi un vector.

Adăugarea unui vector V la o matrice M cu componente disjuncte implică un calcul mai elaborat care constă în:

- se determină mulţimea de vectori V_i din M care au componente comune cu V;
- din vectorul V sunt scăzuţi succesiv vectorii V_i şi rezultatul este adăugat la matricea M.

Reuniunea a două matrice M_1 şi M_2 cu componente disjuncte face prin aplicarea algoritmului de adăugare a unui vector la matrice pentru fiecare element din M_2.

Pentru a transforma o matrice într-o matrice cu componente disjuncte se creează iniţial o matrice N formată din primul vector al matricei date la care se adaugă pe rând restul de vectori din M.

5.7.3 Operaţia de combinare a vectorilor

Considerăm doi vectori $X(x_1x_2x_3...x_c)$ şi $Y(y_1y_2y_3...y_c)$. Operaţia de combinare constă în determinarea vectorilor V formaţi din două

componente: un vector V_x inclus în X şi un vector V_y inclus în Y care au proprietatea că $V_x \setminus V_y \neq \phi$ şi $V_y \setminus V_x \neq \phi$ - $r(X,Y) = (true, -, true)$.

Din relaţia $V = V_x \oplus V_y$ rezultă că cei doi vectori V_x şi V_y au gradul de disjuncţie 0 sau 1. În caz contrar rezultatul reuniunii lor nu mai poate fi exprimat printr-un singur vector.

Relaţia dintre vectorii X şi Y se determină cu ajutorul componentelor diferenţei specifice ($d_{xy}(X,Y)$ şi $d_{yx}(Y,X)$) şi adiacenţei ($a(X,Y)$). În funcţie de gradul de disjuncţie se disting trei cazuri.

Cazul $a\langle X,Y\rangle = 0$

În această situaţie $X \otimes Y = (a_1 a_2 a_3 ... a_c) \neq \phi$. Se pot forma vectori V_x şi V_y cu proprietăţile de mai sus pentru fiecare poziţie i în care d^i_{XY} şi d^i_{YX} sunt diferite de mulţimea vidă. Cei doi vectori sunt:

$$V_x = \left(a_1 \ \ldots \ a_{i-1} \ \ a_i \oplus d^i_{XY} \ \ a_{i+1} \ \ldots \ a_c\right). \qquad (5\text{-}22)$$

$$V_Y = \left(a_1 \ \ldots \ a_{i-1} \ \ a_i \oplus d^i_{YX} \ \ a_{i+1} \ \ldots \ a_c\right). \qquad (5\text{-}23)$$

Vectorul V este:

$$\begin{aligned} V &= V_X \oplus V_Y = \\ &= \left(a_1 \ \ldots \ a_{i-1} \ \ a_i \oplus d^i_{XY} \ \ a_{i+1} \ \ldots \ a_c\right) \oplus \\ &\oplus \left(a_1 \ \ldots \ a_{i-1} \ \ a_i \oplus d^i_{YX} \ \ a_{i+1} \ \ldots \ a_c\right) = \\ &= \left(a_1 \ \ldots \ a_{i-1} \ \ a_i \oplus d^i_{XY} \oplus d^i_{YX} \ \ a_{i+1} \ \ldots \ a_c\right) = \\ &= \left(a_1 \ \ldots \ a_{i-1} \ \ x_i \oplus y_i \ \ a_{i+1} \ \ldots \ a_c\right) \end{aligned} \qquad (5\text{-}24)$$

În această situaţie numărul maxim de vectori combinaţi care se pot forma este egal cu numărul de componente al celor doi vectori.

Cazul $a\langle X,Y\rangle = 1$

Fie i poziţia în care $a_i = \phi$. În acest caz numai dacă d^i_{XY} şi d^i_{YX} sunt diferiţi de mulţimea vidă se poate forma numai perechea de vectori:

$$V_x = \left(a_1 \ \ldots \ a_{i-1} \ \ d^i_{XY} \ \ a_{i+1} \ \ldots \ a_c\right). \qquad (5\text{-}25)$$

şi

$$V_Y = \left(a_1 \ \ldots \ a_{i-1} \ \ d^i_{YX} \ \ a_{i+1} \ \ldots \ a_c\right). \qquad (5\text{-}26)$$

iar vectorul extins V este:

$$
\begin{aligned}
&V = V_X \oplus V_Y = \\
&= \left(a_1 \ \ldots \ a_{i-1} \ d^i_{XY} \ a_{i+1} \ \ldots \ a_c\right) \oplus \\
&\oplus \left(a_1 \ \ldots \ a_{i-1} \ d^i_{YX} \ a_{i+1} \ \ldots \ a_c\right) = \\
&= \left(a_1 \ \ldots \ a_{i-1} \ d^i_{XY} \oplus d^i_{YX} \ a_{i+1} \ \ldots \ a_c\right) = \\
&= \left(a_1 \ \ldots \ a_{i-1} \ x_i \oplus y_i \ a_{i+1} \ \ldots \ a_c\right)
\end{aligned}
\quad . \qquad (5\text{-}27)
$$

Cazul $a\langle X,Y\rangle > 1$

În această situaţie nu se pot forma vectori combinaţi.

Trebuie demonstrat că acest mod de construcţie a complementului unui vector acoperă toţi vectorii primi complementari şi nici un vector din matricea complementară nu se intersectează cu vectorul pentru care s-a calculat complementul.

Teoremă 5-3: Pentru un vector $V = (v_1 v_2 v_3 \ldots v_c)$ matricea

$$
M = \begin{pmatrix}
\overline{v_1} & - & - & \ldots & - \\
- & \overline{v_2} & - & \ldots & - \\
- & - & \overline{v_3} & \ldots & - \\
\ldots & \ldots & \ldots & \ldots & \ldots \\
- & - & - & \ldots & \overline{v_c}
\end{pmatrix}:
$$

a. nu are nici un vector prim comun cu V,
b. orice vector prim din $Z = Z_{n_1} \times Z_{n_2} \times \ldots \times Z_{n_c}$ aparţine vectorului V sau matricei M.

Demonstraţie:

k. Considerăm un rând oarecare i al matricei M, notat cu V_i. Intersecţia $V \otimes V_i$ se calculează astfel:

$$
\begin{aligned}
V_i &= (\ - \ - \ \ldots \ - \ \overline{v_i} \ - \ \ldots \ - \ ) \\
V &= (\ v_1 \ v_2 \ \ldots \ v_{i-1} \ v_i \ v_{i+1} \ \ldots \ v_c \ ), \\
V \otimes V_i &= (\ v_1 \ v_2 \ \ldots \ v_{i-1} \ v_i \otimes \overline{v_i} = \phi \ v_{i+1} \ \ldots \ v_c \ )
\end{aligned}
\qquad (5\text{-}28)
$$

Cardinalitatea lui $V \otimes V_i$ este $|V \otimes V_i| = \left(\prod_{k=1}^{i-1} |v_k|\right) \cdot 0 \cdot \left(\prod_{k=i+1}^{c} |v_k|\right) = 0$. Deci $V \otimes V_0 = \phi$.

În concluzie, V nu se intersectează cu nici un vector din matricea complementară.

l. A demonstra că orice vector prim din $Z = Z_{n_1} \times Z_{n_2} \times \ldots \times Z_{n_c}$ este inclus în V sau în M este echivalent cu a demonstra că V reunit cu M este vectorul total peste Z.

Considerând rândul V_c şi matricea M, prin reuniune se obţine:

$$\begin{array}{rcl} V_c & = & (\ v_1\ v_2\ v_3\ \ldots\ v_{c-1}\ \overline{v_c}\ ) \\ V & = & (\ v_1\ v_2\ v_3\ \ldots\ v_{c-1}\ v_c\ ), \\ V_c \oplus V & = & (\ v_1\ v_2\ v_3\ \ldots\ v_{c-1}\ -\ ) \end{array} \qquad (5\text{-}29)$$

Dacă vom reuni vectorul $(V_c \oplus V)$ cu vectorul V_{c-1} (penultimul rând) obţinem:

$$\begin{array}{rcl} V_{c-1} & = & (\ v_1\ v_2\ v_3\ \ldots\ v_{c-2}\ \overline{v_{c-1}}\ -\ ) \\ S_c = (V_c \oplus V) & = & (\ v_1\ v_2\ v_3\ \ldots\ v_{c-2}\ v_{c-1}\ -\ ), \\ S_{c-1} = (V_{c-1} \oplus S_c) & = & (\ v_1\ v_2\ v_3\ \ldots\ v_{c-2}\ -\ -\ ) \end{array} \qquad (5\text{-}30)$$

Să demonstrăm că $S_k = \left(\underset{i=k}{\overset{c}{\oplus}} V_i \right) \oplus V = (v_1 v_2 \ldots v_{k-2} v_{k-1} - \ldots -)$.

Din $S_{k-1} = \left(\underset{0=i=k}{\overset{c-1}{\oplus}} V_i \right) \oplus V = (v_0 v_1 \ldots v_{k-3} v_{k-2} - - \ldots -)$ obţinem:

$$\begin{array}{rcl} S_k \left(\underset{i=k}{\overset{c}{\oplus}} V_i \right) \oplus V & = & (\ v_1\ v_2\ \ldots\ v_{k-2}\ v_{k-1}\ -\ \ldots\ -\ ) \\ V_{k-1} & = & (\ v_1\ v_2\ \ldots\ v_{k-2}\ \overline{v_{k-1}}\ -\ \ldots\ -\ ). \\ S_{k-1} = \left(\underset{i=k-1}{\overset{c}{\oplus}} V_i \right) \oplus V & = & (\ v_1\ v_2\ \ldots\ v_{k-2}\ -\ -\ \ldots\ -\ ) \end{array} \qquad (5\text{-}31)$$

Deci $S_1 = (-----) = Z_1 \times Z_2 \times Z_3 \times \ldots \times Z_c = Z$.

Putem afirma că $V \otimes M = \varphi$ şi $V \oplus M = Z$, de unde rezultă că $M = \overline{V}$.

Operaţia de negare a unui vector V are ca rezultat matricea M, obţinută prin construcţie, din care au fost eliminate rândurile vide (vectorii vizi). Vectorii vizi se obţin pentru variabilele DC din V, deoarece $\overline{DC} = \varphi$.

5.7.4 Operaţia de diferenţă dintre o matrice şi un vector

Pentru a calcula diferenţa dintre o matrice M şi un vector V se construieşte o matrice D astfel: se calculează diferenţa dintre fiecare vector din matricea M şi vectorul V, iar rezultatul se reuneşte, vector cu vector cu matricea D.

Exemplu:

Fie $M \in Z_3 \times Z_3$, $V \in Z_3 \times Z_3$ unde $M = \begin{Bmatrix} \{0,2\} & - \\ - & \{0,2\} \end{Bmatrix}$ şi $V = (1,-)$.

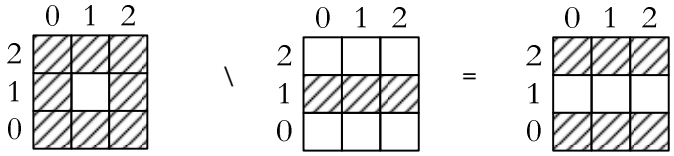
Iniţial $D = \{\ \} \in Z_3 \times Z_3$.

Prima dată se calculează $M_1 \setminus V = (\{0,2\},-) \setminus (1,-) = (\{0,2\},-)$ şi rezultatul se reuneşte cu D: $D = \{\{0,2\}\ -\}$. În continuare se calculează $M_1 \setminus V = (-,\{0,2\}) \setminus (1,-) = (\{0,2\},\{0,2\})$. Când se reuneşte vectorul

$(\{0,2\},\{0,2\})$ cu $D=\{\{0,2\}\ -\}$ se constată că $(\{0,2\},\{0,2\})<(\{0,2\},-)$, ceea ce va lăsa matricea D neschimbată.

În tabelul următor este reprezentată grafic această operaţie.

$$M=\begin{Bmatrix}\{0,2\} & - \\ - & \{0,2\}\end{Bmatrix} \quad \setminus \quad V=(1,-) \quad = \quad D=\{\{0,2\}\ -\}$$



Tab. 5-6 Reprezentarea grafică a operaţiei de diferenţă

5.7.5 Operaţia de diferenţă dintre două matrice

Definiţie 5-2: Diferenţa dintre două matrice M şi N din $Z=Z_{n_1}\times Z_{n_2}\times ...Z_{n_c}$ este o operaţie $\setminus : Z\times Z\to Z$, definită astfel: $M\setminus N=\{V\,|V\subset M\,si\,V\not\subset N\}$.

Rezultatul operaţiei de diferenţă dintre două matrice se obţine prin combinarea operaţiilor de complementare şi intersecţie: $M\setminus N=M\otimes\overline{N}$.

Diferenţa dintre două matrice se mai poate calcula utilizând operaţia de diferenţă dintre o matrice şi un vector. Rezultatul se obţine scăzând, pe rând, toţi vectorii matricei N din matricea M.

5.7.6 Complementarea unei matrice

Pentru a complementa o matrice putem utiliza operaţiile de complementare şi intersecţie ale vectorilor.

Teoremă 5-4: Complementarea unei matrice $M=\{M_1,M_2,M_3,...,M_r\}$ din $Z=Z_{n_1}\times Z_{n_2}\times ...\times Z_{n_c}$ este o operaţie $\bar{}: Z\to Z$, al cărei rezultat $\overline{M}$ este obţinut prin intersecţia matricelor complementare pentru vectorii $M_1,M_2,...,M_r$: $\overline{M}=\overset{r}{\underset{i=1}{\otimes}}\overline{M_i}$.

Demonstaţie: Pentru început voi demonstra că dacă un vector prim V aparţine matricei M, atunci $V\notin\overline{M}$. Dacă $V\in M$ înseamnă că există cel puţin un index $i\in\{1,2,...r\}$, şi numai unul (din construcţia matricei M), astfel încât $V\in M_i$. Deci $V\notin\overline{M_i}$, iar $\overline{M}\subset\overline{M_i}$, ceea ce implică $V\notin\overline{M}$.

Demonstrez că M reunit cu $\overline{M}$ este vectorul total Z. Dacă un vector prim din Z aparţine lui M, atunci el aparţine si lui $M\oplus\overline{M}$. Dacă un vector

prim V nu aparţine lui M, atunci $\forall i, i=\{1,2,...,r\}, V \notin M_i$, ceea ce înseamnă că $\forall i, i=\{1,2,...,r\}, V_p \in \overline{M_i}$. Deci $V \in \overset{r}{\underset{i=1}{\otimes}} \overline{M_i}$, adică $V \in \overline{M}$, de unde deducem că $V \in M \oplus \overline{M}$. Deci $Z \subset M \oplus \overline{M}$. Cum $M \oplus \overline{M} \subset Z$, înseamnă că $Z = M \oplus \overline{M}$, deci operaţia de complementare este bine definită.

5.7.7 Complementarea unei matrice cu ajutorul diferenţei specifice

Complementarea unei matrice se poate obţine şi cu ajutorul diferenţei specifice într-un mod asemănător cu cel al construcţiei matricei complement pentru un vector cu ajutorul diferenţei specifice.

Se porneşte de la matricea formată numai din vectorul total. Din această matrice se scad pe rând vectorii matricei pentru care se doreşte construirea complementului.

Exemplu:

Fie $M \in Z_3 \times Z_3$, unde $M = \begin{Bmatrix} \{0,2\} & - \\ - & \{0,2\} \end{Bmatrix}$.

Pentru a calcula $\overline{M}$ se construieşte o matrice iniţială $D_0 = \{-\ \ -\}$.

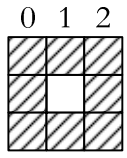
Se calculează $D_1 = D_0 \setminus M_1 = \{-\ \ -\} \setminus (\{0,2\}, -) = \{1\ \ -\}$.

Se calculează $D_2 = D_1 \setminus M_2 = \{1\ \ -\} \setminus (-, \{0,2\}) = \{1\ \ 1\}$.

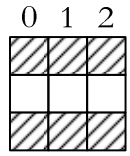
În final ultima matrice D este atribuită matricei $\overline{M}$. În acest caz $\overline{M} = D_2$.

În tabelul următor sunt reprezentaţi grafic operanzii şi rezultatele intermediare:

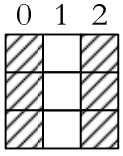
M $\quad \begin{Bmatrix} \{0,2\} & - \\ - & \{0,2\} \end{Bmatrix}$

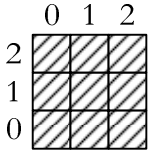
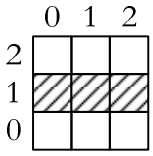
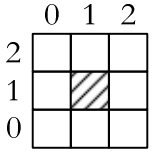


M_1 $\quad (\{0,2\}, -)$



M_2 $\quad (-, \{0,2\})$



D_0	$\{-\ -\}$	
$D_1 = D_0 \setminus M_1$	$\{1\ -\}$	
$\overline{M} = D_1 \setminus M_2$	$\{1\ 1\}$	

Tab. 5-7 Reprezentarea grafică a operaţiei de complementare

5.8 Concluzii

În cadrul acestui capitol am descris un **set de structuri de date** necesare pentru specificarea sistemelor decizionale, structuri utilizate în capitolele următoare dedicate prezentării metodelor de minimizare propuse, şi un **pachet de operatori** asupra acestor structuri, operatori care au stat la baza algoritmilor dezvoltaţi pentru implementarea metodelor.

Am arătat că cea mai bună formă de reprezentare a valorilor multivalente o reprezintă o **codare în** Z_n, submulţimea primelor n numere naturale nenegative. Utilizarea unei codări în altă bază, chiar dacă facilitează utilizarea mecanismului matematic pus la dispoziţie de sistemul de calcul (binar) implică utilizarea unei cantităţi foarte mari de memorie.

Pentru a surprinde cât mai corect situaţiile ce apar în universul sistemelor decizionale am utilizat ca valoare în locul unei submulţimi de cardinalitete 1 a domeniului de definiţie al unei valori **orice submulţime a domeniului de definiţie**, inclusiv submulţimea vidă şi submulţimea identică cu tot domeniul. Atribuirea de submulţimi de valenţe în locul valorilor singulare impune o informaţie suplimentară de interpretare a valorii unei variabile, de tip determinist sau nedeterminist.

Modul de atribuire al valorilor a impus decsrierea operatorilor binari clasici cu ajutorul operatorilor asupra mulţimilor. Dintre operatorii definiţi pe mulţimi am selectat operaţiile de diferenţă specifică şi complementare ca operaţii de bază pentru descrierea tuturor metodelor descrise în lucrare. Am adăugat ca operator de bază relaţia dintre două mulţimi al cărei rezultat exprimat prin trei valori boolene indică rezultatul comparaţiei cu mulţimea vidă a operaţiilor de diferenţă, intersecţie şi reuniune dintre cele două submulţimi. Cu ajutorul celor trei operatori am construit operaţii din ce în ce mai complexe asupra structurilor utilizate în metodele dezvoltate.

Modelarea matematică prezentată în acest capitol se caracterizează prin:

1. utilizarea submulţimilor domeniilor multivalente ca valori pentru variabilele multivalente în locul valenţelor domeniului. Atribuirea unui set de valenţe în locul uneia singure omogenizează relaţiile din specificaţiile multivalente şi dintre reprezentările pentru specificările deterministe şi nedeterministe.
2. interpretarea submulţimilor de valenţe ce pot fi atribuite unei variabile multivalente ca o funcţie corespunde naturii intrinseci binare a circuitelor combinaţionale construite pornind de la forma disjunctivă (suma de produse) pe mai multe nivele [201][202][203]:
   - un nivel decizional, format din operatori de echivalenţă, care decide dacă o valoare de intrare conţine sau nu o anumită valenţă din domeniuol de definiţie;
   - un nivel de porţi „şi” pentru fiecare rând al specificaţiei;
   - un nivel de porţi „sau” pentru fiecare grupă de ieşire a specificaţiei;
   - opţional, un nivel de selectare al valenţei de ieşire, dacă au fost activate simultan mai multe dintre acestea;
   - nivelul de livrare al valenţelor de ieşire.
3. adiacenţa dintre doi vectori şi gradul de disjuncţie (componentă a diferenţa specifică) stau la baza determinării vectorilor consens;
4. relaţia dintre doi vectori este un indicator cu ajutorul căruia pot fi determinate redundanţele şi situaţiile de hazard.

6 Contribuţii privind minimizarea funcţiilor multivalente deterministe

Prezentul capitol face legătura între capitolul care prezintă sistemele logice multivalente şi modelarea matematică a sistemelor decizionale prezentată în capitolul anterior. Pentru minimizarea funcţiilor multivalente am dezvoltat şi testat mai multe metode noi, bazate pe modelarea matematică prezentată, rezumate în trei metode distincte din punct de vedere al abordării problemei de minimizare[234][241]. În esenţă am urmărit descoperirea unei metode de procesare în care ordinea variabilelor să nu fie relevantă sau o metodă de descompunere a funcţiei avantajoasă din punct de vedere al complexităţii calculului pentru determinarea implicanţilor [204][205][206][207][208][209][210].

6.1 Metode propuse

Determinarea implicanţilor maximali se face prin:

Metoda 1: combinarea progresivă a vectorilor iniţiali. Sistemele decizionale exprimate prin vectori cu grad de acoperire mic sunt transformate prin combinare progresivă în sisteme exprimate prin vectori cu acoperire din ce în ce mai mare.

Metoda 2: utilizarea operaţiei de complementare a matricelor deoarece garantează obţinerea de componente maximale.

Metoda 3: descompunerea nedisjunctă a vectorului total până când fiecare din vectorii obţinuţi acoperă sau intersectează vectori iniţiali cu aceeaşi valoare de ieşire.

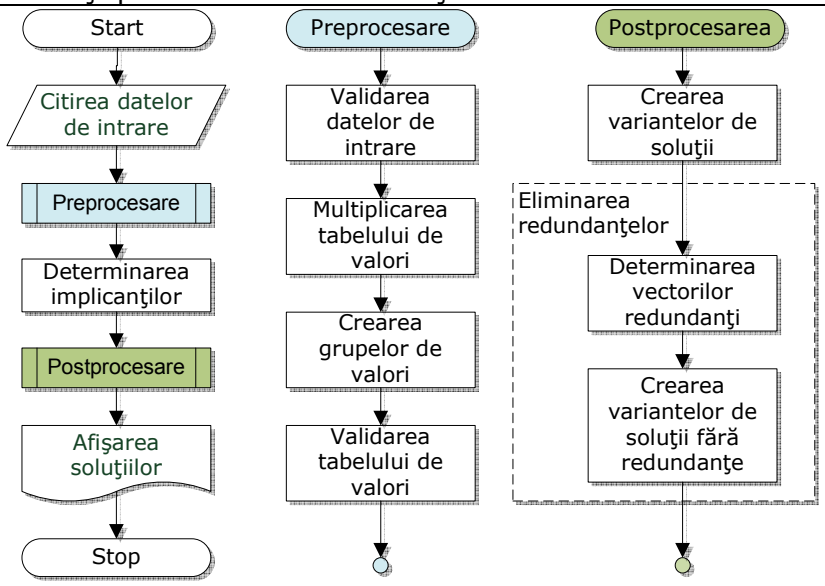


Fig. 6-1 Structura aplicaţiilor de minimizare implementate

Cele trei metode au în comun partea de preprocesare şi postprocesare (Fig. 6-1). Preprocesarea constă în validarea şi pregătirea datelor în vederea extragerii implicanţilor, iar postprocesarea are ca scop determinarea soluţiilor fără redundanţe.

6.2 Determinarea implicanţilor prin combinare (COMB)

Metoda de determinare a implicanţilor prin combinare cuprinde următoarele etape (notate cu **COMB n**):

COMB 1. Descompunerea tabelului de specificare în grupe de rânduri cu aceeaşi valoare de ieşire;

COMB 2. Împărţirea vectorilor de intrare din fiecare grupă în clici utilizând gradul de disjuncţie ca element de decizie. Doi vectori care au gradul de disjuncţie 0 sau 1 fac parte din aceeaşi clică;

COMB 3. Combinarea vectorilor din cadrul aceleiaşi clici pentru a determina implicanţii maximali;

COMB 4. Determinarea unui set minim de vectori maximali pentru fiecare clică.

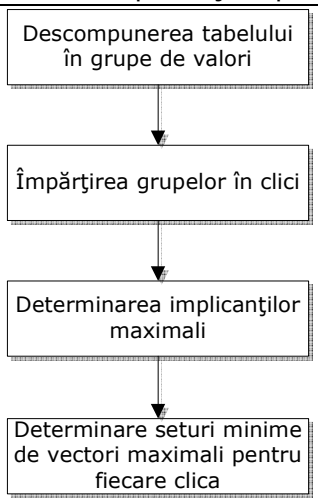


Fig. 6-2 Determinarea vectorilor maximali prin combinare

Descompunerea specificării iniţiale în grupe constă în determinarea vectorilor de intrare cărora le corespunde aceeaşi valoare de ieşire. Aceşti vectori formează o grupă de valori descrisă printr-o matrice cu vectorii de intrare şi un vector cu valorile de ieşire corespunzătoare fiecărei funcţii a sistemului decizional.

Împărţirea grupelor în clici constă în determinarea perechilor de vectori de intrare care pot forma vectori combinaţi (au gradul de disjuncţie mai mic sau egal cu 1).

În Fig. 6-3 matricea $M(V_1,V_2,V_3,V_4,V_5,V_6)$ este împărţită în clici în urma analizei gradului de disjuncţie al vectorilor componenţi. Din $a\langle V_1,V_2\rangle = 0$ şi $a\langle V_2,V_3\rangle = 1$ rezultă clica $M_1(V_1,V_2,V_3)$ şi din $a\langle V_4,V_5\rangle = 0$ şi $a\langle V_5,V_6\rangle = 1$ rezultă $M_2(V_3,V_4,V_5)$.

Operaţia de combinare a vectorilor dintr-o matrice are ca rezultat crearea de vectori noi formaţi numai din vectorii primari din matricea iniţială. În urma aplicării algoritmului de combinare (Fig. 6-4 şi Fig. 6-5) vor rezulta vectorii (V_7,V_8) pentru M_1 şi (V_9,V_{10},V_{11}) pentru M_2.

Am îmbunătăţit convergenţa metodei prin adăugarea unei secvenţe de comparare a vectorilor combinaţi noi cu vectorii iniţiali ai matricei şi cu cei combinaţi obţinuţi anterior în cadrul fiecărei matrice. Dintre vectorii deja existenţi în matrice sunt eliminaţi cei acoperiţi de noul vector. Un vector combinat este adăugat numai dacă nu este acoperit de un vector iniţial sau de un vector combinat obţinut anterior.

Din mulţimea formată din vectorii iniţiali şi combinaţi corespunzătoare fiecărei clici este ales un subset care să acopere toţi vectorii implicanţi. Pentru grupa M_1 sunt selectaţi vectorii (V_1,V_7). Ceilalţi vectori sunt redundanţi (V_2 este acoperit de $V_1 \oplus V_7$, V_3 este acoperit de V_7 şi V_8 este acoperit de $V_1 \oplus V_7$).

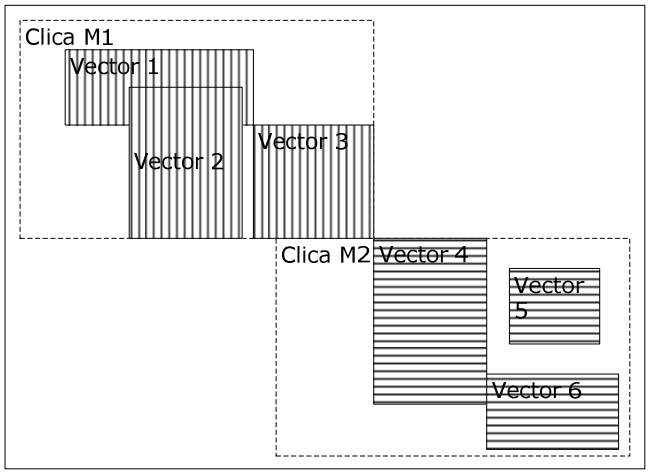


Fig. 6-3 Împărţirea matricei $M(V_1,V_2,V_3,V_4,V_5,V_6)$ în clici

$M(M_1(V_1,V_2,V_3),M_2(V_3,V_4,V_5))$

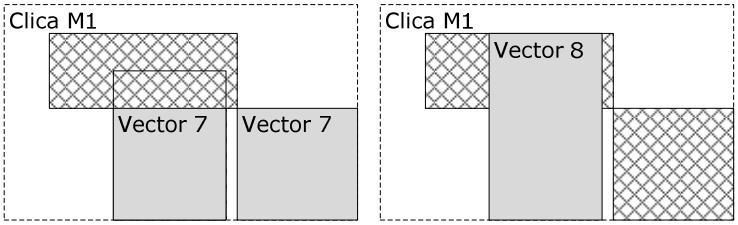


Fig. 6-4 Vectorii combinaţi pentru M_1

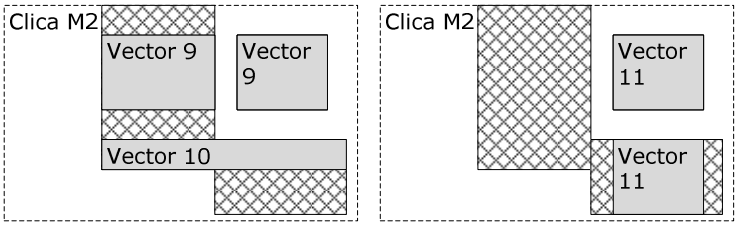


Fig. 6-5 Vectorii combinaţi pentru M_2

Pentru matricea M_2, deoarece nu este evidentă selectarea vectorilor implicanţi maximali, este necesară o abordare analitică. Se creează grupe de vectori primari având drept criteriu mulţimea de vectori implicanţi care îi acoperă. Pentru exemplul dat se formează 11 grupe ($P_1...P_{11}$), ca în figura Fig. 6-6. Grupele de vectori primari se obţin prin transformarea matricei formată din vectorii iniţiali şi vectorii combinaţi într-o matrice de vectori disjuncţi prin descompunerea vectorilor după coloanele de adiacenţă (coloanele se obţin cu ajutorul operatorului de adiacenţă $a\langle\ \rangle$).

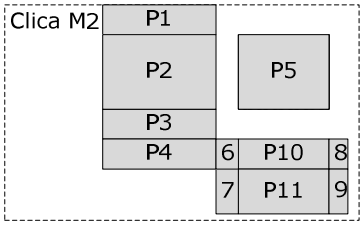


Fig. 6-6 grupe de vectori primari pentru M_2

Soluţia minimă se obţine utilizând, pentru fiecare grupă, matricea de acoperire dintre vectorii implicanţi şi rândurile grupei, conform algoritmului lui Thelen. Tab. 6-1 conţine matricea de acoperire pentru exemplul din Fig. 6-3.

Vectorii V_4 şi V_6 sunt vectori esenţiali deoarece există vectori primari acoperiţi numai de V_4 sau numai de V_6. P_1 şi P_3 sunt acoperiţi doar de V_4, iar P_7 şi P_9 sunt acoperiţi doar de V_6. V_4 şi V_6 vor intra obligatoriu în soluţie.

Matricea de acoperire conform algoritmului lui Thelen este:

	V_4	V_5	V_6	V_9	V_{10}	V_{11}	Nr. acoperiri
P_1	x						1
P_2	x			x			2
P_3	x						1
P_4	x				x		2
P_5		x		x		x	3
P_6			x		x		2
P_7			x				1
P_8			x		x		2
P_9			x				1
P_{10}			x		x	x	3
P_{11}			x			x	2

Tab. 6-1 Matricea de acoperire pentru M_2

Selectarea unui implicant în soluţie este urmată de eliminarea acestuia din matrice împreună cu rândurile acoperite. După eliminarea coloanelor corespunzătoare vectorilor V_4 şi V_6 şi a rândurilor acoperite ($P_1...P_4$ şi $P_6...P_{11}$) rămâne de rezolvat numai P_5 (Tab. 6-2). Pentru P_5 se poate alege oricare dintre vectorii V_5, V_9 sau V_{11}. Dacă există un criteriu suplimentar de selectare (cum ar fi numărul de poziţii specificate din implicanţi) acesta poate fi aplicat pentru a alege unul din cei trei vectori.

	V_5	V_9	V_{11}	Nr. acoperiri
P_5	x	x	x	3

Tab. 6-2 Matricea de acoperire pentru M_2

Concluzii la determinarea implicanţilor maximali prin combinare

Metoda operează ca un motor de inferenţă care face toate deducerile posibile pentru un set de afirmaţii date. Rezultatul este mulţimea de vectori implicanţi maximali.

În forma prezentată metoda nu poate genera combinaţii, în cazul funcţiilor incomplet specificate, între vectorii specificaţi şi combinaţiile de intrare nespecificate astfel încât să se obţină o minimizare eficientă. Este necesară adăugarea unei procesări suplimentare care să determine o matricea cu combinaţiile de intrare nespecificate, numită aici matrice **default**. Matricea fiecărei grupe este reunită cu matricea „**default**" înainte de aplicarea algoritmului.

Pentru selectarea implicanţilor care formează soluţia am utilizat algoritmul lui Thelen, dar urmărind numai acoperirea vectorilor iniţiali. Metoda de complementare utilizată pentru calcul matricei „**default**" conduce la rezultate exprimate prin vectori maximali, ceea ce o face o metodă de minimizare în sine. Calculul matricei „**default**" împreună cu un algoritm propus pentru determinarea implicanţilor prin complementare este prezentat separat, în continuare.

6.3 Determinarea implicanţilor maximali prin complementarea matricelor (COMP)

Pornind de la operaţia de complementare a matricelor şi de la proprietăţile rezultatului acestei operaţii am implementat un algoritm de determinare vectorilor implicanţi maximali.

Etapele algoritmului (notate cu **COMP n**) sunt:

COMP 1. Se împart vectorii de intrare din tabelul de specificare în grupe având aceeaşi valoare de ieşire. Vectorii din fiecare grupă i se memorează în câte o matrice M_i.

COMP 2. Matricele obţinute la etapa 1 se reunesc şi se obţine mulţimea combinaţiilor de intrare specificate .

COMP 3. Prin complementarea matricei M_S se obţin combinaţiile de intrare nespecificate ($\overline{M_S}$).

COMP 4. Reunind fiecare matrice M_i cu matricea $\overline{M_S}$ se construieşte câte o matrice extinsă M_i^E pentru fiecare grupă.

COMP 5. Dubla complementare a matricei M_i^F conduce la determinarea vectorilor implicanţi maximali pentru grupa i. Operaţia se aplică pentru fiecare grupă în parte.

COMP 6. Determinarea soluţiilor se face prin selectarea unei submulţimi de vectori maximali pentru fiecare grupă astfel încât să fie acoperiţi vectorii din matricea M_i fără a ţine cont de cei din matricea M_i^F.

Concluzii la determinarea implicanţilor prin complementare

Faţă de algoritmul precedent acesta are avantajul că utilizează în procesul de minimizare combinaţiile de intrare nespecificate. Algoritmul de determinare a implicanţilor prin combinare se poate îmbunătăţi prin adăugarea etapelor **COMP 2**, **COMP 3** şi **COMP 4** între etapele **COMB 1** şi **COMB 2** şi înlocuirea etapei **COMB 4** cu **COMB 6**.

Algoritmul prezentat are dezavantajul că o parte din vectorii care apar pe parcursul procesării sunt generaţi de mai multe ori, ceea ce conduce la o redundanţă crescută a procesării. Eliminarea acestui dezavantaj se face prin înlăturarea posibilităţii de a genera de mai multe ori acelaşi vector şi/sau prin memorarea rezultatelor obţinute în procesare.

Metoda prezentată în capitolul următor elimină procesările redundante prin:

- utilizarea unei metode de generare a vectorilor implicanţi care nu permite crearea de două ori a aceluiaşi vector;
- memorarea unui set minim de informaţii pentru fiecare vector generat care înglobează rezultatele procesărilor anterioare.

6.4 Determinarea implicanţilor maximali cu număr minim de literale

În cadrul acestei metode determinarea implicanţilor maximali se obţine prin descompunerea vectorului total până când fiecare din vectorii obţinuţi acoperă sau intersectează un singur vector iniţial sau mai mulţi vectori iniţiali cu aceeaşi valoare de ieşire. Această metodă are caracteristic faptul că informaţia determinată în timpul procesării este păstrată pentru fiecare vector şi grupare de vectori cu scopul de a elimina cât mai multe calcule redundante. Procesarea se mai bazează pe aceste informaţii şi pentru a elimina cât mai timpuriu variantele de descompunere care nu pot conduce la determinarea de noi vectori implicanţi.

Pentru a determina caracteristicile algoritmului bazat pe descompunerea (**DESC**) spaţiului de intrare am utilizat pentru început specificaţii tabelare deterministe cu o singură ieşire.

Etapele metodei (notate cu **DESC n**) sunt:

DESC 1. Descompunerea specificării iniţiale în rânduri disjuncte.

DESC 2. Crearea tuturor vectorilor cu o singură poziţie specificată şi determinarea setului de atribute asociat acestora.

DESC 3. Clasificarea rezultatelor în funcţie de atributele acestora.

DESC 4. Marcarea în tabelul iniţial a rândurilor acoperite. Dacă nu mai există rânduri nemarcate algoritmul continuă de la **DESC 6**.

DESC 5. Corelarea vectorilor obţinuţi în unul din paşii anteriori (**DESC 2** sau **DESC 3**) pentru a obţine vectori cu un număr de poziţii specificate mai mare cu 1 decât cei precedenţi. Algoritmul continuă de la **DESC 3**.

DESC 6. Selectarea unui set de vectori implicanţi maximali.

DESC 7. Eliminarea redundanţele din soluţie.

Principiu metodei

Metoda se bazează pe determinarea „efectelor" valorilor specificate (literalelor) dintr-un tabel de intrare în stabilirea valorii de ieşire. Metoda urmăreşte determinarea unui set de literale minim, suficient pentru a discrimina combinaţiile de intrare cu ieşiri diferite. Ideea în jurul căreia este construită această metodă este descompunerea nedisjunctă a vectorului total în concordanţă cu literalele utilizate în specificarea iniţială.

Descompunere primară a vectorului total

Iniţial se construieşte o structură de date neafectată de o ordine fixată a variabilelor. Ea este formată dintr-o mulţime de vectori nedisjuncţi care au asociate atributele (informaţiile) de plecare ale algoritmului.

Descompunerea are următoarele caracteristici:

- fiecare vector are o singură poziţie specificată;
- numărul de vectori este egal cu suma numărului de valenţe utilizate în specificare pentru toate domeniile de intrare;
- fiecărui vector i se asociază mulţimea de rânduri acoperite din tabelul iniţial;
- fiecărui vector i se asociază mulţimea de rânduri care nu au valoare specificată (conţin DC) pe poziţia literalului din vector. Aceste rânduri se intersectează cu vectorul. Informaţia este comună pentru toţi vectorii care au specificat literal pe aceeaşi poziţie.

Notaţii utilizate. Exemplu.

Pachetele de informaţiile obţinute în timpul descompunerii primare sunt notate cu C_i^v, unde:

c. i reprezintă indicele variabilei de intrare;
d. v reprezintă valoarea variabilei respective.

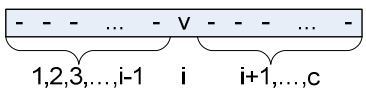


Fig. 6-7 Vectorul asociat mulţimii C_i^v pentru un tabel cu C coloane de intrare

Dacă în coloana corespunzătoare intrării i dintr-o specificaţie se regăseşte valenţa v pe rândurile 6, 7 şi 9 şi valenţa este nespecificată (DC) pe rândurile 1, 2, 3 4 5 şi 8, atunci variabilei i pentru valenţa v îi sunt asociate două mulţimi: $C_i^v = \{6,7,9\}$ şi $C_i^{DC} = \{1,2,3,4,5,8\}$.

Pentru funcţia $f : \{1,2\}^6 \rightarrow \{1,2\}$, descrisă în tabelul Tab. 6-3,

#	Intrare						Ieşire
1	0	0	-	0	-	-	0
2	0	-	0	0	-	-	0
3	0	-	-	0	-	0	0
4	1	0	1	-	-	-	0
5	-	1	1	-	-	1	1
6	-	1	-	1	-	-	1
7	1	1	-	-	-	-	1
8	0	-	-	1	-	-	1
9	-	-	0	1	-	-	1
10	1	-	0	-	-	-	1

Tab. 6-3 Tabel de intrare

rezultatul descompunerii primare se află în Tab. 6-4.

Intrare i	Valoare v	Informaţii C_i^v
1	-	{4,5,8}
	0	{0,1,2,7}
	1	{3,6,9}
2	-	{1,2,7,8,9}
	0	{3}
	1	{4,5,6}
3	-	{0,2,5,6,7}
	0	{1,8,9}
	1	{3,4}
4	-	{3,4,6,9}
	0	{0,1,2}
	1	{5,7,8}
5	-	{0,1,2,3,4,5,6,7,8,9}
	0	ϕ
	1	ϕ
6	-	{0,1,3,5,7,8,9}
	0	{2}
	1	{4}

Tab. 6-4 Tabel cu mulţimi de rânduri acoperite

Tab. 6-3 are 6 coloane numerotate de la 1 la 6. pentru fiecare coloană se va crea câte un număr de mulţimi egal cu numărul de valenţe şi una suplimentară, pentru utilizarea valorii DC. Cele şase variabile binare vor avea asociate câte 3 mulţimi (2 de rânduri acoperite şi una de rânduri intersectate).

Dacă într-o coloană corespunzătoare unei variabile de intrare nu este utilizată valoarea DC, atunci mulţimea C^{DC} este vidă.

Pentru fiecare set de mulţimi asociat unei variabile fiecare rând este conţinut într-una din aceste mulţimi.

Aparatul matematic asociat descompunerii tabelului de intrare

Teoremă: Mulţimile de acoperire corespunzătoare unei variabile sunt disjuncte.

Demonstraţie: Teorema de mai sus este evidentă deoarece din modul de construcţie al mulţimilor un rând este adăugat unei singure mulţimi.

Teoremă: Reuniunea mulţimilor de acoperire asociate unei variabile este egală cu mulţimea tuturor rândurilor tabelului de intrare.

Demonstraţie: Din modul de construcţie este evident că fiecare rând a fost adăugat unei mulţimi şi implicit, reuniunea acestor mulţimi este egală cu mulţimea tuturor rândurilor.

Pentru fiecare mulţime C_i^V va fi construit un vector care conţine numai poziţii nespecificate (DC), cu excepţia coloanei i, unde rândul va avea valoarea v, ca în Fig. 6-7.

Vectorul asociat cu C_i^V $(v \neq DC)$ are următoarele proprietăţi:

1. are elemente comune cu toate rândurile din mulţime dacă este interpretat ca un subspaţiu al unui spaţiu c - dimensional (unde c este numărul de dimensiuni), atunci vectorul include toate rândurile care au valoarea v pe coloana i. În Fig. 6-8 este prezentat un spaţiu tridimensional $A \times B \times C$, unde $A = \{0,1,2,3\}$, $B = \{0,1,2,3\}$ şi $C = \{0,1,2,3,4,5\}$. În acest spaţiu este marcată acoperirea dată de vectorul asociat mulţimii C_B^2;
2. nu are elemente comune cu un C_i^w, unde $w \neq v$ şi $w \neq DC$;
3. se intersectează cu vectorii asociaţi rândurilor C_i^{DC}, dar nu include (nu acoperă) în totalitate aceste rânduri;
4. se intersectează cu vectorul asociat mulţimii C_j^w, unde $j \neq i$, dar nu este acoperit de acesta şi nici nu îl acoperă;
5. se intersectează cu toate rândurile din mulţimea C_j^{DC}, unde $j \neq i$, dar nu acoperă aceste rânduri şi nici nu este acoperit de ele.

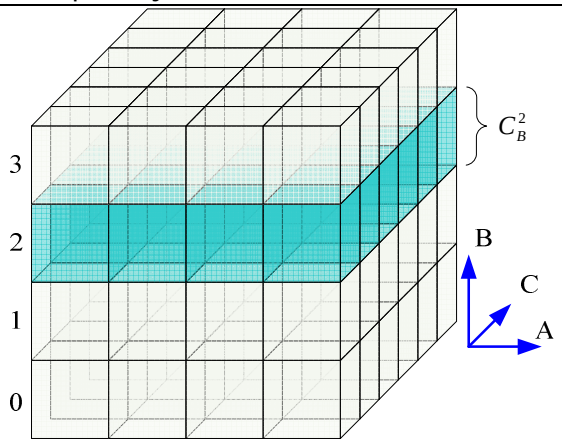


Fig. 6-8 Acoperirea lui C_B^2

Exemplu si notaţii complementare

Datorită proprietăţilor aparte ale vectorului asociat mulţimii C_i^{DC} acesta va fi notat cu I_i. În exemplul de mai jos este prezentată o funcţie care conţine şi variabile ce nu sunt binare $f : Z_5 \times Z_2 \times Z_3 \to Z_4$.

#	I			O
1	3	5	7	9
2	1	7	2	7
3	5	7	2	7
4	2	-	4	3
5	4	-	7	0
6	3	7	7	0
7	-	5	2	0

Tab. 6-5 Tabel de intrare

Tabelul Tab. 6-6 conţine mulţimile de acoperire asociate variabilelor de intrare ale tabelului.

Intrare	Valoare	Notaţie	Rânduri acoperite
1	-	I_1	$\{7\}$
	1	C_1^1	$\{2\}$
	2	C_1^2	$\{4\}$
	3	C_1^3	$\{1,6\}$
	4	C_1^4	$\{5\}$
	5	C_1^5	$\{3\}$

2	-	I_2	$\{4,5\}$
	5	C_2^5	$\{1,7\}$
	7	C_2^7	$\{2,3,6\}$
3	-	I_3	$\{\Phi\}$
	2	C_3^2	$\{2,3,7\}$
	4	C_3^4	$\{4\}$
	7	C_3^7	$\{1,5,6\}$

Tab. 6-6 Mulţimi de acoperire

Tripletul $\left(C_i^v I_i V_i^v\right)$ va fi numit vector cu atribute şi se va nota cu S_i^v, fiind descris prin:

1. mulţimea C_i^v ;
2. mulţimea I_i
3. vectorul V_i^v, format numai din valori DC, cu excepţia poziţiei i, unde se regăseşte valenţa v a variabilei de intrare i.

Corelarea vectorilor cu atribute

Considerăm doi vectori cu atribute: $S_i^v = \left(C_i^v I_i V_i^v\right)$ şi $S_j^w = \left(C_j^w I_j V_j^w\right)$. Vectorii caracterizaţi de S_i^v şi S_j^w sunt $V_i^v = (--...-v-...--)$, cu valenţa v pe poziţia pe poziţia i şi respectiv $v_j^w = (--...-w-...--)$ cu valenţa w pe poziţia pe poziţia j. Considerăm $i < j$. Intersecţia vectorilor V_i^v şi V_j^w este vectorul V_{ij}^{vw} din Fig. 6-9. Corelarea dintre S_i^v şi S_j^w înseamnă determinarea vectorului cu atribute S_{ij}^{vw}.

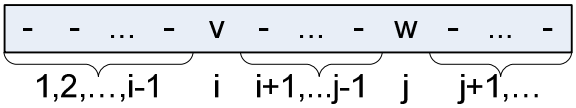


Fig. 6-9 Vectorul asociat mulţimii C_{ij}^{vw}

Determinarea atributelor C_{ij}^{vw} şi I_{ij} înseamnă determinarea rândurilor acoperite şi a rândurilor intersectate vectorul V_{ij}^{vw}. Acestea se află numai printre elementele mulţimilor C_i^v, C_j^w, I_i şi I_j. Rândurile intersectate sau acoperite de S_i^v sunt $C_i^v \cup I_i$ iar de S_j^w sunt $C_j^w \cup I_j$.

Rândurile acoperite sau intersectate de S_{ij}^{vw} fac parte din mulţimea $\left(C_i^v \cup I_i\right) \cap \left(C_j^w \cup I_i\right)$.

$$\left(C_i^v \cup I_i\right) \cap \left(C_j^w \cup I_i\right) = \left(C_i^v \cap C_j^w\right) \cup \left(C_i^v \cap I_j\right) \cup \left(I_j \cap C_j^w\right) \cup \left(I_i \cap I_j\right), \quad (6\text{-}1).$$

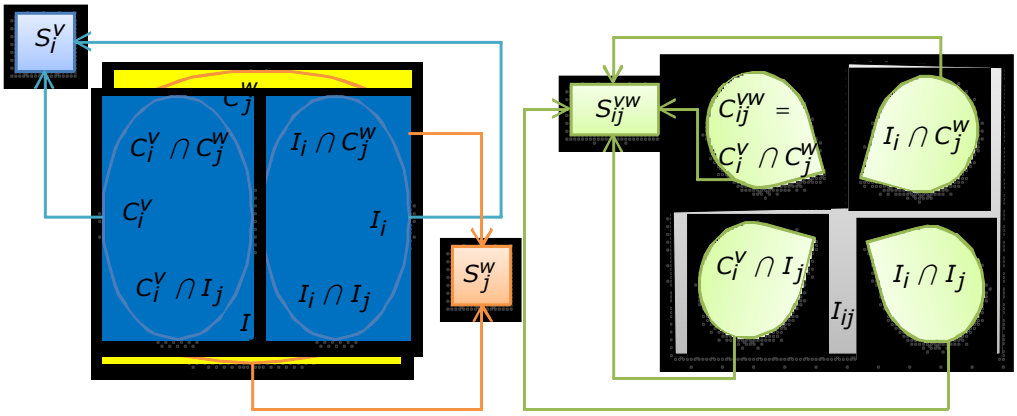


Fig. 6-10 Intersecţia vectorilor V_i^V şi V_j^W şi calculul mulţimilor corespunzătoare

Rezultatul conţine patru termeni disjuncţi:

1. $C_i^v \cap C_j^w$: reprezintă rândurile complet acoperite şi de V_i^v şi de V_j^w - deci total acoperite de V_{ij}^{vw}
2. $C_i^v \cap I_j$: reprezintă rândurile acoperite de V_i^v şi intersectate de V_j^w - parţial acoperite de V_{ij}^{vw};
3. $I_i \cap C_j^w$: reprezintă rândurile acoperite de V_j^w şi parţial acoperite de V_i^v - deci parţial acoperite de V_{ij}^{vw};
4. $I_i \cap I_j$: reprezintă rândurile parţial acoperite şi de V_i^v şi de V_j^w - deci parţial acoperite de V_{ij}^{vw}.

Aceste mulţimi corespund rândurilor acoperite de C_{ij}^{vw}, respectiv intersectate I_{ij} de vectorul V_{ij}^{vw}.

În continuare se va nota cu „$\otimes$” operatorul de corelare definit astfel:

$$S_i^v\left(C_i^v I_i V_i^v\right) \otimes S_j^w\left(C_j^w I_j V_j^w\right) = S_{ij}^{vw}\left(C_{ij}^{vw} I_{ij} V_{ij}^{vw}\right) \quad (6\text{-}2)$$

unde

$$C_{ij}^{vw} = C_i^v \cap C_j^w \tag{6-3}$$

$$I_{ij} = \left(C_i^v \cap I_j\right) \cup \left(I_i \cap C_j^w\right) \cup \left(I_i \cap I_j\right) \tag{6-4}$$

Observaţie Operatorul de corelare aplicat pentru doi vectori care au mulţimile de rânduri acoperite disjuncte conduce la un vector care, chiar dacă mai poate intersecta rânduri din tabelul iniţial, nu mai acoperă total nici un rând. Această observaţie poate fi utilizată pentru optimizarea implementării operaţiei de corelare.

Operatorul este comutativ şi asociativ. Pentru un tabel cu c coloane de intrare elementul neutru este $I\left(U, \Phi, DC^c\right)$ şi elementul nul este $O(\Phi, \Phi, -)$.

Clasificarea vectorilor cu atribute

Vectorii se clasifică în funcţie de cardinalitatea mulţimilor de rânduri acoperite şi intersectate şi de numărul de valori de ieşire distincte corespunzătoare rândurilor din mulţimile C şi I. Vom nota numărul de valori de ieşire distincte cu O. În tabelul Tab. 6-7 sunt prezentate tipurile de vectori cu atribute posibile.

Nr.Crt.	C	I	O	Descrierea vectorului	Clasificare
1	ϕ	ϕ	-	acoperă numai combinaţii de intrare nespecificate	Neviabil
2	ϕ	$\neq \phi$	$= 1$	acoperă vectori primari din aceeaşi grupă de valori	Neviabil
3	ϕ	$\neq \phi$	> 1	acoperă vectori primari din diverse grupe de valori	Neviabil
4	$\neq \phi$	ϕ	$= 1$	este implicant	Implicant
5	$\neq \phi$	ϕ	> 1	acoperă rânduri din diverse grupe de valori	Candidat
6	$\neq \phi$	$\neq \phi$	$= 1$	este implicant	Implicant
7	$\neq \phi$	$\neq \phi$	> 1	vectorul este implicant	Candidat

Tab. 6-7 Clase de vectori

Vectori implicanţi - Un vector este implicant dacă toate rândurile acoperite şi intersectate au aceeaşi ieşire, iar mulţimea rândurilor acoperite nu este vidă.

Vectori candidaţi - Un vector este candidat dacă mulţimea de rânduri acoperite nu este vidă şi există cel puţin două rânduri intersectate sau acoperite care au ieşiri diferite. Prin operaţia de corelare a acestuia cu un alt vector candidat rezultatul poate conţine numai rânduri cu aceeaşi ieşire, astfel încât să se obţină un vector implicant.

Vectori neviabili - Vectorii care nu acoperă nici un rând sunt eliminaţi deoarece ei acoperă combinaţii de intrare nespecificate de sistemul decizional.

Rafinarea algoritmului de determinare a implicanţilor (IMPL)

Paşii algoritmului pentru determinarea implicanţilor (notaţi cu **IMPL n**) sunt:

IMPL 1. Toate rândurile tabelului iniţial sunt marcate ca nerezolvate.

IMPL 2. Se formează atributele pentru vectorii cu o singură poziţie specificată.

IMPL 3. Clasificarea vectorilor obţinuţi la pasul executat anterior (**IMPL 2** sau **IMPL 8**) în implicanţi, candidaţi şi neviabili.

IMPL 4. Vectorii neviabili sunt eliminaţi.

IMPL 5. Sunt marcate rândurile acoperite de vectorii implicanţi.

IMPL 6. Dacă toate rândurile iniţiale au fost acoperite algoritmul se opreşte.

IMPL 7. Vectorii implicanţi sunt adăugaţi la soluţie şi eliminaţi din pachetul de vectori cu care se continuă procesarea.

IMPL 8. Se formează vectorii cu mai mult cu o poziţie specificată decât vectorii obţinuţi în pasul **IMPL 2** sau **IMPL 8** executat anterior prin combinarea vectorilor candidaţi cu vectorii obţinuţi în etapa **IMPL 2**.

IMPL 9. Reluarea algoritmului de la etapa **IMPL 3**.

Algoritmul formează vectori cu număr din ce în ce mai mare de poziţii specificate. Operatorul de corelare este utilizat pentru a determina caracteristicile noilor vectori pe baza vectorilor precedenţi. Pe măsură ce sunt determinaţi, vectorii sunt clasificaţi şi sunt păstraţi doar cei de tip candidat – care pot conduce la formarea de noi vectori implicanţi. Pe măsură ce evoluează algoritmul rândurile tabelului considerate iniţial neacoperite (**IMPL 1**) sunt marcate ca acoperite (**IMPL 5**) de către vectorii implicanţi. Rularea se opreşte când s-a creat un set suficient de implicanţi care să acopere toate rândurile tabelului dat (**IMPL 6**).

Pregătirea datelor utilizate în derularea algoritmului (**IMPL 2**) constă în determinarea vectorilor cu o singură poziţie diferită de *DC* . Determinarea setului de atribute al acestor vectori se obţine din analiza valorilor specificate ale fiecărui vector din tabelul iniţial. În sistemul specificat prin tabelul Tab. 6-3 fiecare variabilă de intrare are două valenţe. Pe coloana variabilei 5 nu apare decât valoarea *DC* . Practic variabila 5 nu afectează funcţionarea sistemului decizional. Pentru variabilele 1, 2, 3, 4 şi 6 sunt folosite ambele valenţe. Pentru fiecare dintre ele vor fi formate câte două mulţimi de rânduri acoperite şi câte o mulţime de rânduri intersectate.

Diagrama algoritmului este prezentată în Fig. 6-11

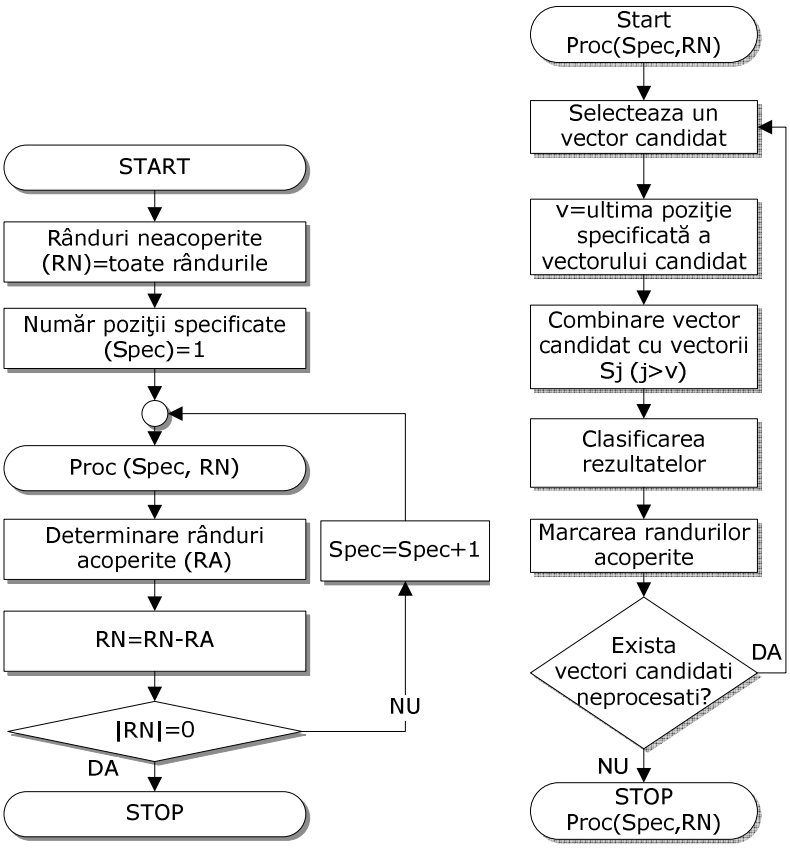


Fig. 6-11 Organigrama generală a algoritmului

Optimizarea algoritmului

a) Pentru a evita creşterea nejustificată a memoriei utilizate şi implicit a spaţiului de căutare, în cadrul etapei **IMPL 2** sunt utilizate următoarele tehnici:
   - vectorii sunt clasificaţi imediat ce au fost obţinuţi şi sunt eliminaţi din pachetul de date procesat dacă sunt implicanţi sau neviabili.
   - cardinalitatea mulţimii rândurilor acoperite stabileşte dacă un vector poate să conducă spre obţinerea unui vector implicant şi de aceea se va efectua prima dată numai determinarea acestei mulţimi şi, dacă această mulţime nu este vidă, se construieşte şi mulţimea rândurilor intersectate.

b) Pentru ca fiecare vector să fie generat o singură dată adăugarea unei noi poziţii specificate se face după următoarea regulă: un vector cu poziţiile specificate p_1, p_2, … , p_n unde $p_1 < p_2 < ... < p_n$ se combină numai cu vectori cu o poziţie specificată p unde $p_n < p$.

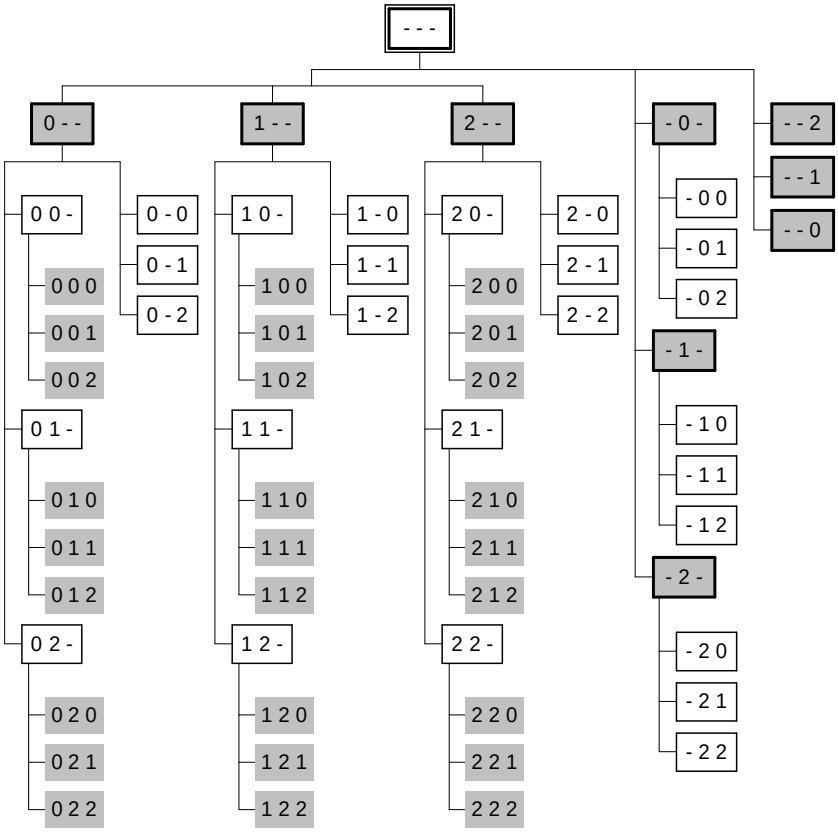


Fig. 6-12 Arborele de vectori pentru o funcţie cu trei intrări multivalente

Fiecare vector trebuie să conţină memorată ultima valoare specificată pentru a selecta direct, fără calcul suplimentar, vectorii cu care va fi compus.

Selectarea soluţiilor (SOL)

Mulţimile de vectori implicanţi obţinute în fiecare iteraţie a algoritmului sunt reunite, formând mulţimea vectorilor implicanţi din care urmează să fie selectată o soluţie.

Selectarea vectorilor implicanţi se face în mai multe etape (notate **SOL n**):

SOL 1. Determinarea implicanţilor esenţiali.

SOL 2. Eliminarea implicanţilor esenţiali şi a rândurilor acoperite de aceştia.

SOL 3. Eliminarea rândurilor dominate.

SOL 4. Eliminarea coloanelor dominate.

SOL 5. Dacă a existat un implicant esenţial, un rând dominat sau o coloană dominată se reia algoritmul de la etapa **SOL 1**.

SOL 6. Determinarea nucleelor de vectori implicanţi independenţi.

SOL 7. Alegerea unei soluţii pentru fiecare nucleu în parte.

SOL 8. Crearea soluţiei finale prin reunirea soluţiilor alese pentru fiecare nucleu în parte şi a implicanţilor esenţiali determinaţi la **SOL 1**.

Reducerea volumului de calcul pentru determinarea soluţiei se face prin procesarea independentă a nucleelor (**NUCL**) de vectori implicanţi [211] - submulţimi disjuncte de implicanţi care pot fi prelucrate independent. Fiecărui nucleu îi este asociată o mulţime de rânduri pe care le acoperă implicanţii din acesta. Mulţimile de rânduri asociate nucleelor sunt disjuncte. Determinarea nucleelor se face utilizând următorul algoritm (ai cărui paşi sunt notaţi **NUCL n**):

NUCL 1. Se marchează toţi implicanţii ca neprelucraţi.

NUCL 2. Se marchează toate rândurile ca neprelucrate.

NUCL 3. Dacă nu există implicanţi neprelucraţi algoritmul se termină.

NUCL 4. Se creează un nucleu nou vid *CORE* şi o mulţime vidă de rânduri *ROWS* asociată nucleului.

NUCL 5. Se alege un implicant *I* neprelucrat şi se adaugă la nucleul *CORE* .

NUCL 6. Se marchează implicantul *I* ca nevizitat.

NUCL 7. Pentru fiecare implicant nevizitat din *CORE* se adaugă rândurile neprelucrate la mulţimea de rânduri acoperite *ROWS* . Implicantul se marchează ca vizitat iar rândurile adăugate ca nevizitate.

NUCL 8. Pentru fiecare rând din *ROWS* nevizitat se adaugă implicanţii neprelucraţi care îl acoperă în *CORE* . Rândul este marcat ca vizitat iar implicanţii adăugaţi ca nevizitaţi.

NUCL 9. Dacă există implicanţi nevizitaţi în *CORE* se reia algoritmul de la **NUCL 7**.

NUCL 10. Se reia algoritmul de la **NUCL 3**.

Alegerea soluţiei pentru fiecare nucleu în parte are ca obiectiv obţinerea unui număr cât mai mic de vectori implicanţi care să acopere toate rândurile din nucleu.

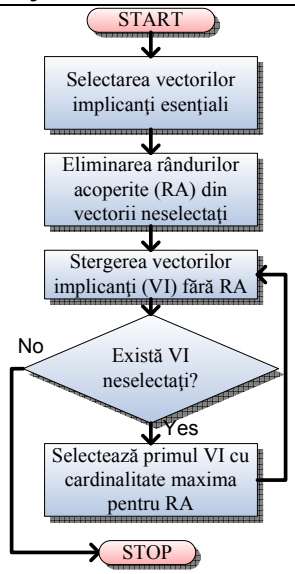


Fig. 6-13 Algoritmul de creare a unei soluţii

Alegerea unei soluţii se face pentru fiecare nucleu în parte cu un algoritm de tip Greedy (**GRE**). Paşii acestui algoritm (notaţi **GRE n**) sunt:

GRE 1. Se sortează în ordine descrescătoare vectorii implicanţi după numărul de rânduri acoperite

GRE 2. Se adaugă primul vector într-o mulţime a vectorilor ce aparţin soluţiei şi se marchează în nucleu rândurile acoperite de acest vector.

GRE 3. Se elimină toate rândurile acoperite de rândul selectat din mulţimile de rânduri acoperite.

GRE 4. Se elimină primul vector (cel selectat) şi toţi vectorii implicanţi care nu mai acoperă nici un rând.

GRE 5. Dacă mai există rânduri neacoperite în tabelul iniţial se reia algoritmul de la primul pas.

Rezultate

Utilizarea acestei metode a dat rezultate foarte bune în comparaţie cu metodele BOOM şi MVSIS.

De exemplu, pentru o specificaţie cu 50 de variabile de intrare şi 150 de rânduri MVSIS a dat un rezultat cu procedura euristică format din 3895 de rânduri, iar cu procedura exactă nu a terminat execuţia. Aplicaţia BOOM a dat ca rezultat tabelul de mai jos:

----10----------0----------10-0-	1
--------1---------1---01---1----------0-	1
-------0--------1---1------10----------	1
----------1---0--0----------0-1-	1
----------0----------010----0--	1
1---------0-1----------1----------0-	1
0------1---1----------1-------0----------	1
----------1---0----------0-10-0----------	1
--1-11------00----------0----------	1
----------0-------1------111	1
-------0----------1-----10--1----------	1
----------11-1--00-1----------	1
--1--110----1----------	1
----------1-00--1-	1
-------1----------0--0----------1-01-1----	1
-1----------1--0----------1-----0	1
--------0----------0---0----1-------0-	1
----------0--------11----------0--------1-----	1
----------00-00---0----------	1
----------0---------0--1----------1--1---	1
----------1--0--1----1----------1---	1
----------0--1----------00----------0------	1
------0--------0---------00----------1-0------	1
----1-1--0----------1---0--1-	1
1----------0----------1----------0-1--	1
----------1--011----------1----------	1
---1----------0--------1--01-1--------	1
----------00------0---0---0----------	1
----------1--0----1---0----0--0----------	1

Tab. 6-8 Minimizare realizată cu BOOM

Rezultatul obţinut în urma rulării cu metoda prezentată, conform schemei din Fig. 6-1, cu toţi algoritmii cuplaţi, este:

----------0----101-----	1
----------0---------10----1----	1
----------0----1---1---------1--	1
----------1----------01---------1	1
----------0----0-1----------0---	1
----------0----------0---1-0---	1
---------0-----0---1--0----------	1
------1----0----------0---------0-------	1
---0----------0-01--------	1
---0----0----------0----------0-	1
-0-11----------0----------	1
-1----------1--0----------0----------	1
-10----------0--------0----------	1
1----------1----0----------1------	1
1-1----1------1----------	1

Tab. 6-9 Minimizare realizată cu metoda prezentată

Ca urmare a minimizării efectuate cu noua metodă nu au mai fost necesare variabilele 1, 6, 9, 23, 35 şi 44. Tabelul este mai mic cu 6 coloane de intrare.

Rezultatul aplicaţiei BOOM este format din 155 de literale iar cel obţinut cu această metodă este format din 60 de literale.

Pentru a compara timpul de execuţie am comparat metoda din aplicaţia MVSIS şi metoda prezentată pe tabele multivalente cu 16 variabile şi 65000 de rânduri. Timpul execuţie este de două ori mai mare decât cel de la MVSIS. Testele au fost efectuate numai până la 16 variabile deoarece MVSIS nu poate minimiza decât euristic tabele mai multe variabile de intrare.

6.5 Preprocesarea şi postprocesarea datelor

6.5.1 Încărcarea şi validarea sistemelor decizionale stocate în fişiere

Procesul de minimizare implică suplimentar faţă de calculul propriu-zis etapele de încărcare a sistemelor decizionale şi de validare a acestora. Încărcarea constă în instanţierea structurilor de date asociate sistemului decizional, iar validarea în analiza acestora cu scopul de a determina dacă sistemul poate fi minimizat cu metoda prezentată. Metodele din acest capitol au fost dezvoltate numai pentru minimizarea sistemelor univoce cu o singură ieşire. Sistemele neunivoce pot fi transformate în sisteme univoce prin eliminarea neunivocităţilor.

Structura fişierelor de intrare

Sistemele decizionale sunt stocate în fişiere care utilizează un format adecvat pentru reprezentarea domeniilor variabilelor de intrare şi de ieşire şi pentru specificarea corespondenţei intrare/ieşire. Fişierul conţine trei tipuri de specificaţii:

e. descrieri de domenii $: v_0 v_1 ... v_{n-1}$. Valorile utilizate în cadrul corespondenţelor intrare/ieşire nu trebuie adăugate la definiţia domeniului deoarece aplicaţia face implicit completarea domeniului. Prin intermediul acestor definiţii se adăugă valori neutilizate în corespondenţele intrare/ieşire;
f. descrieri de corespondenţe V_i / V_0 formate dintr-un vector de intrare şi un vector de ieşire;
g. descrierea unei valori implicite $/V_0$ formate numai dintr-un vector de ieşire. Ieşirea V_0 se activează când la ieşirea sistemului este prezentă o combinaţie care nu apare în nici o corespondenţă intrare/ieşire.

Exemplu: În fişierul de mai jos primele două linii specifică valorile neutilizate ale variabilelor de intrare, a treia linie indică valoarea de ieşire pentru combinaţia $(F \ F)$, iar cea de patra linie indică ieşirea pentru combinaţiile de intrare nespecificate (în acest caz este vorba de combinaţiile care conţin valoarea T pe prima sau pe a doua poziţie).

Fig. 6-14 Exemplu de fişier de intrare

În specificarea de mai jos nu mai este necesară descrierea domeniilor de definiţie deoarece toate valenţele variabilelor de intrare sunt utilizate în descrierea relaţiilor intrare/ieşire. Ambele variabile de intrare au acelaşi domeniu de definiţie $\{F,T\}$:

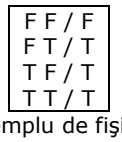


Fig. 6-15 Exemplu de fişier de intrare

Validarea structurii fişierelor

Pe măsură ce este citit fişierul de intrare sunt instanţiate structurile de date asociate sistemului decizional şi sunt verificate următoarele:

- fiecare rând trebuie să conţină o secvenţă de valori de intrare şi o secvenţă de valori de ieşire sau numai o secvenţă de valori de ieşire atunci când este prezentă o valoare de ieşire explicită.
- să existe cel mult o secvenţă de valori de ieşire implicite;
- numărul de coloane de intrare şi ieşire să fie acelaşi pentru toate rândurile care specifică funcţia;
- numărul de definiţii de domenii nu este mai mare decât numărul de coloane de intrare şi coloane de ieşire.

Multiplicarea sistemelor decizionale cu ieşiri multiple

Dacă sistemul decizional are mai multe ieşiri atunci acesta este multiplicat pentru fiecare ieşire. Să considerăm exemplul de mai jos pentru o funcţie $F_{F_1,F_2} : Z_2 \times Z_2 \to Z_3 \times Z_2$:

Nr. Crt.	Intrare		Ieşire	
	x	y	F_1	F_2
1	0	0	0	0
2	0	1	0	1
3	2	-	1	-
4	1	2	-	1
5	În rest		2	-

Tabel iniţial

Nr. Crt.	Intrare		Ieşire
	x	y	F_1
1	0	0	0
2	0	1	0
3	2	-	1
4	1	2	-
5	În rest		2

Tabel cu prima ieşire

Nr. Crt.	Intrare		Ieşire
		y	F_2
1	0	0	0
2	0	1	1
3	2	-	-
4	1	2	1
5	În rest		-

Tabel cu a doua ieşire

Tab. 6-10 Multiplicarea pentru funcţia $F_{F_1,F_2} : Z_2 \times Z_2 \to Z_3 \times Z_2$

Eliminarea specificaţiilor nerelevante

Din tabelele obţinute se elimină, respectând regulile de mai jos, rândurile care au ieşirea DC .

Regula 1. Dacă în urma multiplicării tabelului de adevăr apar rânduri cu ieşirea DC acestea sunt eliminate doar dacă nu este specificată o valoare de ieşire implicită diferită de DC .

După cum se observă, din tabelul pentru ieşirea F_2 se elimină rândurile 3 şi 5. Pentru ieşirea F_1 se elimină rândul 4, însă rândul 5 trebuie expandat la tot domeniul de intrare nespecificat înainte de eliminarea rândului 4: $R_{5.1.}(1\ 0\ /\ 2)$, $R_{5.2.}(1\ 1\ /\ 2)$ şi $R_{5.3.}(0\ 2\ /\ 2)$. Rândul 5 va fi înlocuit cu rândurile 5.1, 5.2 şi 5.3. După această operaţie poate fi eliminat rândul 4.

Nr. Crt.	Intrare		Ieşire
	x	y	F_1
1	0	0	0
2	0	1	0
3	2	-	1
5.1.	1	0	2
5.2.	1	1	2
5.3	0	2	2

Tabel cu prima ieşire

Nr. Crt.	Intrare		Ieşire
	x	y	F_2
1	0	0	0
2	0	1	1
4	1	1	1

Tabel cu a doua ieşire

Tab. 6-11 Multiplicarea tabelului de intrare

Regula 2. Dacă un tabel obţinut în urma multiplicării tabelului de adevăr conţine un rând cu ieşire „don't care" şi există o valoare implicită pentru combinaţiile de intrare nespecificate atunci rândul este eliminat doar după generarea tuturor combinaţiilor de intrare pentru ieşirea implicită.

Gruparea specificaţiilor

Rândurile tabelului iniţial sunt grupate după valoarea de ieşire. Combinaţiile de intrare care produc aceeaşi ieşire formează o matrice. Valoarea de ieşire se consideră o matrice formată dintr-un singur vector care, la rândul lui, conţine o singură valoare. Grupa de valori este o pereche de matrice $G=(M_i,M_0)$. Valoarea de ieşire este conţinută de matricea M_o iar matricea M_i conţine toate combinaţiile de intrare care produc ieşirea respectivă.

Pentru un codificator din Z_3 în Z_5 cu trei variabile de intrare $f: D\times B\times C\to E\times F$, $D,C,B=Z_3=\{0,1,2\}$ şi $E,F=Z_5=\{0,1,2,3,4\}$ (Tab. 6-12)

Nr. Crt.	Intrare			Ieşire	
	D	*B*	*C*	*E*	*F*
1	0	0	0	0	0
2	0	0	1	0	1
3	0	0	2	0	2
4	0	1	0	0	3
5	0	1	1	0	4
6	0	1	2	1	0
7	0	2	0	1	1
8	0	2	1	1	2
9	0	2	2	1	3
10	1	0	0	1	4

Convertor „zecimal $Z_3 \rightarrow$ zecimal Z_5"

Nr. Crt.	Intrare			Ieşire
	D	*B*	*C*	*E*
1	0	0	0	0
2	0	0	1	0
3	0	0	2	0
4	0	1	0	0
5	0	1	1	0
6	0	1	2	1
7	0	2	0	1
8	0	2	1	1
9	0	2	2	1
10	1	0	0	1

Prima ieşire

Nr. Crt.	Intrare			Ieşire
	D	*B*	*C*	*F*
1	0	0	0	0
2	0	0	1	1
3	0	0	2	2
4	0	1	0	3
5	0	1	1	4
6	0	1	2	0
7	0	2	0	1
8	0	2	1	2
9	0	2	2	3
10	1	0	0	4

A doua ieşire

Separarea ieşirilor

Tab. 6-12 Convertor „zecimal $Z_3 \rightarrow$ zecimal Z_5"

gruparea rândurilor specificaţiilor obţinute în urma multiplicării este:

$$G_1 = \left(\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \quad (0) \right) \qquad G_1 = \left(\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 2 \end{pmatrix} \quad (0) \right)$$

$$G_2 = \left(\begin{pmatrix} 0 & 1 & 2 \\ 0 & 2 & 0 \\ 0 & 2 & 1 \\ 0 & 2 & 2 \\ 1 & 0 & 0 \end{pmatrix} \quad (1) \right) \qquad G_2 = \left(\begin{pmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \end{pmatrix} \quad (1) \right)$$

$$G_3 = \left(\begin{pmatrix} 0 & 0 & 2 \\ 0 & 2 & 1 \end{pmatrix} \quad (2) \right)$$

$$G_4 = \left(\begin{pmatrix} 0 & 1 & 0 \\ 0 & 2 & 2 \end{pmatrix} \quad (3) \right)$$

$$G_5 = \left(\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad (4) \right)$$

Grupe pentru prima ieşire Grupe pentru a doua ieşire

Tab. 6-13 Grupele pentru convertorului „zecimal $Z_3 \rightarrow$ zecimal Z_5"

Validarea şi metodele de corectare a grupelor de ieşire (COR)

Pentru a valida grupele de valori se analizează două câte două: dacă operaţia de intersecţie dintre matricele de intrare a două grupe nu are ca rezultat mulţimea vidă, atunci se vor modifica grupele respective alegând una din metodele:

COR 1. Eliminarea combinaţiilor comune din ambele grupe de valori în cazul în care sunt utilizate de proiectant doar pentru a reduce efortul de specificare. Se consideră că aceste combinaţii de intrare nu apar în timpul utilizării sistemului decizional şi ieşirea lor este **DC**.

COR 2. Crearea unei grupe noi de ieşire pentru combinaţiile de intrare comune dacă acestea activează simultan mai multe valenţe pe ieşire.

Dacă există astfel de combinaţii atunci funcţia este neunivocă.

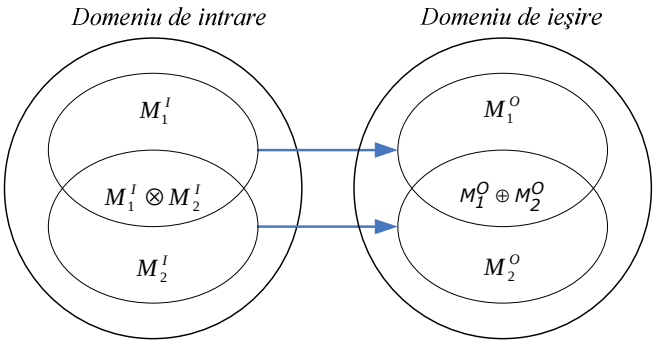


Fig. 6-16 Specificare neunivocă

Corectarea prin eliminarea combinaţiilor de intrare comune (COR 1)

Fie $G_1\left(M_1^I, M_1^O\right)$ şi $G_2\left(M_2^I, M_2^O\right)$.

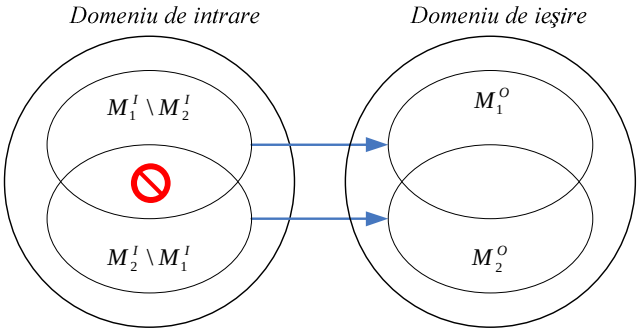


Fig. 6-17 Corectarea prin eliminarea combinaţiilor de intrare comune

Grupele G_1 şi G_2 sunt înlocuite cu grupele $G_1^{'}$ şi $G_2^{'}$:

$$\begin{cases} G_1\left(M_1^I, M_1^0\right) \\ G_1\left(M_2^I, M_2^0\right) \\ M_1^I \otimes M_2^I \neq \varphi \end{cases} \Rightarrow \begin{cases} G_1^{'}\left(M_1^I \setminus M_2^I, M_1^0\right) \\ G_2^{'}\left(M_2^I \setminus M_1^I, M_2^0\right) \end{cases}, \tag{6-5}$$

Transformarea pentru întregul tabel se face în felul următor:

Fie grupele numerotate de la 1 la g. Se compară fiecare grupă cu fiecare din grupele următoare. Dacă se determină o pereche de grupe pentru care intersecţia este diferită de mulţimea vidă atunci acestea sunt înlocuite cu $G_1^{'}$ şi $G_2^{'}$. Nu este necesară revenirea asupra testelor anterioare deoarece ele nu sunt afectate de această modificare.

Corectarea prin crearea unei noi grupe (COR 2)

Fie $G_1\left(M_1^I, M_1^O\right)$ şi $G_2\left(M_2^I, M_2^O\right)$ două grupe de ieşire.

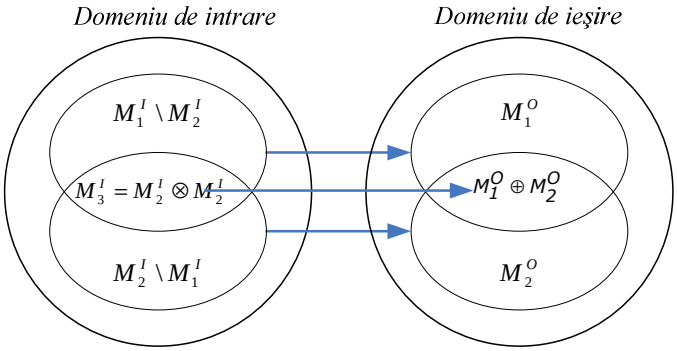


Fig. 6-18 Corectarea prin crearea unei grupe noi

Din cele două grupe se creează 3 grupe: $G_1^{'}$, $G_2^{'}$ şi G_{12}:

$$\begin{cases} G_1\left(M_1^I, M_1^0\right) \\ G_2\left(M_2^I, M_2^0\right) \\ M_1^I \otimes M_2^I \neq \varphi \end{cases} \Rightarrow \begin{cases} G_1^{'}\left(M_1^I \setminus M_2^I, M_1^0\right) \\ G_2^{'}\left(M_2^I \setminus M_1^I, M_2^0\right) \\ G_{12}\left(M_1^I \otimes M_2^I, M_1^0 \oplus M_2^0\right) \end{cases}, \qquad (6\text{-}6)$$

Transformarea pentru întregul tabel se face în felul următor:

Fie grupele numerotate de la 1 la g. Se compară fiecare grupă cu fiecare din grupele următoare. Dacă se determină o pereche de grupe pentru care intersecţia matricelor de intrare nu este vidă acestea sunt înlocuite cu grupele $G_1^{'}$, $G_2^{'}$ şi G_{12}. Este necesară o revenire asupra testelor anterioare deoarece este posibil să existe o altă grupă G cu ieşirea $M_1^0 \oplus M_2^0$. Într-o astfel de situaţie matricea G_{12} este reunită cu G.

În urma preprocesării se obţin structurile cu datele de intrare pentru algoritmul de determinare a implicanţilor. Pe lângă operaţiile prezentate a mai fost necesară implementarea unor analizoare lexicale, sintactice care să încarce datele. Preprocesarea reprezintă analiza semantică a acestora.

6.5.2 Extinderea matricei de acoperire pentru eliminarea redundanţelor

Un vector V dintr-o soluţie este redundant dacă mulţimea de vectori primi acoperiţi de soluţie nu se modifică prin eliminarea acestuia. O soluţie minimă nu conţine vectori redundanţi.

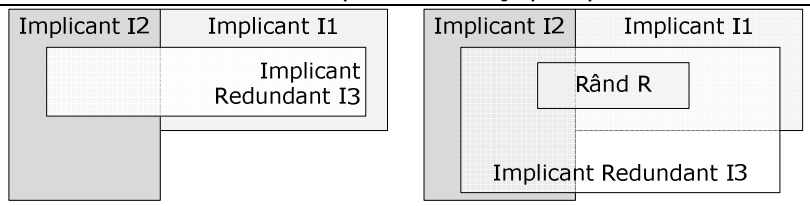


Fig. 6-19 Exemple de implicanţi redundanţi

În exemplul a) implicantul I_3 este acoperit de reuniunea vectorilor implicanţi I_1 şi I_2, deci este redundant. În exemplul b) implicant I_3 este redundant deoarece rândul R acoperit de el este acoperit de reuniunea implicanţilor I_1 şi I_2. Nu este necesar ca I_3 să fie acoperit integral de I_1 şi I_2. Cazul b) reprezintă cazul general. Dacă rândul R este acoperit numai de implicantul I_3 atunci rândul corespunzător acestuia în matricea de acoperire arată ca în Fig. 6-20.

Să presupunem că în afară de implicanţii I_1, I_2 şi I_3 toţi ceilalţi implicanţi nu se intersectează cu R. Valorile 0 din dreptul rândului R indică implicanţii care nu acoperă acest rând. Există două variante:

- implicantul nu intersectează rândul R;
- implicantul se intersectează cu rândul R, dar nu-l acoperă.

	I_1	I_2	I_3		
	...	...	...	...	...
R	0	0	0	1	0
	...	...	...	...	...

Fig. 6-20 Matrice de acoperire clasică

Această informaţie este deja calculată şi se află în mulţimile de atribute ale fiecărui implicant. Pentru a evidenţia aceste situaţii în matrice, valorile de 0 din dreptul unui vector implicant care intersectează şi nu acoperă rândul se înlocuiesc cu o valoare distinctă: -1.

	I_1	I_2	I_3		
	...	...	...	...	...
R	0	-1	-1	1	0
	...	...	...	...	...

Fig. 6-21 Matrice de acoperire extinsă

Se ridică problema afectării criteriilor de selectare a vectorilor implicanţi esenţiali, a raportului dintre rânduri şi a raportului dintre coloane. De asemenea, într-o soluţie deja formată pot exista vectori redundanţi deoarece rândurile acoperite de aceştia sunt acoperite şi de un alt implicant sau sunt acoperite printr-un cumul de implicanţi (implicanţii pe ale căror coloane apare valoarea -1 în dreptul rândurilor respective).

Modificarea algoritmului de selectare a implicanţilor esenţiali

Pentru matricea de acoperire extinsă algoritmul de determinare a implicanţilor esenţiali ia în considerare şi implicanţii care intersectează rândurile. Un implicant este esenţial în raport cu un rând dacă şi numai dacă reuniunea proiecţiilor celorlalţi implicanţi pe rândul respectiv formează o tautologie în raport cu rândul considerat.

Proiecţia unui implicant pe un rând se obţine prin eliminarea din implicant a tuturor poziţiilor în care rândul este definit (are valori diferite de DC). Proiecţiile sunt memorate într-o matrice şi se verifică dacă matricea este tautologie.

În Fig. 6-22 sunt notate zonele distincte din Fig. 6-19 b) cu litere mari de la A la G.

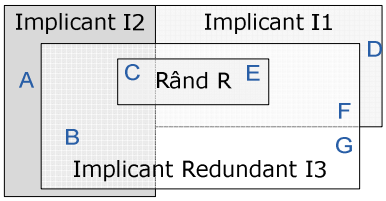


Fig. 6-22 Marcarea zonelor distincte

Proiecţiile implicanţilor I_1, I_2 şi I_3 pe rândul R sunt E, C şi respectiv $C \cup E$. Înlocuind în matricea de acoperire valorile 0, -1 şi 1 cu mulţimile corespunzătoare se obţine matricea de mai jos.

	I_1	I_2	I_3	
	...	...	...	...	...
R	ϕ	E	C	$R = C \cup E$	ϕ
	...	...	...	...	...

Fig. 6-23 Matricea de acoperire reprezentată prin intersecţii

Matricea proiecţiilor vectorilor I_1 şi I_2 pe rândul R este:

	R
I_1	E
I_2	C

Fig. 6-24 Matricea proiecţiilor

Matricea proiecţiilor este tautologie (**TAU**)[211][212][313] deoarece $C \cup E = R$.

Implicantul I_3 este redundant deoarece acoperirea rândului R se poate face şi prin utilizarea perechii de vectori (I_1, I_2) deoarece $(I_1 \cap R) = (I_1 \cap R) \cup (I_2 \cap R)$.

Etapele algoritmului de verificare a tautologiei (notate cu **TAU n**) sunt:

TAU 1: Dacă există un rând în matrice care conţine numai valoarea DC atunci matricea este tautologie şi algoritmul se termină.

TAU 2: Dacă matricea este formată dintr-o singură coloană şi apar toate valenţele domeniului corespunzător coloanei atunci este tautologie şi algoritmul se termină.

TAU 3: Dacă matricea este formată dintr-o singură coloană şi nu apar toate valenţele domeniului corespunzător coloanei atunci nu este tautologie şi algoritmul se termină.

TAU 4: Se alege o coloană care are număr maxim de valori diferite de DC. Pentru fiecare valenţă v din domeniul de valori al coloanei respective se creează câte o matrice care conţine numai rândurile în care, pe coloana selectată, apare valenţa v sau DC. Din matricele noi se elimină coloana după care s-a făcut descompunerea.

TAU 5: Se verifică dacă fiecare din matricele obţinute în **TAU 4** sunt tautologii. Dacă una dintre aceste matrice nu este tautologie atunci matricea iniţială nu este tautologie.

Aplicarea algoritmului de verificare a tautologiei pentru matricea M din Tab. 6-14 conduce la rezultatul "M este tautologie".

#	$C_1 \in \{0,1\}$	$C_2 \in \{0,1\}$	$C_3 \in \{0,1\}$	$C_4 \in \{0,1\}$
1	-	-	0	1
2	-	-	1	0
3	-	0	0	-
4	-	1	-	0
5	0	-	-	-
6	1	-	1	1

Tab. 6-14 Matrice M pentru care se verifică proprietatea de tautologie

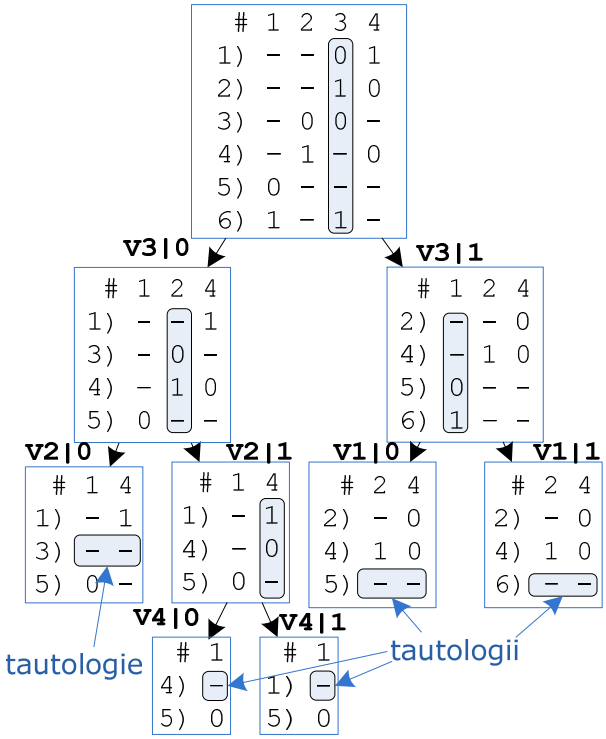


Fig. 6-25 Aplicarea algoritmului de verificare a tautologiei

Algoritmul pentru determinarea implicanţilor esenţiali este identic în partea de început cu cel deja prezentat, tratând valoarea -1 ca fiind 0. Apare o decizie suplimentară dată de verificarea redundanţei implicanţilor esenţiali. Pentru un implicant s-a luat decizia că este esenţial în raport cu cel puţin un rând. Proiecţiile pe acest rând a implicanţilor încă neselectaţi care intersectează rândul respectiv (indicaţi de prezenţa valorii -1) trebuie să nu formeze o tautologie.

Adăugarea la soluţie a implicanţilor esenţiali atrage după sine eliminarea lor precum şi a rândurilor acoperite de aceştia din matricea rămasă. În schimb, pentru fiecare rând rămas doar intersectat de implicantul esenţial se memorează partea acestuia acoperită de implicantul eliminat. Ea va fi folosită în procesul de verificare a tautologiei. La proiecţiile implicanţilor rămaşi, utilizaţi în construirea matricei pentru verificarea tautologiei, se adaugă şi proiecţiile implicanţilor deja selectaţi în soluţie.

6.6 Concluzii

În acest capitol am prezentat mai multe metode de minimizare distincte concepute cu scopul de a testa diverse tehnici de prelucrare a specificaţiilor tabelare. Implementarea primelor două metode a avut ca efect obţinerea a doi algoritmi de minimizare exacţi dar cu performanţe

relativ scăzute. Observarea comportamentului celor două implementări a condus la crearea celei de a treia metode care abordează complet diferit problema minimizării.

Metoda determinării implicanţilor maximali cu număr minim de poziţii specificate s-a dovedit viabilă peste tot spectrul de probleme de minimizare. Primele două metode, chiar dacă au dat rezultate foarte bune pentru anumite clase de probleme, nu s-au dovedit a fi satisfăcătoare în anumite cazuri. Cea de a treia metodă este o combinaţie de algoritmi şi structuri de date dezvoltaţi în primele două implementări. Principalele caracteristici ale acestei metode sunt:

- eliminarea redundanţele de procesare observate în primele două metode
- crearea unui „pachet" de informaţii asociat fiecărui vector obţinut în timpul procesării util în determinarea cât mai timpurie a vectorilor care compun soluţia şi a vectorilor care nu mai pot influenţa rezultatul (numiţi, în algoritm, neviabili).

Testarea metodelor a fost efectuată numai pentru sisteme multivalente deterministe.

Metoda selectată cuprinde următoarele etape:

- **DESC** - **descompunerea** primară nedisjunctă a vectorului total, neafectată de o ordine prestabilită a variabilelor de intrare, în vectori cu o singură poziţie specificată;
- **crearea unui set de atribute** asociat fiecărui vector obţinut în cadrul operaţiei de descompunere. Atributele sunt determinate direct din tabelul iniţial o singură dată. În cadrul evoluţiei algoritmului se vor determina noi vectori pentru care atributele se calculează pe baza celor existente în vectorii mai vechi;
- pornind de la setul de vectori din descompunerea primară şi atributele acestora se determină vectori cu din ce în ce mai multe poziţii specificate şi atributele acestora. Pe baza atributelor noilor vectori aceştia sunt clasificaţi, determinându-se **vectori implicanţi** până când toate rândurile din tabelul iniţial sunt acoperite de aceştia;
- din mulţimea de vectori implicanţi, pentru a forma o **soluţie**, este construită matricea de acoperire şi sunt extraşi cei **esenţiali**;
- se determină **nucleele ciclice** independente de vectori implicanţi;
- **nucleele ciclice** sunt **rezolvate** utilizând un algoritm de tip **Greedy**;
- selectarea unui implicant în soluţie este dublată de **operaţii** anexe **de construire a proiecţiilor** acestuia **asupra rândurilor** încă neacoperite din tabelul iniţial;
- rândurile ale căror **matrice de proiecţie** au devenit **tautologii** sunt eliminate pentru a evita adăugarea de **implicanţi redundanţi** în soluţie.

Metoda lui Thelen, singura integrată în algoritm dintre cele existente, este utilizată într-o formă extinsă pentru a permite detectarea

redundanţelor. Matricea de acoperire a fost definită de Thelen ca având numai 2 valori: 0 sau 1. Valoarea 0 a fost păstrată, dar valoarea 0 a fost înlocuită cu valori distincte pentru cele două cazuri care o produc: 0 dacă un rând nu se intersectează cu un implicant şi -1 dacă un implicant intersectează un rând fără să-l acopere.

Prelucrările prezentate în acest capitol sunt doar parţial adecvate specificaţiilor nedeterministe. Ele vor suferi modificări substanţiale pentru tratarea specificaţiilor nedeterministe.

Toate metodele prezentate au în comun o parte de preprocesare pentru validarea şi eventual pregătirea datelor de intrare şi o parte de postprocesare pentru eliminarea redundanţelor şi a hazardului.

7 Principiile şi strategiile de minimizare propuse

Din analiza metodelor şi tehnicilor de minimizare elaborate şi prezentate în capitolul anterior am extras:

- un set de principii care stă la baza operaţiilor de minimizare şi
- un set de strategii ce se aplică în funcţie de
  - principiiile utilizate şi
  - semantica atribuită tabelului de valori.

Fiecare dintre metodele şi tehnicile utilizate subscriu unui subset de principii şi a fost implementată utilizând una sau mai multe strategii. Combinarea dintre principii şi strategii (nu neapărat un singur principiu şi o singură strategie) conduce la elaborarea de diverse metode de minimizare.

O parte din acestea pot fi aplicate simultan, dar unele sunt incompatibile. În schimb se pot utiliza două principii sau strategii incompatibile dacă sunt înglobate într-o metodă secvenţială, care nu le aplică simultan.

În acest capitol sunt prezentate distinct fiecare dintre principiile şi strategiile deduse cu scopul de a evidenţia diferenţele dintre acestea şi de a urmări traducerea lor în tehnici implementate în aplicaţia finală.

7.1 Metrica

Metrica utilizată în evaluarea calităţii unui tabel de valori – a datelor de intrare şi a datelor de ieşire – este definită prin numărul de literale utilizate şi, opţional, prin numărul de rânduri [214][215][216][217]. Procesul de minimizare are ca scop reducerea numărului de literale şi rânduri din tabelul primit ca intrare construind un nou tabel de valori, echivalent, care păstrează integral sau parţial, după cum s-a optat, specificaţiile iniţiale.

7.2 Principiile minimizării

Am dedus şase principii de minimizare (Fig. 7-1) care acoperă metoda implementată. Dintre acestea numai principiului discriminării (P1) a fost preluat din metoda discriminării. Utilizarea principiului discriminării precum şi celelalte cinci principii sunt contribuţii personale.

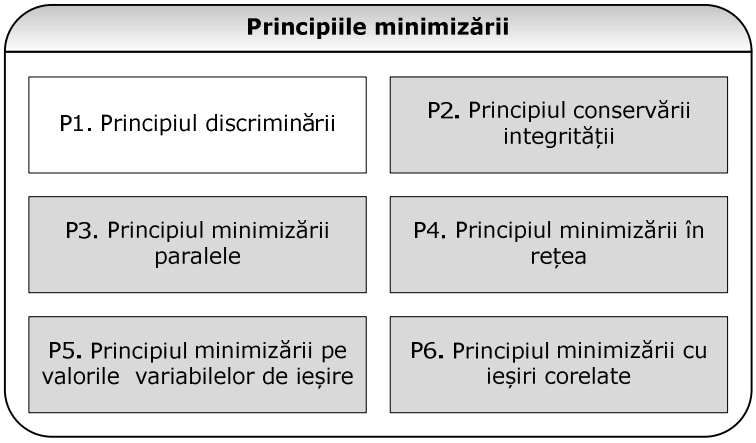


Fig. 7-1 Principii de minimizare propuse

7.2.1 Principiul discriminării

Principiul discriminării [14], în contextul minimizării sistemelor decizionale, constă în determinarea elementelor care diferenţiază combinaţiile de intrare cu ieşiri diferite.

Două combinaţii de intrare sunt **discriminate** (nu au zone comune) dacă pe cel puţin o poziţie au valori disjuncte. Conform acestui principiu, regula urmărită este ca în fiecare rând al tabelului de valori rezultat să se păstreze în partea de intrare numărul minim de poziţii specificate astfel încât rândul să fie totuşi diferit, **discriminat** [14], [177] în raport cu alte rânduri ale tabelului cărora le corespund valori de ieşire diferite faţă de rândul considerat.

Principiul discriminării diferenţiază această tehnica faţă de toate celelalte [218], şi permite în majoritatea cazurilor rezultate mai bune, în special în minimizarea tabelelor cu ieşiri multivalente şi multiple şi în cazul minimizărilor în reţea a tabelelor deterministe şi nedeterministe.

Această metodă de abordare a minimizării nu ia în considerare în mod explicit combinaţiile de intrare nefolosite. Zona nespecificată devine cu adevărat „**don't care**".

Principiul discriminării în raport cu principiile minimizării „clasice"

Toate metodele urmăresc aceeaşi metrică: cât mai puţine literale şi cât mai puţine rânduri. Metodele clasice minimizează în cadrul aceleiaşi valori a variabilei de ieşire (pe grupe valorice). Diferenţa dintre cele două „stiluri" poate fi exprimată astfel: metodele clasice încearcă să construiască „grupuri" cât mai mari de combinaţii de intrare cu aceeaşi ieşire – care se exprimă prin implicanţi cu cât mai puţine poziţii specificate - utilizând spaţiul liber de tip „**don't care**" disponibil, în timp ce aplicarea principiului

discriminării implică decompunerea spaţiului urmărind disjuncţiile dintre combinaţiile de intrare.

Cu alte cuvinte:

- metodele clasice obţin numărul minim de literale în implicanţi prin folosirea directă a potenţialului din DC pentru fiecare valoare de ieşire în parte;
- metoda discriminării obţine numărul minim de literale în implicanţi prin discriminarea faţă de rândurile celorlalte valori.

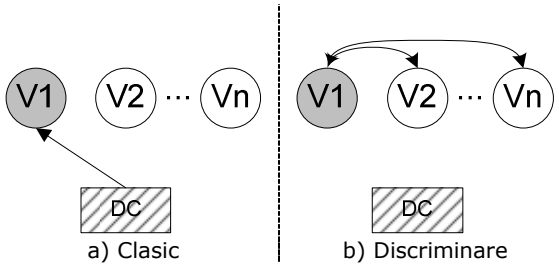


Fig. 7-2 Minimizarea rândurilor care determină valoarea de ieşire V_1

7.2.2 Principiul conservării integrităţii funcţionale a specificaţiei iniţiale

Rezultatul minimizării (tabelul minimizat) trebuie să conserve toate corespondenţele intrare/ieşire din specificaţia iniţială. Când o combinaţie de intrare din specificaţia iniţială este aplicată tabelului minimizat trebuie să se obţină aceleaşi valori de ieşire ca şi în tabelul iniţial, chiar dacă acesta este neunivoc determinist.

Metodele clasice nu permit păstrarea integrităţii funcţionale pentru funcţiile neunivoce deterministe. Acest rezultat poate fi obţinut prin artificii la nivel de reprezentare al tabelului cum ar fi tratarea distinctă a fiecărei valenţe de ieşire.

Pentru conservarea integrităţii funcţionale am conceput o metodă de minimizare care permite prezenţa pe ieşire a tuturor valenţelor specificate ca ieşiri în tabelul iniţial.

Am adăugat o opţiune de minimizare, pentru specificaţii neunivoce nedeterminist, care permite selectarea automată a unei submulţimi optime a mulţimii valorilor de ieşire. Opţiunea poate fi considerată ca o specificare de proiect, în caz contrar se aplică principiul conservării integrităţii.

7.2.3 Principiul minimizării paralele

Pentru specificările cu ieşiri multiple am enunţat principiul minimizării paralele care înseamnă minimizarea independentă a fiecărei

ieşiri. Pentru fiecare variabilă de ieşire se creează câte un tabel care include toate coloanele de intrare şi doar coloana de ieşire corespunzătoare variabile. Fiecare tabel este minimizat separat.

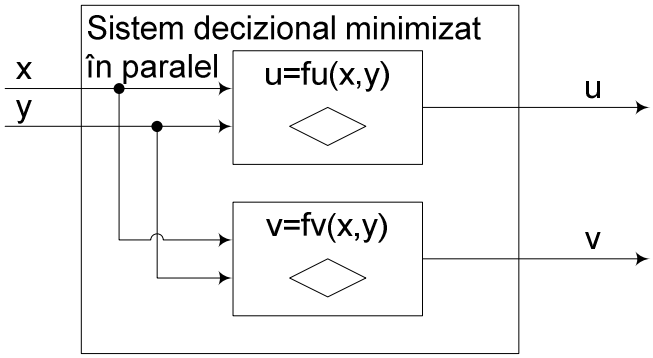


Fig. 7-3 Organigrama principiului de minimizare în paralel

```
.func f x y / u v      .func f_PM_u x y / u   .func f_PM_v x y / v
//x y / u v           //x y / u              //x y / v
0 0 / 0 0             0 - / 0                0 1 / 1
0 1 / 0 1             - 0 / 0                0 0 / 0
1 0 / 0 1             1 1 / 1                1 0 / 1
1 1 / 1 0                                    1 1 / 0
```

Tab. 7-1 Tabelele minimizate în paralel

Rândurile care au valoarea $(011/3)$ pe ieşire sunt eliminate deoarece nu influenţează procesul de minimizare.

În urma operaţiei de scindare a tabelului iniţial în specificaţii separate pentru fiecare variabilă de ieşire se pierd eventualele corelaţii dintre valorile variabilelor de ieşire.

7.2.4 Principiul minimizării în reţea

Minimizarea în reţea constă în utilizarea unor variabile de ieşire, alese în baza unor anumite criterii, ca variabile de intrare pentru calcularea celorlalte variabile de ieşire.

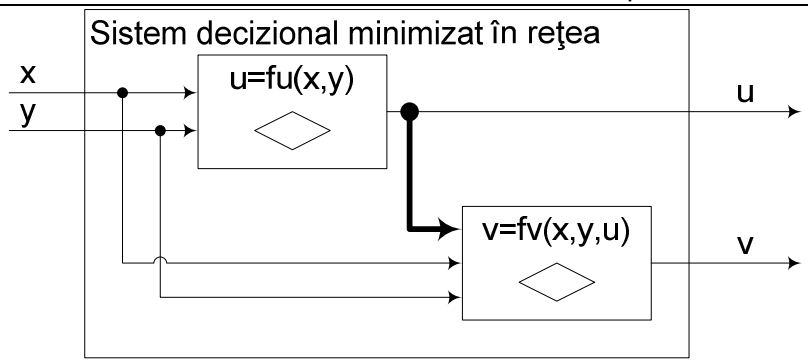


Fig. 7-4 Organigrama principiului de minimizare în paralel

```
.func f x y / u v      .func f_NM_u x y / u    .func f_NM_v x y u / v
//x y / u v           //x y / u               //x y u / v
0 0 / 0 0             0 - / 0                 0 0 - / 0
0 1 / 0 1             - 0 / 0                 - - 1 / 0
1 0 / 0 1             1 1 / 1                 0 1 - / 1
1 1 / 1 0                                     1 0 - / 1
```

Tab. 7-2 Tabelele minimizate în reţea

Minimizarea în reţea [219] se poate aplica atât sistemelor cu mai multe ieşiri, ca în figura de mai sus, cât şi sistemelor cu o singură ieşire dacă sunt expandate pe valenţele variabilei de ieşire (principiul minimizării pe valenţele variabilelor de ieşire) sau mixt pentru sisteme cu mai multe ieşiri expandate.

O asemenea abordare permite:

- minimizarea în reţea chiar şi a tabelelor care au numai o singură ieşire;
- minimizarea în reţea a tabelelor multivalente nedeterministe;
- păstrarea corelaţiilor dintre diferite ieşiri (dacă acest aspect este dorit), sau dintre diferitele grupuri de ieşiri (cazul specificărilor vectoriale) dintr-o reţea nedeterministă; de exemplu, unui rând de intrare dintr-un tabel având (q_1q_2) variabile de ieşire, îi corespund 3 seturi de valori de ieşire: f_2, $\{3,0\}$, b şi se doreşte ca numai aceste combinaţii să apară pe ieşire.

Programul oferă opţiunea de a minimiza tabelul nedeterminist, considerând ieşirile cuplate sau independente.

7.2.5 Principiul minimizării pe valorile variabilelor de ieşire

Rezultatul minimizării unui sistem decizional multivalent produce decizii pentru activarea sau blocarea fiecărei valori de ieşire în parte. Astfel, ieşirile minimizate sunt de fapt ieşiri valorice (ieşiri pe valoare).

```
a b c / x
3 5 7 / 9
1 7 2 / 7
5 7 2 / 7
2 - 4 / 3
4 - 7 / 0
3 7 7 / 0
- 5 4 / 0
```

Tab. 7-3 Tabel iniţial

```
a b c / x(0) x(3) x(7) x(9) x(5)
3 5 7 / 0 0 0 1 0
1 7 2 / 0 0 1 0 0
5 7 2 / 0 0 1 0 0
2 - 4 / 0 1 0 0 0
4 - 7 / 1 0 0 0 0
3 7 7 / 1 0 0 0 0
- 5 4 / 1 0 0 0 0
```

Tab. 7-4 Tabel expandat pe valorile variabilelor de ieşire

Această abordare generează soluţii mai bune atât în minimizarea în reţea a funcţiilor cu variabile multiple de ieşire, cât şi în cazul funcţiilor cu o singură ieşire multivalentă, întrucât variabila expandată pe valori poate fi minimizată în reţea.

Expandarea variabilelor de ieşire pe valorile lor permite atât tabelelor deterministe cât şi tabelelor nedeterministe cu univocităţi explicite (unui rând de intrare îi corespunde o mulţime de valori de ieşire în loc de o singură valoare) să se obţină o mai bună minimizare în reţea.

7.2.6 Principiul minimizării cu ieşiri corelate

Specificaţiile cu ieşiri multiple pot fi minimizate în reţea considerând posibilele corelaţii ce se pot evidenţia între diferitele valori ale variabilelor de ieşire. În exemplu de mai jos pentru combinaţia de intrare (011) sunt posibile câte două valori pentru fiecare variabilă de ieşire: 1 şi 2 pentru a şi 3 şi 4 pentru b, dar atunci când a primeşte valoarea 2, b nu mai poate avea decât valoarea 3. Minimizarea se poate face fără corelarea valorilor de ieşire şi atunci pentru intrarea $(011/\{1,2\})$ sunt posibile patru combinaţii de ieşire, sau cu corelare, şi atunci pentru intrarea (011) sunt posibile trei combinaţii de ieşire.

```
.library vectorned
.variable x 0 1
.variable y 0 1
.variable z 0 1
.variable a 0 1 2
.variable b 2 3 4
```

```
.function f x y z / a b
//x y z / a b
0 0 0 / 1 2
0 1 1 / 1 {3, 4}
0 1 1 / 2 3
1 0 0 / 2 2
1 1 1 / 0 -
```

```
.function f_CNM_a_sol x y b / a
//x y b / a
0 - - / 1
0 - 3 / {1, 2}
1 0 - / 2
/ 0
```

```
.function f_NM_a_sol x y z / a
//x y z / a
0 - - / 1
0 - 1 / {1, 2}
1 0 - / 2
/ 0
```

```
.function f_CNM _b_sol y / b
//y / b
1 / {3, 4}
/ 2
```

Minimizare cu corelarea ieşirilor

```
.function f_NM_b_sol y / b
//y / b
1 / {3, 4}
/ 2
```

Minimizare fără corelarea ieşirilor

Tab. 7-5 Evidenţierea opţiunii de corelare a ieşirilor în minimizare (vezi ieşirea a)

Minimizarea cu corelarea ieşirilor are sens doar pentru minimizarea în reţea. În cazul minimizării paralele nu se poate conserva corelaţia dintre ieşiri.

7.3 Opţiuni şi strategii de minimizare

Procesul de minimizare se poate efectua urmând diverse strategii în funcţie de principiile de minimizare alese şi de tipul rezultatului dorit. Fiecare strategie în parte poate fi „afectată" de opţiunile care modifică modul în care sunt interpretate informaţiile din tabelul de valori. Strategiile şi opţiunile prezentate în continuare (Fig. 7-5) respectă principiile cu care sunt compatibile şi creează o imagine completă asupra metodei dezvoltate în această lucrare.

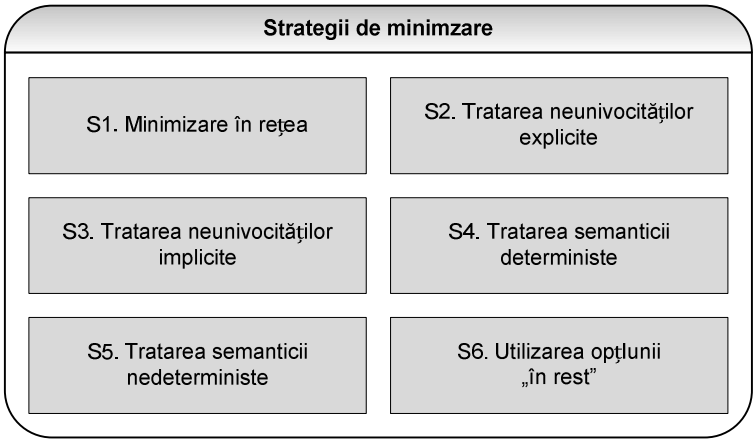


Fig. 7-5 Strategii de minimizare propuse

7.3.1 Strategii de minimizare în reţea

Ideea generală a minimizării în reţea constă în folosirea variabilelor de ieşire, ca intrări pentru celelalte variabile de ieşire, pentru ca, împreună cu informaţia intrărilor iniţiale ale nodurilor, să determine expresii minimizate mai simple. Acest efect se bazează pe faptul că o parte a informaţiei este deja prelucrată în variabilele de ieşire minimizate anterior.

Strategia de minimizare în reţea [219][220] se poate aplica sistemelor binare sau multivalente cu ieşiri multiple, sau sistemelor având doar o singură ieşire multivalentă, expandând tabelul pe valenţele ieşirilor.

```
.f func x y z / a b
//x y z / a b
0 0 0 / 1 2
0 1 1 / 1 {3, 4}
0 1 1 / 2 3
1 0 0 / 2 2
1 1 1 / 0 -
```

```
.fun_1 y / b
//y / b
1 / {3, 4}
/ 2
```

```
.f func_2 x y b / a
//x y b / a
0 - - / 1
0 - 3 / {1, 2}
1 0 - / 2
/ 0
```

Tab. 7-6 Minimizare în reţea

```
.function f x y z / a b
//x y z / a b
0 0 0 / 1 2
0 1 1 / 1 {3, 4}
0 1 1 / 2 3
1 0 0 / 2 2
1 1 1 / 0 –
```

```
.function f_BE x y z / a0 a1 a2 b2 b3 b4
//x y z / a(0) a(1) a(2) b(2) b(3) b(4)
0 0 0 / 0 1 0 1 0 0
0 1 1 / 0 1 0 0 1 1
0 1 1 / 0 0 1 0 1 0
1 0 0 / 0 0 1 1 0 0
1 1 1 / 1 0 0 0 0 0
```

Tab. 7-7 Expandare pe valenţele valorilor de ieşire

```
//y / b(2)
0 / 1
/ 0
```

```
//x y / a(0)
1 1 / 1
/ 0
```

```
//x y / b(3)
0 1 / 1
/ 0
```

```
//x y / a(1)
- 1 / 0
1 - / 0
0 - / 1
```

```
//a(1) a(0) / a(2)
0 0 / 1
/ 0
```

```
//a(2) b(3) / b(4)
0 1 / 1
/ 0
```

Tab. 7-8 Minimizare în reţea pe valenţele valorilor de ieşire

Pentru strategia de minimizare în reţea nu se cunosc decât rezultatele. În ceea ce priveşte prezenta metodă, minimizarea în reţea se face astfel: după o minimizare separată a fiecărei ieşiri (minimizare paralelă), ieşirea care are cea mai simplă expresie este păstrată ca parte a soluţiei finale şi folosită ca intrare suplimentară pentru restul variabilelor de ieşire. Se efectuează încă o minimizare în paralel şi se alege o nouă variabilă de ieşire, care se foloseşte şi ea ca intrare pentru minimizarea restului de variabile ş.a.m.d., până când toate ieşirile sunt minimizate;

Metoda se aplică oricărui tip de tabel de valori cu mai multe ieşiri.

În cazul ieşirilor multiple sau singulare (cunoscut fiind faptul că ieşirile reprezintă rezultatul unor decizii logice care stabilesc dacă o anumită valoare a ieşirii este sau nu livrată ca ieşire a variabilei), fiecare nod poate fi desfăcut într-un set de variabile corespunzătoare valorilor fiecărei variabile de ieşire. Rezultatul este un tabel generat, având ieşiri binare - fiecare apariţie a unei anumite valori pe ieşirea variabilei este marcată prin 1 în tabela expandată şi prin 0 când este blocată. Tabelul rezultat prin

expandare poate fi minimizat în reţea, în conformitate cu metoda expusă mai sus.

7.3.2 Minimizarea sistemelor cu neunivocitate explicită

Neunivocitatea explicită evidenţiază situaţia unui rând dintr-un tabel de valori căruia îi corespunde ca ieşire o mulţime de valori din spaţiul ieşirilor.

Minimizarea pentru această situaţie trebuie să respecte principiul conservării integrităţii funcţionale (conservarea specificaţiei iniţiale), deoarece în implementare pot să apară următoarele opţiuni:

- folosirea simultană a valorilor în vederea iniţierii unor prelucrări paralele;
- transmiterea simultană a mai multor valori pe un singur canal, de exemplu, diferite frecvenţe pe un canal optic, sau prin radio, etc.

În cazul în care neunivocităţile explicite reprezintă un nedeterminism real, oferindu-se posibilitatea alegerii unei anumite valori din mulţimea de ieşiri corespunzătoare unui anumit vector de intrare, atunci se poate avea în vedere o anumită funcţie de selecţie, cu condiţia ca această cerinţă să fie adăugată, ca specificaţie, la tabelul de valori iniţial.

```
.library vectorned
.variable x 0 1 2
.variable y 0 1 2
.variable z 0 1 2
.variable u 0 1 2

.function f x y z / u
//x y z / u
0 0 0 / ~
0 1 0 / {0, 1}
0 0 1 / {0, 2}

.function f_n/n_PM_u y z / u
//y z / u
0 - / {0, 2}
- 0 / {0, 1}
```

Tab. 7-9 Specificare cu neunivocităţi explicite

Programul include o opţiune suplimentară de selectare optimă a unei submulţimi din totalul de valori de ieşire corespunzătoare unui rând de intrare.

```
.library vectorned
.variable x 0 1 2
.variable y 0 1 2
.variable z 0 1 2
```

```
.variable u 0 1 2

.function f x y z / u
//x y z / u
0 0 0 / ~
0 1 0 / {0, 1}
0 0 1 / {0, 2}

.function f_k/n_PM_u  / u
/ 0
```

Tab. 7-10 Minimizarea unei specificări cu neunivocităţi explicite

7.3.3 Minimizarea sistemelor cu neunivocitate implicită

O neunivocitate implicită este evidenţiată de rânduri cu ieşiri diferite pentru care se intersectează combinaţiile din spaţiul de intrare.

Fiecare tabel nedeterminist având neunivocităţi implicite este convertit într-un tabel echivalent, dar având numai neunivocităţi explicite:

Exemplu:

```
.f func x y / z
// x y / z
1 - / 0
- 1 / 1
```

Tab. 7-11 Specificare cu neunivocităţi implicite

care se transformă în tabelul echivalent cu neunivocităţi explicite:

```
.f func x y / z
// x y / z
1 0 / 0
0 1 / 1
1 1 / {0,1}
```

Tab. 7-12 Minimizarea unei specificări cu neunivocităţi implicite

7.3.4 Neunivocitate vs. Nedeterminism

Diferenţa dintre neunivocitate şi nedeterminism nu se manifestă la nivel de reprezentare în tabelul de valori. Această diferenţă se manifestă doar la nivel de interpretare a tabelului. Afirmaţia: „Combinaţia de intrare $(2,1,-)$ are ca valoare de ieşire mulţimea $(011/2)$”. Poate fi interpretată în două feluri:

```
.f func x y z / a
//x y z / a
……….
2 1 - / {0,2}
……….
```

Tab. 7-13 Specificare cu nedeterminism

- neunivoc: pentru combinaţia de intrare $(2,1,-)$ se activează valorile de ieşire 0 şi 2 – acolo unde este posibil.
- nedeterminist: pentru combinaţia de intrare $(011/1)$ se activează valoarea de ieşire 0 sau 2, fără să se poată preciza din start care anume.

Neunivocitatea nu reprezintă un nedeterminism. Pentru exemplu de mai sus, combinaţia de intrare $(2,1,-)$ va activa ambele valori pe ieşire 0 şi 2, ceea ce reprezintă o situaţie deterministă, fără posibilitatea de a opta între mai multe variante.

7.3.5 Minimizarea sistemelor neunivoce cu semantică deterministă (n / n sau Full)

Minimizarea tabelelor neunivoce cu semantică deterministă conduce la un tabel care păstrează integral funcţionalitatea tabelului iniţial, şi anume: prin aplicarea fiecărei combinaţii de intrare din tabelul iniţial în tabelul minimizat se obţin aceleaşi valori de ieşire. Cu alte cuvinte, mulţimea valorilor de ieşire corespunzătoare unei combinaţii de intrare se regăseşte integral – explicit sau implicit – în tabelul minimizat.

7.3.6 Minimizarea sistemelor neunivoce cu semantică nedeterministă (k/n)

Minimizarea tabelelor neunivoce cu semantică nedeterministă conduce la un tabel care nu păstrează integral funcţionalitatea tabelului iniţial, şi anume: prin aplicarea fiecărei combinaţii de intrare din tabelul iniţial în tabelul minimizat se obţin submulţimi ne vide ale mulţimii de valori de ieşire corespunzătoare din tabelul iniţial. Cu alte cuvinte, mulţimea valorilor de ieşire corespunzătoare unei combinaţii de intrare nu se mai regăseşte întotdeauna integral – explicit sau implicit – în tabelul minimizat.

În exemplul de mai jos sunt prezentate comparativ rezultatele minimizării *n / n* şi *k / n* :

```
 .library vectorned
.variable x 0 1 2
.variable y 0 1 2
.variable z 0 1 2
.variable u 0 1 2
```

```
.function f x y z / u
//x y z / u
0 0 0 / ~
0 1 0 / {0, 1}
0 0 1 / {0, 2}
```

```
.function f_n/n_PM_u y z / u
//y z / u
0 - / {0, 2}
- 0 / {0, 1}
```

Minimizare cu opţiunea f_3

```
.function f_k/n_PM_u  / u
/ 0
```

Minimizare cu opţiune k / n

Tab. 7-14 Comparaţie între minimizarea f_2 şi k / n

7.3.7 Utilizarea opţiunii „în rest" cu valoarea „default"

Utilizarea opţiunii „**în rest**" se face alegând - conform metricii - cea mai slabă grupă de rânduri cu aceeaşi valoare de ieşire. Această grupă este înlocuită prin „**în rest**" având ca valoare „**default**" valoarea de ieşire a grupei. Implicit în această declaraţie logică sunt asimilate şi combinaţiile de intrare nespecificate.

Tabelele cu nedeterminism implicit pot utiliza ca valoare „**default**" doar mulţimi de valori care nu se regăsesc total sau parţial în alte valori de ieşire (mulţimi de valori de ieşire diferite) corespunzătoare rândurilor rămase în urma utilizării acestei opţiuni.

Dacă condiţia de mai înainte nu este îndeplinită pentru nici o valoare, atunci tabelul nu poate conţine o valoare „**default**", aşa cum este exemplificat în tabelul de mai jos:

```
x y / y
1 - / 0
- 1 / 1
```

Tab. 7-15 Specificare incompletă

O asemenea condiţie se ia automat în considerare în cazul tabelelor nedeterministe.

Aşa cum s-a precizat mai înainte, este posibil ca enunţul unui tabel nedeterminist de valori să aibă precizată o valoare „**default**" care să nu îndeplinească condiţia de mai sus. Într-un asemenea caz, programul de minimizare detectează neconformitatea, ţine cont de neunivocităţi şi produce soluţia corectă, în care, ca opţiune, se poate prezenta orice rezultat cu valoare „**default**", dacă condiţia menţionată este îndeplinită.

`În tabelul de mai jos sunt prezentate comparativ rezultatele minimizării cu şi fără opţiunea „**în rest**".

```
.library vectorned
.variable x 0 1
.variable y 0 1
.variable u 0 1
.variable v 0 1
```

```
.function f x y / u v
//x y / u v
0 0 / 0 0
0 1 / 0 1
1 0 / 0 1
1 1 / 1 0
```

Minimizare fără opţiunea „**în rest**”	Minimizare cu opţiunea „**în rest**”

`.function f_k/n_NM_u x y / u`
`//x y / u`
`0 - / 0`
`- 0 / 0`
`1 1 / 1`
	`.function f_k/n_NM_u_sol x y / u`
	`//x y / u`
	`1 1 / 1`
	`/ 0`

`.function f_k/n_NM_v x y u / v`
`//x y u / v`
`0 0 - / 0`
`- - 1 / 0`
`0 1 - / 1`
`1 0 - / 1`
	`.function f_k/n_NM_v_sol x y u / v`
	`//x y u / v`
	`0 0 - / 0`
	`- - 1 / 0`
	`/ 1`

Tab. 7-16 Comparaţie între minimizarea cu şi fără opţiunea „**în rest**”

7.4 Concluzii

Pentru procesul de minimizare am propus urmatorul set de de principii ce trebuie urmat :

- principiul discriminării;
- principiul conservării integrităţii funcţionale;
- principiul minimizării paralele;
- principiul minimizării în reţea;
- principiul minimizării pe valorile variabilelor de ieşire;
- principiul minimizării cu ieşiri corelate.

Principiul **discriminării** se aplică întotdeauna.

Principiul **conservării integrităţii funcţionale** îşi produce efectul ca o opţiune de minimizare care impune păstrarea integrală a tuturor valorilor de ieşire pentru fiecare combinaţie de intrare specificată iniţial. Metodele clasice tratează sistemele neunivoce ca sisteme nedeterministe. Am introdus acest principiu deoarece într-un sistem natural multivalent (cum ar fi transmisia de date pe fibră optică) se poate cere simultan emiterea mai multor valenţe şi nu a uneia singură dintr-un set dat. Acest principiu dă posibilitatea minimizării unor astfel de sisteme fără o conversie prealabilă a specificaţiei iniţiale într-o formă care să permită rezultate neunivoce deterministe.

Principiul **minimizării paralele** se aplică sistemelor cu mai multe ieşiri independente funcţional. El permite minimizări separate pentru fiecare dintre ieşirile sistemului. Utilizarea acestui principiu conduce la timpi de răspuns uniformi pentru sistemul minimizat. El intră în contradicţie cu principiul **minimizării în reţea**, care presupune utilizarea unor ieşiri pentru determinarea valorilor celorlalte ieşiri. Şi acest principiu nu poate fi aplicat decât sistemelor cu mai multe ieşiri. Am introdus şi acest principiu deoarece, chiar dacă are dezavantajul obţinerii unor sisteme cu timp de

răspuns în general mai mare decât al sistemelor minimizate în paralel, produce specificaţii cu o metrică globală foarte bună.

Utilizarea principiului minimizării în reţea produce rezultate al căror timp de răspuns se poate amplifica maxim, în raport cu sistemele minimizate paralel, cu un factor egal cu numărul de variabile de ieşire. Practic, minimizarea în reţea detectează în mod implicit corelaţiile dintre variabilele de ieşire (impuneri de seturi de valori de ieşire pentru o combinaţie dată de intrare) şi utilizează aceste corelaţii dacă pot conduce la un rezultat superior din punct de vedere al metricii utilizate.

Am introdus principiul **minimizării pe valorile variabilelor de ieşire** deoarece deschide calea utilizării principiului minimizării în reţea pentru sisteme cu o singură ieşire. Dezavantajul utilizării lui este că timpul de răspuns al sistemului minimizat poate să crească maxim cu un factor egal cu numărul de valenţe al domeniului variabilei de ieşire, dar are avantajul că obţine rezultate mult mai compacte.

Principiul **minimizării cu ieşiri corelate** este întrucâtva asemănător cu principiul minimizării în reţea, dar l-am introdus ca un principiu separat deoarece impune întotdeauna selectarea unei variante de soluţie care păstrează corelarea dintre ieşiri, chair dacă există o variantă mai bună din punct de vedere al metricii utilizate, dar care nu corespunde vectorilor de valori de ieşire aşa cum au fost specificaţi iniţial.

Aplicarea principiilor de mai sus m-a condus la dezvoltatarea unor strategii adecvate de minimizare. Aplicaţia are un „motor” care asigură determinarea implicanţilor şi selectarea soluţiilor pentru o funcţie cu mai multe ieşiri. Pentru a respecta principiile de mai sus sunt necesare aceste strategii care influenţează modul de pregătire al specificaţiei iniţiale pentru minimizare, modul de clasificare al vectorilor candidaţi şi condiţia de terminare a algoritmului principal.

Am determinat un set de şase strategii suficiente pentru respectarea principiilor de mai sus:

- strategia de minimizare în reţea;
- strategia pentru minimizarea sistemelor cu neunivocitate explicită;
- strategia pentru minimizarea sistemelor cu neunivocitate implicită;
- strategia pentru minimizarea sistemelor cu semantică nedeterministă;
- strategia pentru minimizarea sistemelor cu semantică deterministă;
- strategia pentru utilizarea opţiunii în rest.

Strategia de **minimizare în reţea** rezolvă o singură ieşire a sistemului la un moment dat. Dintre toate soluţiile obţinute (câte una pentru fiecare ieşire) este selectată cea mai bună soluţie din punct de vedere al metricii date. Ieşirea respectivă este mutată pe partea de intrare a sistemului deoarece ea va fi generată înaintea celorlalte variabile şi poate

fi utilizată ca intrare pentru restul de ieşiri ale sistemului. Procesul continuă până când sunt minimizate toate funcţiile.

Strategia pentru **minimizarea sistemelor cu neunivocitate explicită** influenţează modul în care un rând este considerat acoperit. Dacă sistemul are **semantică nedeterministă** atunci este suficientă rezolvarea uneia dintre intrări, iar dacă sistemul are **semantică deterministă**, rândul este considerat acoperit doar atunci când au fost rezolvate toate ieşirile lui. Strategia de **minimizare a sistemelor cu neunivocitate implicită** impune transformarea specificaţiei iniţiale într-una care conţine numai neunivocităţi explicite.

Utilizarea opţiunii „în rest” se poate face doar pentru valorile din sistem care permit acest lucru. Am prezentat o tehnică prin care se determină dacă o grupă de ieşire poate fi eliminată prin considerarea ieşirii grupei ca valoare implicită de ieşire.

În procesul de minimizare pot fi utilizate simultan mai multe principii şi strategii adecvate. În capitolul următor acestea sunt înglobate în aplicaţia finală dezvoltată pentru sisteme nedeterministe. Sistemele deterministe devin o clasă particulară de probleme.

8 Metodă finală propusă pentru sisteme deterministe și nedeterministe

Toate principiile şi strategiile prezentate în capitolul precedent au fost incorporate în aplicaţia COMIN (COmbinatorial MINimization)[236][237][238][239][242]. Aplicaţia înglobează nucleul dezvoltat iniţial pentru minimizarea sistemelor decizionale [177],[218],[220], dar extinde spectrul de aplicare asupra unei game mult mai largi de tabele. În acest capitol este descrisă implementarea aplicaţiei COMIN: procesările tabelelor de valori, algoritmii utilizaţi pentru implementare şi opţiunile disponibile.

Procesul de minimizare se caracterizează prin trei etape majore:

- prima etapă prelucrează specificaţia primară, transformând această specificaţie într-un tabel echivalent din punct de vedere funcţional, dar adaptat algoritmului de minimizăre;
- în a doua etapă este determinat, prin generare, setul complet de vectori implicanţi cu număr minim de literale;
- în ultima etapă este selectată o submulţime a setului complet de vectori implicanţi care îndeplineşte criteriile unei soluţii neredundante.

Metoda prezentată tratează unitar toate tipurile de tabele: univoce si neunivoce, deterministe sau nedeterministe, complet sau incomplet specificate, binare sau multivalente, cu ieşiri multiple sau singulare.

8.1 Extragerea funcţiilor şi mutarea variabilelor

Operaţia de construire de tabele individuale pentru funcţiile sistemului decizional a fost numită generare de tabele. Am implementat un generator de tabele care permite construirea de specificaţii noi pornind de la cele deja existente. Operaţia este utilizată şi pentru a muta variabile de ieşire pe intrare.

De exemplu, dintr-un tabel de valori care defineşte o funcţie $f : Z_2 \times Z_2 \times Z_2 \to Z_3 \times Z_5$ (xyz / ab) se pot construi funcţiile:

- $f_1 : Z_2 \times Z_2 \times Z_2 \to Z_3$ pentru relaţia (xyz / a),
- $f_2 : Z_2 \times Z_2 \times Z_2 \to Z_5$ pentru relaţia (xyz / b) şi
- $f_3 : Z_2 \times Z_2 \times Z_2 \times Z_5 \to Z_3$ pentru relaţia $(xyzb / a)$:

```
.library vectorned
.variable x 0 1
.variable y 0 1
.variable z 0 1
.variable a 0 1 2
.variable b 0 1 2 3 4

.function f x y z / a b
//x y z / a b
0 0 0 / 1 2
0 1 1 / 1 {3, 4}
0 1 1 / 2 3
1 0 0 / 2 2
1 1 1 / 0 -
```

```
.function f_1 x y z / a
//x y z / a
1 1 1 / 0
0 0 0 / 1
0 1 1 / 1
0 1 1 / 2
1 0 0 / 2

.function f_2 x y z / b
//x y z / b
0 0 0 / 2
1 0 0 / 2
0 1 1 / 3
0 1 1 / {3, 4}

.function f_3 x y z b / a
//x y z b / a
1 1 1 - / 0
0 0 0 2 / 1
0 1 1 {3, 4} / 1
0 1 1 3 / 2
1 0 0 2 / 2
```

Tab. 8-1 Extragerea funcţiilor (f_1, f_2) şi mutarea ieşirii b pe intrare componentelor (f_3)

Rezultatul generării de tabele este utilizat şi în vederea compactării şi eliminării parţiale a redundanţelor.

În rezultatul pentru variabila a (f_1) rândurile $(011/1)$ şi $(011/2)$ sunt compactate într-un singur rând $(011/\{1,2\})$:

```
.function f_1 x y z / a
//x y z / a
1 1 1 / 0
0 0 0 / 1
0 1 1 / 1
0 1 1 / 2
1 0 0 / 2
```

```
.function f_1_N x y z / a
//x y z / a
1 1 1 / 0
0 0 0 / 1
0 1 1 / {1,2}
1 0 0 / 2
```

Tab. 8-2 Minimizarea primei componente

Rezultatul pentru variabila pentru variabila b (f_2) este compactat prin eliminarea rândului $(011/3)$, cuprins în rândul $(011/\{3,4\})$:

```
.function f_2 x y z / b
//x y z / b
0 0 0 / 2
1 0 0 / 2
0 1 1 / 3
0 1 1 / {3, 4}
```

```
.function f_2_N x y z / b
//x y z / b
0 0 0 / 2
1 0 0 / 2
0 1 1 / {3, 4}
```

Tab. 8-3 Minimizarea celei de a doua componente

Pentru un tabel cu n rânduri care implică k variabile, complexitatea algoritmului pentru generarea unui tabel este $O(kn)$.

8.2 Corelarea valorilor variabilelor de ieşire

Corelarea valorilor variabilelor de ieşire are sens doar pentru tabelele cu ieşiri multiple. Corelarea înseamnă conservarea combinaţiilor de valori de ieşire posibile pentru fiecare combinaţie de intrare. Considerăm o funcţie f şi tabelele independente f_1 şi f_2 pentru fiecare variabilă de ieşire în parte.

```
.function f x y z / a b
//x y z / a b
0 0 0 / 1 2
0 1 1 / 1 {3, 4}
0 1 1 / 2 3
1 0 0 / 2 2
1 1 1 / 0 –
```

```
.function f_1 x y z / a
//x y z / a
1 1 1 / 0
0 0 0 / 1
0 1 1 / {1,2}
1 0 0 / 2
```

```
.function f_2 x y z / b
//x y z / b
0 0 0 / 2
1 0 0 / 2
0 1 1 / {3, 4}
```

Tab. 8-4 Extragerea subfuncţiilor dintr-o specificare cu ieşiri corelate

Combinaţia de intrare (011) este prezentă în două rânduri ale tabelului. Pentru această combinaţie, variabila a poate avea valorile de ieşire 1 sau 2 iar variabila b poate avea valorile de ieşire 3 sau 4. Tratarea independentă a celor două funcţii, pentru variabilele a şi b nu este corectă deoarece nu păstrează o informaţie intrinsecă a tabelului de mai sus: dacă variabila a are valoarea 2 atunci variabila b nu poate avea valoarea 4. Prin recuplarea celor două tabele se obţine funcţia f_12

```
.function f_12 x y z / a b
//x y z / a b
0 0 0 / 1 2
0 1 1 / {1, 2} {3, 4}
1 0 0 / 2 2
1 1 1 / 0 –
```

Tab. 8-5 Cuplarea a două funcţii

După cum se poate observa, funcţia f este diferită de funcţia f_12 deoarece pentru intrarea (011), pe ieşire poate să apară şi perechea (24) care nu exista iniţial în tabel. Se poate afirma că prin decuplare şi recuplare s-a pierdut corelarea dintre valorile variabilelor de ieşire şi au apărut rânduri suplimentare.

```
.function f x y z / a b
//x y z / a b
0 0 0 / 1 2
0 1 1 / 1 3
0 1 1 / 1 3
0 1 1 / 2 3
1 0 0 / 2 2
1 1 1 / 0 –
```

```
.function f_12 x y z / a b
//x y z / a b
0 0 0 / 1 2
0 1 1 / 1 3
0 1 1 / 1 3
0 1 1 / 2 3
0 1 1 / 2 4
1 0 0 / 2 2
1 1 1 / 0 –
```

Tab. 8-6 Minimizarea functiilor recuplate

Corelarea este o opţiune a programului care poate fi activată sau nu. Modul în care este setată această opţiune influenţează secvenţele automate de procesare pentru obţinerea minimizării.

Complexitatea algoritmului nu este influenţată de utilizarea cuplării.

8.3 Descompunerea intrărilor nedisjuncte şi a intrărilor compuse

Specificaţia primită trebuie preprocesată pentru a obţine un tabel care poate fi acceptat pentru prelucrare în pasul următor (determinarea vectorilor implicanţi). Rezultatul preprocesării este un tabel cu rânduri disjuncte din punct de vedere al intrărilor.

Prin această prelucrare sunt eliminate neunivocităţile implicite. Vectorii compuşi sunt descompuşi în vectori primi – vectori care au ca poziţii specificate ori o singură valoare din domeniu, ori DC – şi comprimaţi cu un algoritm Greedy.

În primul rând, orice pereche de rânduri care are nedeterminism implicit este descompusă în rânduri cu nedeterminism explicit.

Să considerăm următoarea funcţie cu ieşire „**în rest**” $f : Z_3 \times Z_3 \times Z_3 \times Z_3 \to Z_2 \times Z_3$:

```
   .function f i1 i2 i3 i4 / o1 o2
   //i1 i2 i3 i4 / o1 o2
1) 1 - 2 - / 1 0
2) - 2 - 0 / 1 1
3) 2 - 2 2 / - 2
4) / 0 –
```

Tab. 8-7 Specificare neunivocă implicită

Rândurile 1 şi 2 au nedeterminism implicit deoarece combinaţia de intrare (1220) este inclusă în ambele rânduri. Ieşirea o_2 ia valoarea 0 pentru primul rând şi 1 pentru al doilea rând. Algoritmul foloseşte diferenţa dintre mulţumi:

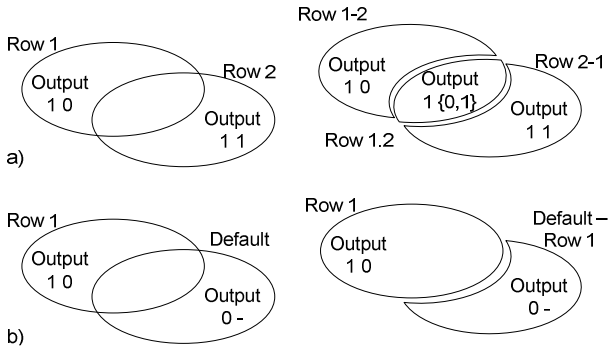


Fig. 8-1 Descompunerea a) dintre două rânduri obişnuite şi b) dintre un rând obişnuit şi specificarea „în rest”

Rezultatul descompunerii este o mulţime de rânduri cu nedeterminism explicit (rândul 1 din rezultat):

```
   .function f_D i1 i2 i3 i4 / o1 o2
   //i1 i2 i3 i4 / o1 o2
1) 1      2      2      0      / 1 {0, 1}
2) 1      {0, 1} 2      {1, 2} / 1 0
3) 1      {0, 1} 2      0      / 1 0
4) 1      2      2      {1, 2} / 1 0
5) {0, 2} 2      {0, 1} 0      / 1 1
6) {0, 2} 2      2      0      / 1 1
7) 1      2      {0, 1} 0      / 1 1
```

Tab. 8-8 Transformarea neunivocităţilor din implicite în explicite

Descompunerea este aplicată pentru fiecare pereche de rânduri – rând iniţial sau rezultat în urma unei descompuneri – care au nedeterminism implicit.

Următorul pas al procesări este descompunerea rândurilor care nu conţin numai valori prime în rânduri cu valori prime: (rândurile 2, 3, 4, 5, 6 şi 7 din funcţia de mai sus).

```
    .function f_D_P i1 i2 i3 i4 / o1 o2
    //i1 i2 i3 i4 / o1 o2
 1) 1 2 2 0 / 1 {0, 1}
 2) 1 0 2 1 / 1 0
 3) 1 1 2 1 / 1 0
 4) 1 0 2 2 / 1 0
 5) 1 1 2 2 / 1 0
 6) 1 0 2 0 / 1 0
```

```
 7) 1 1 2 0 / 1 0
 8) 1 2 2 1 / 1 0
 9) 1 2 2 2 / 1 0
10) 0 2 0 0 / 1 1
11) 2 2 0 0 / 1 1
12) 0 2 1 0 / 1 1
13) 2 2 1 0 / 1 1
14) 0 2 2 0 / 1 1
15) 2 2 2 0 / 1 1
16) 1 2 0 0 / 1 1
17) 1 2 1 0 / 1 1
```

Tab. 8-9 Eliminarea valorilor multiple diferite de DC

Ultima operaţie a preprocesării este operaţia de compresie rapidă a rândurilor. Rândul *0 2 – 0 / 1 1* de exemplu, înlocuieşte rândurile 10, 12 şi 14. Acest proces elimină redundanţa de calcul fără a denatura rezultatul.

În urma acestor procesări, considerând şi specificaţia "în rest", specificaţia iniţială devine:

```
.function f_N_C i1 i2 i3 i4 / o1 o2
//i1 i2 i3 i4 / o1 o2
- 2 0 0 / 1 1
- 2 0 1 / 0 -
- 2 0 2 / 0 -
- 2 1 0 / 1 1
- 2 1 1 / 0 -
- 2 1 2 / 0 -
0 0 0 - / 0 -
0 0 1 - / 0 -
0 0 2 0 / 0 -
0 - 2 1 / 0 -
0 - 2 2 / 0 -
0 1 0 - / 0 -
0 1 1 - / 0 -
0 1 2 0 / 0 -
0 2 2 0 / 1 1
1 0 0 - / 0 -
1 0 1 - / 0 –
1 0 2 0 / 1 0
1 - 2 1 / 1 0
1- 2 2 / 1 0
1 1 0 - / 0 -
1 1 1 - / 0 -
1 1 2 0 / 1 0
1 2 2 0 / 1 {0, 1}
2 - 2 2 / - 2
2 0 0 1 / 0 -
2 0 0 2 / 0 -
2 0 - 0 / 0 -
2 0 1 1 / 0 -
2 0 1 2 / 0 -
2 - 2 1 / 0 -
2 1 0 1 / 0 -
2 1 0 2 / 0 -
2 1 - 0 / 0 -
2 1 1 1 / 0 -
2 1 1 2 / 0 -
2 2 2 0 / 1 1
```

Tab. 8-10 Rezultatul minimizării

Complexitatea algoritmului de eliminare a disjuncţiilor depinde de logica intrinsecă a tabelului. În cel mai favorabil caz, pentru un tabel cu n rânduri şi k coloane de intrare, dacă nu trebuie efectuată nicio modificare în tabel, verificarea proprietăţii că toate rândurile sunt disjuncte se face prin compararea a k mulţimi pentru fiecare pereche de rânduri: $O\left(kn^2\right)$. În cel

mai defavorabil caz tabelul este transformat într-unul complet specificat fără nici o valoare *DC* sau *ANY* pe intrare. Pentru un tabel cu k coloane de intrare cu câte v, considerând că toate variabilele de intrare au acelaşi număr de valenţe, numărul de rânduri generat este v^k. Numărul de rânduri diferite din punct de vedere structural ce pot exista iniţial în tabel este $\left(2^v\right)^k$, unde 2^v reprezintă numărul de submulţimi ce pot fi folosite pentru fiecare poziţie ca valoare, dar în urma operaţiei de descompunere a rândurilor nu pot rezulta mai mult de v^k rânduri distincte. Numărul maxim de rânduri noi ce poate apare în urma descompunerii a două rânduri ca urmare a comparării acestora este $2k+1$, adică un rând pentru partea comună şi câte k rânduri pentru partea distinctă a fiecăruia din cele două rânduri, corespunzătoare fiecărei variabile de intrare în parte. Dacă notăm cu N maximul dintre numărul de rânduri iniţial şi numărul de rânduri final atunci se vor efectua $(N-1)(N-2)/2$ seturi de câte maxim k comparaţii – adică $O\left(N^2\right)$. Complexitatea în cazul în care se efectuează operaţia de descompunere este dată de $O\left(kN^2\right)$ comparaţii şi $O\left(kN^2\right)$ vectori generaţi.

Complexitatea operaţiei de descompunere este $O\left(kN^2\right)$, unde N este maximul dintre numărul de rânduri iniţiale din tabel şi numărul de rânduri rezultate în urma descompunerii. Este posibil ca în timpul descompunerii, temporar, numărul de rânduri din tabel să depăşească valoarea N, dar această situaţie este greu de anticipat deoarece depinde numai şi numai de logica intrinsecă din tabel.

8.4 Comprimarea tabelului de valori

Tabelul de valori este comprimat înainte de minimizare pentru a reduce numărul de informaţii procesat ulterior. Comprimarea se aplică numai pentru tabele descompuse în rânduri disjuncte. Comprimarea se face numai între rânduri care au aceleaşi valori de ieşire pentru toate variabilele de ieşire.

Algoritmul urmăreşte determinarea perechilor de rânduri care diferă printr-o singură poziţie. Aplicarea acestuia are ca rezultat reducerea numărului de rânduri şi a numărului de literale din tabel. Există o limită inferioară până la care poate fi redus un tabel aplicând această operaţie. Nu mi-am propus obţinerea unui rezultat optim. Metoda implementată utilizează un algoritm de tip Greedy.

Rândurile sunt grupate după valorile de ieşire (grupe de ieşire). În cadrul fiecărei grupe rândurile sunt sortate de k ori, de fiecare dată după câte $k-1$ dintre variabilele de intrare. Se compară rândurile alăturate două câte două pentru $k-1$ poziţii (poziţiile după care s-a făcut sortarea). Dacă

două rânduri sunt identice conform criteriului comparaţiei sunt înlocuite cu un singur rând care conţine doar poziţiile identice, iar pe poziţia k va fi trecută reuniunea seturilor de valori din aceste rânduri. Rezultatul unei astfel de compresii este prezentat în exemplul de mai jos:

```
.f p i01 i02 i03 i04 / out
//i01 i02 i03 i04 / out
0 0 0 0 / 0
0 0 1 0 / 0
0 1 1 0 / 0
1 0 0 0 / 0
1 0 0 1 / 0
1 0 1 0 / 0

0 0 0 1 / 1
0 0 1 1 / 1
0 1 0 0 / 1
0 1 0 1 / 1
0 1 1 1 / 1
1 0 1 1 / 1
1 1 0 0 / 1
1 1 0 1 / 1
1 1 1 0 / 1
1 1 1 1 / 1
```

```
.f p_C i01 i02 i03 i04 / out
//i01 i02 i03 i04 / out

0 1 1 0 / 0
- 0 - 0 / 0
1 0 0 1 / 0

0 0 0 1 / 1
- - 1 1 / 1
- 1 0 - / 1
1 1 1 0 / 1
```

Date iniţiale Rezultatul compresiei

Tab. 8-11 Comprimarea specificaţiei iniţiale

În timpul operaţiei de compresie a unei grupe cu n_g rânduri extrasă dintr-un tabel cu k coloane se fac k sortări de complexitate $O(kn_g \log n_g)$ şi k traversări în care se efectuează $k(n_g - 1)$ comparaţii. Pentru o grupă, complexitatea dată de sortări si comparaţii este $O(k^2 n_g \log n_g) + O(k^2 n_g)$.

Numărul de grupe este g şi numărul de rânduri este $n = \sum_{i=1}^{g} n_i$. Pentru o distribuţie echilibrată a rândurilor în grupe se poate afirma că $n_i \approx n / g$ pentru orice grupă. Complexitatea algoritmului este $O(g \times k^2 (n/g) \log n_g + g \times k^2 n_g) = O(k^2 n \log n_g)$, adică $O(k^2 n(\log n - \log g))$. Dacă g este mult mai mic decât n atunci complexitatea este $O(k^2 n \log n)$.

8.5 Expandarea pe valorile variabilelor de ieşire

Expandarea tabelului de intrare pe valorile variabilelor de ieşire se face în vederea unei minimizări mai pronunţate. Prin mărirea numărului de ieşiri al reţelei, funcţia cu una sau mai multe ieşiri multivalente este expandată pe valenţele fiecărei variabile de ieşire, generându-se un nou tabel.

Fiecare variabilă de ieşire este înlocuită cu un număr de variabile binare egal cu numărul de valenţe al domeniului variabilei respective. Expandarea este influenţată de opţiunea de corelare.

```
.library vectorned
.variable x 0 1
.variable y 0 1
.variable z 0 1
.variable a 0 1 2
.variable b 0 1 2 3
4
.variable a(0) 0 1
.variable a(1) 0 1
.variable a(2) 0 1
.variable b(0) 0 1
.variable b(1) 0 1
.variable b(2) 0 1
.variable b(3) 0 1
.variable b(4) 0 1

.function f x y z /
a b
//x y z / a b
0 0 0 / 1 2
0 1 1 / 1 {3, 4}
0 1 1 / 2 3
1 0 0 / 2 2
1 1 1 / 0 -
```

```
// expandare cu corelare
.function f_BE x y z / a(0) a(1) a(2) b(0)
b(1) b(2) b(3) b(4)
//x y z / a(0) a(1) a(2) b(0) b(1) b(2)
b(3) b(4)

0 0 0 / 0 1 0 0 0 1 0 0
0 1 1 / 0 1 0 0 0 0 1 1
0 1 1 / 0 0 1 0 0 0 1 0
1 0 0 / 0 0 1 0 0 1 0 0
1 1 1 / 1 0 0 - - - - -

// expandare fără corelare
.function f_BE_NCO x y z / a(0) a(1) a(2)
b(0) b(1) b(2) b(3) b(4)
//x y z / a(0) a(1) a(2) b(0) b(1) b(2)
b(3) b(4)

0 0 0 / 0 1 0 0 0 1 0 0
0 1 1 / 0 1 1 0 0 0 1 1
1 0 0 / 0 0 1 0 0 1 0 0
1 1 1 / 1 0 0 - - - - -
```

Tab. 8-12 Expandarea specificaţiei pe valorile variabilelor de ieşire

Pentru un tabel cu n rânduri, k_i variabile de intrare şi k_o variabile de ieşire cu v valenţe fiecare, complexitatea algoritmului este $O\left(\left(k_i + vk_o\right)n\right)$.

Rezultatul expandării cu ieşiri corelate nu se compactează deoarece se pierde corelarea şi, în urma minimizării, se obţine un rezultat identic cu cel al minimizării în paralele.

8.6 Determinarea implicanţilor

În această etapă de procesare este generată mulţimea de vectori implicanţi cu număr minim de literale [221][222]. Algoritmul generează implicanţi cu o singură poziţie specificată şi, după ce determină rândurile acoperite de aceştia, dacă mai există rânduri neacoperite continuă să caute implicanţi cu două poziţii specificate, trei poziţii speciifcate etc. până când toate rândurile sunt acoperite. Utilizând această metodă este evident că se obţin soluţii cu vectori implicanţi cu număr minim de literale (poziţii specificate).

Fiecare iteraţie a metodei este formată din două faze: în prima fază sunt generaţi implicanţi cu un anumit număr de poziţii spcificate şi în cea de a doua implicanţii generaţi sunt clasificaţi în vectori implicanţi, candidaţi (care pot conduce la obţinerea de vectori implicanţi) şi neviabili (care trebuie eliminaţi deoarece nu mai pot conduce la obţinerea de rezultate).

Cea de a doua fază include:

- calcularea atributelor;
- clasificarea candidaţilor;
- determinarea randurilor acoperite.

8.6.1 Generatorul

Generatorul foloseşte pachete, care împart spaţiul de căutare a implicanţilor în zone disjuncte. Candidaţii cuprinşi într-un pachet au cel puţin o componentă comună care separă aceşti candidaţi de cei din alte pachete. Să considerăm o funcţie definită pe domeniul $Z_3 \times Z_3 \times Z_3$. Generatorul creează un vector pentru fiecare valoare a fiecărui subdomeniu de intrare. Prima linie a tabelului de mai jos conţine primul pachet.

Pachet	Candidaţi								
	1	2	3	4	5	6	7	8	9
1	0--	1--	2--	-0-	-1-	-2-	--0	--1	--2
1.1	00-	01-	02-	0-0	0-1	0-2			
1.2	10-	11-	12-	1-0	1-1	1-2			
1.3	20-	21-	22-	2-0	2-1	2-2			
1.4	-00	-01	-02						
1.5	-10	-11	-12						
1.6	-20	-21	-22						
1.1.1	000	001	002						
1.1.2	010	011	012						
1.1.3	020	021	022						
1.2.1	100	101	102						
1.2.2	110	111	112						
1.2.3	120	121	122						
1.3.1	200	201	202						
1.3.2	210	211	212						
1.3.3	220	221	222						

Tab. 8-13 Generarea vectorilor

Al doilea pachet (1.1) este generat din pachetul 1, reţinând atributele lui. Primul candidat este combinat cu candidaţii de la 4 la 9 ai primului pachet. Vectorii 2 şi 3 nu pot fi combinaţi cu primul candidat, deoarece nu se intersectează (prezintă valenţe diferite pe aceeaşi poziţie).

Pachetul 1 este sursa pentru 6 pachete distincte cu două poziţii diferite de DC . Pachetele 1.1, 1.2 şi 1.3 generează fiecare câte trei pachete distincte. Fiecare pachet este construit adăugând la partea comună, dată de candidatul ales ca generator, ultima valoare diferită de DC a fiecărui candidat utilizat.

8.6.2 Procesarea atributelor

Pentru fiecare candidat este construit un set de atribute din care pot fi extrase caracteristicile acestuia. Setul de atribute este format din două părţi cu funcţionalitate distinctă:

- rândurile acoperite si intersectate de vectorul candidat;
- valorile de ieşire posibile ale acestuia pentru fiecare funcţie (ieşire a sistemului) în parte.

Prima parte conţine vectorii de intrare acoperiţi sau intersectaţi. Candidatul $(0 - - -)$ compus, la un moment dat, pentru funcţia prezentată în tabelul de mai jos acoperă rândurile 7, 8, 9, 10, 11, 12, 13, 14 şi 15, deoarece au aceeaşi valoare specificată pe prima poziţie şi intersectează rândurile 1, 2, 3, 4, 5 şi 6, deoarece au valoarea DC pe prima poziţie. Acest atribut nu este afectat de funcţiile de ieşire

```
    .function f_N_C i1 i2 i3 i4 /
o1 o2
    //i1 i2 i3 i4 / o1 o2
 1) - 2 0 0 / 1 1
 2) - 2 0 1 / 0 -
 3) - 2 0 2 / 0 -
 4) - 2 1 0 / 1 1
 5) - 2 1 1 / 0 -
 6) - 2 1 2 / 0 -
 7) 0 0 0 - / 0 -
 8) 0 0 1 - / 0 -
 9) 0 0 2 0 / 0 -
10) 0 - 2 1 / 0 -
11) 0 - 2 2 / 0 -
12) 0 1 0 - / 0 -
13) 0 1 1 - / 0 -
14) 0 1 2 0 / 0 -
15) 0 2 2 0 / 1 1
16) 1 0 0 - / 0 -
17) 1 0 1 - / 0 –
18) 1 0 2 0 / 1 0
19) 1 - 2 1 / 1 0
20) 1- 2 2 / 1 0
21) 1 1 0 - / 0 -
22) 1 1 1 - / 0 -
23) 1 1 2 0 / 1 0
24) 1 2 2 0 / 1 {0, 1}
25) 2 - 2 2 / - 2
26) 2 0 0 1 / 0 -
27) 2 0 0 2 / 0 -
28) 2 0 - 0 / 0 -
29) 2 0 1 1 / 0 -
30) 2 0 1 2 / 0 -
31) 2 - 2 1 / 0 -
32) 2 1 0 1 / 0 -
33) 2 1 0 2 / 0 -
34) 2 1 - 0 / 0 -
35) 2 1 1 1 / 0 -
36) 2 1 1 2 / 0 -
37) 2 2 2 0 / 1 1
```

Tab. 8-14 Lista cu implicanţi generaţi

A doua parte a atributului conţine, pentru fiecare funcţie, valorile de ieşire ale rândurilor acoperite sau intersectate. Dacă acest candidat este vector implicant, atunci el va fi memorat cu valorile de ieşire date de acest atribut.

Candidatul $(0 \ - \ - \ -)$ are mai multe ieşiri posibile (0 şi 1) pentru prima funcţie $O1$ şi permite orice valoare de ieşire pentru a doua funcţie $O2$, cu excepţia rândului 15 pentru care valoarea de ieşire poate fi doar 1. Prezenţa lui DC ca valoare de ieşire înseamnă ca toate valorile domeniului corespunzător sunt posibile.

Ca o concluzie, atributul $< C, I, O_f >$ asociat unui rând pentru funcţia f conţine o mulţime C de rânduri acoperite, o mulţime I de rânduri intersectate şi o mulţime O_f de valori posibile pentru ieşirea f.

În Fig. 8-2 este redat un caz în care atributele unui vector conţin rânduri acoperite (R_1 şi R_2), rânduri intersectate (R_3) şi există mai multe valori de ieşire pentru fiecare dintre aceste rânduri. Dacă acest vector este cosiderat implicant pentru rândurile acoperite atunci singura valoare posibilă de ieşire este 1. Ieşirea nu poate fi 2 deoarece R_3 nu are această valoare de ieşire în zona comună, nu poate fi 3 deoarece R_2 nu are această valoare de ieşire, nu poate fi 4 deoarece R_1 nu are această valoare de ieşire şi nu poate fi 5 deoarece R_1 şi R_2 nu au această valoare de ieşire. Vectorul este pe de o parte implicant pentru rândurile R_1 şi R_2 (numai pentru ieşirea 1) cât şi, pe de altă parte, candidat, deoarece prin adăugarea de poziţii specificate este posibil să nu mai intersecteze rândul R_3, ceea ce ar conduce la un implicant cu ieşire neunivocă $\{1,2\}$.

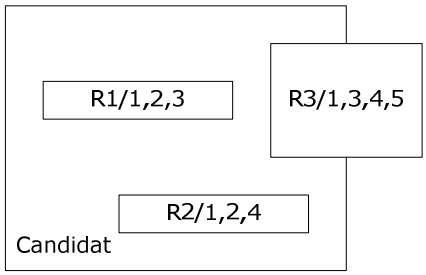


Fig. 8-2 Reprezentarea mulţimilor din atribute

Atributul candidatului pentru o funcţie cu n ieşiri este $< C, I, < O_{f_1}, O_{f_2}, O_{f_3}, ..., O_{f_n} >>$.

Structura de date pentru memorarea valorilor de ieşire este o listă de perechi $< v, a >$ pentru fiecare funcţie (v reprezintă o valoare posibilă de ieşire iar a reprezintă numărul de rânduri acoperite sau intersectate care

conţin valoarea v ca ieşire). Pentru vectorul din Fig. 8-2 elementele atributului $\langle C, I, \langle O_f \rangle \rangle$ au următoarele valori:

- $C = \{R_1, R_2\}$,
- $I = \{R_3\}$ şi
- $O_f = -\{\langle 1,3\rangle, \langle 2,2\rangle, \langle 3,2\rangle, \langle 4,2\rangle, \langle 5,1\rangle\}$.

Pornind de la doi candidaţi $V_1 < C_1, I_1, O_{1f} >$ şi $V_2 < C_2, I_2, O_{2f} >$, atributele noului candidat $V < C, I, O_f >$ sunt obţinute astfel:

- $C = C_1 \cap C_2$
- $I = (C_1 \cap I_2) \cup (I_1 \cap C_2) \cup (I_1 \cap I_2)$
- O_f este obţinut pornind de la O_{1f} sau O_{2f}. Dacă se porneşte de la O_{1f}, în perechile de valori de ieşire din acesta a este decrementat pentru fiecare rând acoperit sau intersectat de O_{1f}, care nu mai aparţine noului candidat.

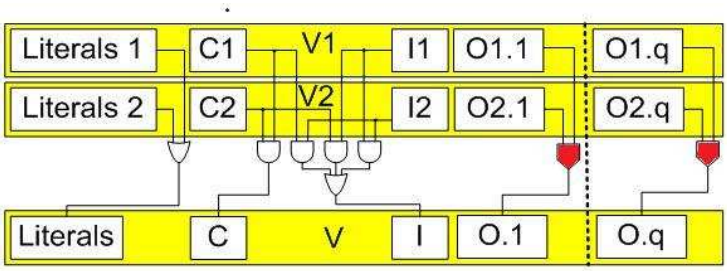


Fig. 8-3 Procesarea atributelor

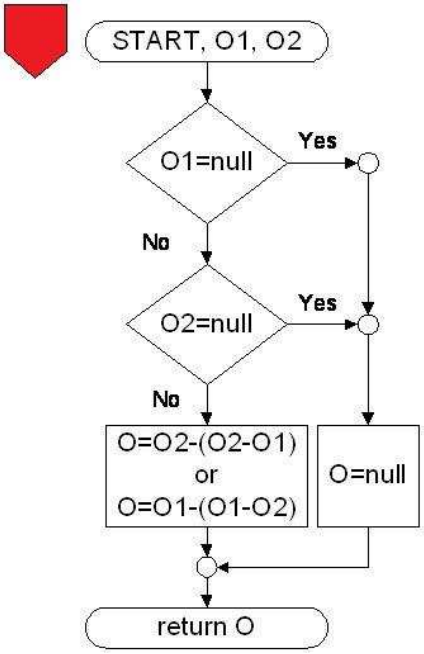


Fig. 8-4 Combinarea valorilor de ieşire din două pachete de atribute

Structurile O_f memorează numărul de apariţii al fiecărei valori de ieşire pentru rândurile acoperite sau intersectate. Dacă numărul de apariţii al unei valori este egal cu suma numărului de rânduri acoperite sau intersectate înseamnă că poate fi valoare de ieşire. Această valoare este numită în continuare valoare de ieşire permisă. Candidatul este implicant dacă există cel puţin o valoare de ieşire permisă.

Metoda evită calculele redundante, permiţând determinarea valorilor atributului iterativ, printr-un număr minim de procesări.

8.6.3 Clasificarea candidaţilor

Fiecare candidat este clasificat prin analiză potrivit atributelor sale. Clasificarea este efectuată după calcularea fiecărui candidat.

Dacă mulţimea C este vidă, atunci candidatul nu este viabil şi algoritmul îl şterge din pachet.

Dacă mulţimea C nu este vidă şi există valori de ieşire disponibile pentru o funcţie (O_f nu este vid), este o implicaţie pentru acea funcţie şi copia sa este păstrată în setul de rezultate. În pachet, atributul O_f este setat „**null**”, şi algoritmul nu va mai calcula atributul pentru această funcţie şi pentru nici un alt descendent al candidatului.

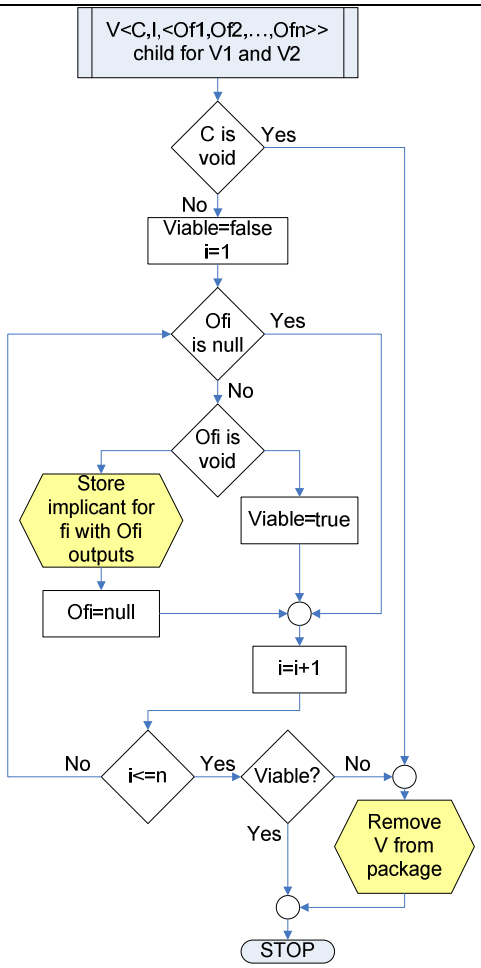


Fig. 8-5 Clasificarea candidaţilor

Algoritmul păstrează candidaţii pentru care mulţimea C nu este vidă şi cel puţin un set O_f nu este „**null**”.

8.6.4 Condiţia de oprire a algoritmilor

După generarea tuturor implicaţiilor cu acelaşi număr de literale, este actualizată o structură generală care memorează valorile acoperite din fiecare rând, pentru fiecare funcţie. Algoritmul se opreşte după ce toate valorile fiecărui rând sunt acoperite pentru toate funcţiile (toate valorile sale de ieşire sunt acoperite pentru fiecare funcţie).

O altă abordare ar putea fi făcută dacă definiţia unei funcţii acoperite este schimbată, astfel încât o funcţie pentru un rând este acoperită dacă cel puţin o valoare de ieşire a funcţiei este acoperită. Această condiţie de oprire

conduce la un rezultat care nu păstrează toate specificaţiile iniţiale, dar, în multe cazuri, generează o soluţie satisfăcătoare.

Structura care păstrează starea algoritmului poate fi actualizată după generarea tuturor implicanţilor cu acelaşi număr de literale, sau după generarea fiecărui implicant. În al doilea caz, algoritmul are o convergenţă mai mare, dar nu toţi implicanţii cu acelaşi număr de literale sunt generaţi.

8.7 Extragerea soluţiei

În această etapă, algoritmul extrage din setul implicanţilor, un subset redus care acoperă rândurile specificaţiei iniţiale.

Extragerea setului de implicanţi

Implicanţii sunt sortaţi în ordine descrescătoare, în funcţie de numărul de vectori iniţial acoperiţi. Primul implicant din listă este adăugat la soluţie. Valorile acoperite de implicantul selectat sunt şterse din atributele implicanţilor rămaşi. Implicanţii care nu sunt selectaţi şi nu acoperită nici o funcţie sunt şterşi.

Procesul continuă pană când fiecare implicant este şters sau selectat în soluţie.

Considerând numărul de valori acoperite, algoritmul poate fi aplicat în două feluri:

- soluţia este calculată pentru fiecare funcţie în parte, şi numărul de valori acoperite este calculat distinct pentru fiecare funcţie;
- soluţia este calculată pentru toate funcţiile simultan, şi numărul valorilor acoperite ale implicantului este însumat pentru toate funcţiile. Această metodă are avantajul că implicanţii comuni sunt descoperiţi implicit.

Complexitatea algoritmului de extragere a unui set minim de implicanţi este dată de numărul de implicanţi n, de numărul de rânduri r din tabelul iniţial şi de numărul de valenţe mediu pe ieşire v. În cel mai rău caz este necesară alegerea a exact $r \times v$ implicanţi (câte unul pentru fiecare rând). După fiecare selectare de implicant se parcurge lista de implicanţi rămaşi pentru a ajusta mulţimile de rânduri rămase neacoperite şi se face o sortare a implicanţilor rămaşi (care acoperă rânduri neacoperite încă). Complexitatea este $O(rvn \log n)$. Aceasta este complexitatea maximă pentru extragerea setului minim de implicanţi. Odată cu selectarea unui implicant sunt acoperite mai multe rânduri simultan.

Eliminarea vectorilor redundanţi

La alcătuirea soluţiei, se adaugă vectori după numărul de rânduri acoperite, dar fiecare vector nou introdus acoperă un set de rânduri care au fost deja acoperite de vectori adăugaţi anterior în soluţie şi un set de

rânduri pe care îi acoperă doar el (motiv pentru care a fost ales). Dar după introducerea mai multor vectori se poate întâmpla ca toţi vectorii care apăreau ca neacoperiţi la momentul introduceri lui să apară în listele vectorilor implicanţi introduşi ulterior. Acest vector este vector redundant, eliminarea lui nu modifică cu nimic specificaţia funcţiei.

Complexitatea algoritmului pentru eliminarea vectorilor redundanţi este determinată de numărul de vectori implicanţi din soluţie selectaţi până în această etapă şi de numărul de rânduri din tabelul iniţial r. Complexitatea este $O\left(rn^2\right)$.

Selecţia unei valori „default”

Utilizarea unei valori „default” într-un tabel se face numai în cazul în care combinaţiile de intrare corespunzătoare grupei alese pentru a fi specificată astfel sunt disjuncte (nu au nici o intersecţie) faţă de combinaţiile de intrare corespunzătoare celorlalte valori ale respectivei variabile de ieşire.

În cazul în care tabelul iniţial prezintă o specificare „**în rest**” care contravine regulii enunţate în propoziţia anterioară, aceasta nu afectează procesul de minimizare, dar la specificarea unei valori „**default**” pe soluţia minimizată, se aplică această regulă.

Eliminarea intrărilor nediscriminatorii

În urma minimizării anumite valori ale unor variabile de intrare nu vor mai fi folosite. S-a determinat că ele nu ajută în procesul de discriminare. Când toate valorile unei variabile nu mai sunt folosite (la toate rândurile apare valoarea DC) variabila poate fi ştearsă. Neavând pe nici un rând valoare specificată ea nu ajută în procesul de discriminare, deci rezultatul funcţiei nu depinde de această variabilă.

8.8 Minimizarea paralelă în raport cu minimizărea în reţea

Minimizarea se poate face în două feluri: în paralel sau în reţea.

Minimizarea paralelă se obţine prin descompunerea tabelul iniţial în câte un tabel pentru fiecare variabilă de ieşire. Fiecare tabel rezultat este minimizat distinct. Rezultatul este reprezentat de tabele care au ca variabile de intrare numai variabilele de intrare ale tabelului iniţial.

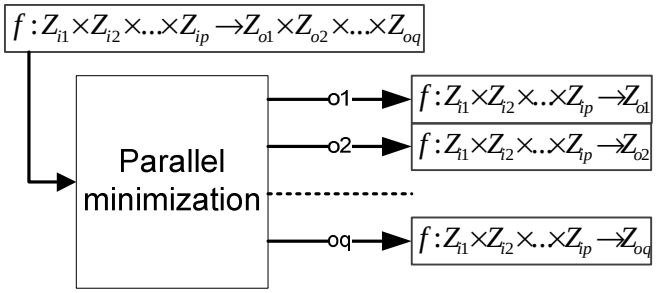


Fig. 8-6 Minimizarea paralelă

Minimizarea în reţea întoarce pe intrare toate variabilele de ieşire deja calculate. Procesul de minimizare în reţea porneşte cu acoperirea în paralel a rândurilor tabelului iniţial generând vectori cu din ce în ce mai multe poziţii specificate. Dacă la un pas este acoperită cel puţin una dintre variabilele de intrare se aplică procesul de selectare a soluţiei pentru fiecare variabilă acoperită în parte şi se alege variabila al cărei tabel rezultat este cel mai simplu. Se reţine numai această soluţie. Variabila este mutată în tabelul iniţial din partea de ieşire în partea de intrare a tabelului şi se reia procesul de minimizare. Procesul de minimizare continuă până când au fost rezolvate toate variabilele de ieşire.

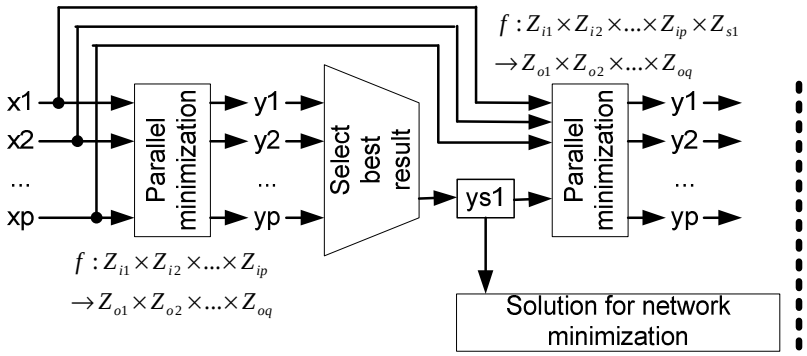


Fig. 8-7 Minimizarea în reţea

Minimizarea în reţea corespunde într-o implementare software cu folosirea variabilelor temporare.

8.9 Minimizarea pe variabile în raport cu minimizarea pe valenţe

O abordare nouă, prezentată în această lucrare, este minimizarea sistemului decizional al funcţiei, în loc de funcţia în sine. Ieşirile specificaţiilor date sunt transformate în ieşiri binare, însemnând activarea numai a valenţelor care corespund ieşirii funcţiei.

Rezultatul minimizării în reţea pentru variabilele *a* şi *b* este:

```
.library vectorned
.variable x 0 1
.variable y 0 1
.variable z 0 1
.variable a 0 1 2
.variable b 2 3 4

.function f x y z / a b
//x y z / a b
0 0 0 / 1 2
0 1 1 / 1 {3, 4}
0 1 1 / 2 3
1 0 0 / 2 2
1 1 1 / 0 -
```

```
.function f_n/n_NM_b_sol y / b
//y / b
1 / {3, 4}
/ 2

.function f_n/n_NM_a_sol x y b / a
//x y b / a
0 - - / 1
0 - 3 / {1, 2}
1 0 - / 2
/ 0
```

Fig. 8-8 Minimizare în reţea

Folosind transformarea ieşirilor în ieşiri pe valenţe funcţia $f : Z_3 \times Z_3 \times Z_3 \times Z_3 \to Z_2 \times Z_3$ va deveni o funcţie cu cinci ieşiri binare $f_{ov} : Z_3 \times Z_3 \times Z_3 \times Z_3 \to (Z_2 \times Z_2) \times (Z_2 \times Z_2 \times Z_2)$.

```
.function f x y z / a b
//x y z / a b
0 0 0 / 1 2
0 1 1 / 1 {3, 4}
0 1 1 / 2 3
1 0 0 / 2 2
1 1 1 / 0 –
```

```
.function f_BE x y z / a0 a1 a2 b2 b3 b4
//x y z / a(0) a(1) a(2) b(2) b(3) b(4)
0 0 0 / 0 1 0 1 0 0
0 1 1 / 0 1 0 0 1 1
0 1 1 / 0 0 1 0 1 0
1 0 0 / 0 0 1 1 0 0
1 1 1 / 1 0 0 0 0 0
```

Fig. 8-9 Minimizare cu expandare în binar

Rezultatul minimizării în reţea a funcţiei pe valenţele de ieşire este:

```
//y / b(2)
0 / 1
/ 0

//x y / a(1)
- 1 / 0
1 - / 0
0 - / 1

//x y / a(0)
1 1 / 1
/ 0
```

```
//a(1) a(0) / a(2)
0 0 / 1
/ 0

//x y / b(3)
0 1 / 1
/ 0

//a(2) b(3) / b(4)
0 1 / 1
/ 0
```

Fig. 8-10 Minimizare în reţea după expandare în binar

Cum această abordare oferă mai multe opţiuni de interconectare a reţelei, rezultatele minimizării sunt mai bune ca în cazul minimizării pe

variabile, când ieşirile sunt considerate variabile multivalente pure. De asemenea, această transformare permite utilizarea minimizării în reţea pentru sisteme decizionale cu o singură ieşire [219].

8.10 Concluzii

În acest capitol am prezentat o metodă proprie de minimizare, complexă prin faptul că acceptă o gamă completă de sisteme decizionale şi permite specificarea unui număr ridicat de opţiuni de procesare. În cadrul prezentării am descris metodele prin care rezolv fiecare etapă de minimizare şi am exemplificat procesarea cu rezultate obţinute cu ajutorul aplicaţiei COMIN. Aplicaţia implementată integrează modelului dezvoltat. COMIN are următoarele caracteristici:

- acceptă variabile de intrare şi ieşire binare şi multivalente,
- se aplică corespondenţelor cu una sau mai multe variabile de intrare şi cu una sau mai multe variabile de ieşire,
- permite minimizarea separată sau în reţea a ieşirile multiple (conform strategiei de structurare alese);
- acceptă funcţii complet sau incomplet specificate,
- acceptă specificarea completă prin utilizarea unei declaraţii logice (printr-un rând din tabel) de forma „în rest ieşirea are valoarea …”.
- se aplică tabelelor de valori deterministe sau nedeterministe, care pot prezenta neunivocităţi explicite şi/sau implicite;
- permite procesarea independentă sau corelată a funcţiilor din speciifcaţia iniţială.

Concluzii asupra metodei de determinare a implicanţilor.

Complexitatea procesului de minimizare dezvoltat este determinată atât de cantitatea de date de intrare cât şi de complexitatea intrinsecă a logicii integrate în tabelul de valori. Cantitatea de literale şi rânduri este uşor cuantificabilă – prin contorizare directă. Logica intrinsecă a tabelului este o mărime necuantificabilă şi influenţează dramatic evoluţia algoritmului atât din punct de vedere al timpului de procesare cât şi din punct de vedere al memoriei utilizate.

Operaţia esenţială pentru minimizare constă în determinarea de cuburi adiacente din spaţiul combinaţiilor de intrare cărora le corespunde aceeaşi valoare de ieşire. Odată determinate două astfel de cuburi este înlocuită, dacă este posibil, variabila de intrare care le diferenţiază cu „**don’t care**”. Există situaţii în care două specificaţii identice din punct de vedere al metricii (număr de rânduri şi număr de literale pe fiecare rând) care conduc la rezultate cu o complexitate diametral opusă şi care procesări de complexităţi extreme. Consider o specificaţie completă care conţine patru variabile de intrare binare şi două ieşiri binare. Una dintre funcţii

(prima ieşire) implementează funcţia *„AND”* şi cealaltă (a doua ieşire) implementează funcţia *„XOR”*. Tabelul de valori este:

#	Intrări				Ieşiri	
	V1	V2	V2	V3	AND	XOR
1	0	0	0	0	0	0
2	0	0	0	1	0	1
3	0	0	1	0	0	1
4	0	0	1	1	0	0
5	0	1	0	0	0	1
6	0	1	0	1	0	0
7	0	1	1	0	0	0
8	0	1	1	1	0	1
9	1	0	0	0	0	1
10	1	0	0	1	0	0
11	1	0	1	0	0	0
12	1	0	1	1	0	1
13	1	1	0	0	0	0
14	1	1	0	1	0	1
15	1	1	1	0	0	1
16	1	1	1	1	1	0

Tab. 8-15 Funcţiile *AND* şi *XOR*

AND	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	1	0
10	0	0	0	0

Tab. 8-16 Diagramă *AND*

XOR	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

Tab. 8-17 Diagramă *XOR*

Rezultatul minimizării funcţiei *„AND”* este:

```
.function AND i1 i2 i3 i4 / and
//i1 i2 i3 i4 / and
0 - - - / 0
- 0 - - / 0
- - 0 - / 0
- - - 0 / 0
1 1 1 1 / 1
```

Tab. 8-18 Minimizarea funcţiei *„AND”* cu 4 variabile de intrare

Rezultatul minimizării funcţiei *„XOR”* este:

```
.function XOR i1 i2 i3 i4 / xor
//i1 i2 i3 i4 / xor
0 0 0 0 / 0
0 0 0 1 / 1
0 0 1 0 / 1
0 0 1 1 / 0
0 1 0 0 / 1
0 1 0 1 / 0
0 1 1 0 / 0
0 1 1 1 / 1
1 0 0 0 / 1
1 0 0 1 / 0
1 0 1 0 / 0
1 0 1 1 / 1
```

1 1 0 0	/ 0
1 1 0 1	/ 1
1 1 1 0	/ 1
1 1 1 1	/ 0

Tab. 8-19 Minimizarea funcţiei „ *XOR* ” cu 4 variabile de intrare

Cele două tabele de intrare au acelaşi număr de rânduri – 16 – şi acelaşi număr de literale – 80. Cu toate acestea, rezultatul minimizării primului tabel este format din 13 literale şi 5 rânduri, iar rezultatul minimizării celui de al doilea tabel este format din 80 de literale şi 16 rânduri. Această diferenţă se explică numai prin logica intrinsecă diferită din cele două tabele.

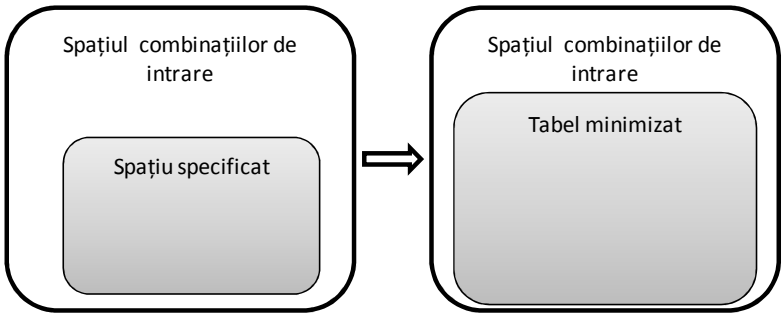


Fig. 8-11 Tabelul minimizat acoperă cel puţin spaţiul specificat

Minimizarea se obţine din două surse:

- cuplarea rândurilor din tabelul iniţial cu ieşiri identice care reprezintă cuburi adiacente din spaţiul combinaţiilor de intrare prin înlocuirea poziţiilor specificate diferite cu „**don’t care**”;

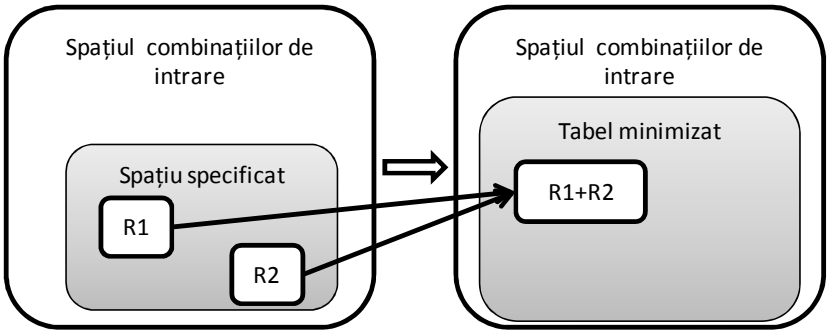


Fig. 8-12 Prima sursă a minimizării – cuplarea

- asimilarea la rândurile specificate de cuburi din spaţiul nespecificat iniţial al combinaţiilor de intrare. Valoarea de ieşire a rândului iniţial este atribuită ca valoare de ieşire şi cubului asimilat. Acelaşi cub poate fi asimilat de rânduri cu ieşiri diferite. Ca urmare a acestui proces rezultatul minimizării poate fi

nedeterminist. Asimilarea de zone nespecificate nu reprezintă un scop al tehnicii prezentate. Asimilarea este un efect al aplicării metodei.

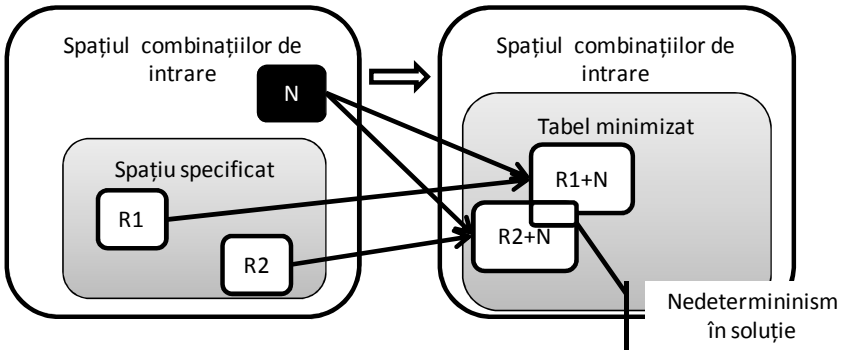


Fig. 8-13 A doua sursă a minimizării – utilizarea spaţiului nespecificat – poate introduce nedeterminism

Complexitatea algoritmului de determinare a implicanţilor nu poate fi stabilită cu certitudine, dar se pot determina limitele acestuia.

Vom presupune că în această etapă de procesare datele de intrare sunt formate din n rânduri, cu câte k_i coloane de intrare, k_o coloane de ieşire, v valenţe pe domeniu şi p poziţii specificate pe combinaţia de intrare a fiecărui rând. În plus, toate combinaţiile de intrare sunt disjuncte. Numărul minim de vectori implicanţi necesari este egal cu numărul de grupe de ieşire. În schimb numărul maxim de vectori generaţi este egal cu numărul de vectori distincţi cu număr de poziţii specificate mai mic decât p. Numărul acestor vectori este $\sum_{j=0}^{p} C_k^j v^j$. În funcţie de structura combinaţiilor de intrare şi a valorilor de ieşire această cantitate de vectori procesaţi se poate reduce dramatic.

190 Metodă finală propusă pentru sisteme deterministe şi nedeterministe - 8

9 Studii de caz - Rezultate

În acest capitol prezint aplicaţia finală, opţiunile puse la dispoziţie de aceasta şi cazuri de utilizare pentru a evidenţia aria de probleme ce pot fi soluţionate cu această metodă.

9.1 Aplicaţia de minimizare COMIN

Am implementat în aplicaţia COMIN metoda dezvoltată în această lucrare. Fig. 8-2 conţine interfaţa pusă la dispoziţia utilizatorului. Specificarea unui sistem decizional include:

- numele sistemului
- declararea variabilelor utilizate
- descrierea domeniilor variabilelor prin enumerarea valenţelor. Descrierea domeniilor poate fi incompletă deoarece aplicaţia adaugă domeniului fiecărei variabile fiecare valoare regăsită în descrierea funcţiilor sistemului pe coloana corespunzătoare variabilei respective.
- descrierea funcţiilor sistemului decizional prin:
  - specificarea unui antet care conţine numele funcţiei, numele variabilelor de intrare şi al variabilelor de ieşire
  - specificarea corespondenţelor valorice intrare/ieşire
  - specificarea opţională a unei valori de ieşire pentru combinaţiile de intrare ce nu au fost prinse în corespondenţele explicite

Specificarea sistemului decizional nu include elementele de semantică ce determină modul în care decurge procesul de minimizare. Strategiile utilizate în timpul minimizării prevăzute ca opţiuni ce trebuie selectate înainte de lansarea procesului de minimizare din interfaţa aplicaţiei sunt:

- tipul de minimizare;
- criteriul care stă la baza selecţiilor în timpul minimizării;
- semantica deterministă sau nedeterministă;
- determinarea dacă este posibil sau nu a unei valori „în rest”;
- modul de prezentare al rezultatelor.

Alte operaţii auxiliare puse la dispoziţia utilizatorului sunt:

- descompunerea funcţiilor în funcţii cu rânduri disjuncte
  - fără a denatura specificaţia integrată în aceasta,

  - o ţinând cont sau nu de corelarea valorilor de ieşire,
- comprimarea rapidă a funcţiilor fără a integra combinaţii de intrare nespecificate,
- sortarea rândurilor din specificaţia funcţiilor după
  - o valorile variabilelor de intrare sau
  - o după valorile variabilelor de ieşire,
- descompunerea funcţiilor după valorile variabilelor de ieşire,
- multiplicarea funcţiilor pentru fiecare variabilă de ieşire cu eliminarea rândurilor neesenţiale,
- mutarea de variabile de pe ieşirea pe intrarea funcţiilor.

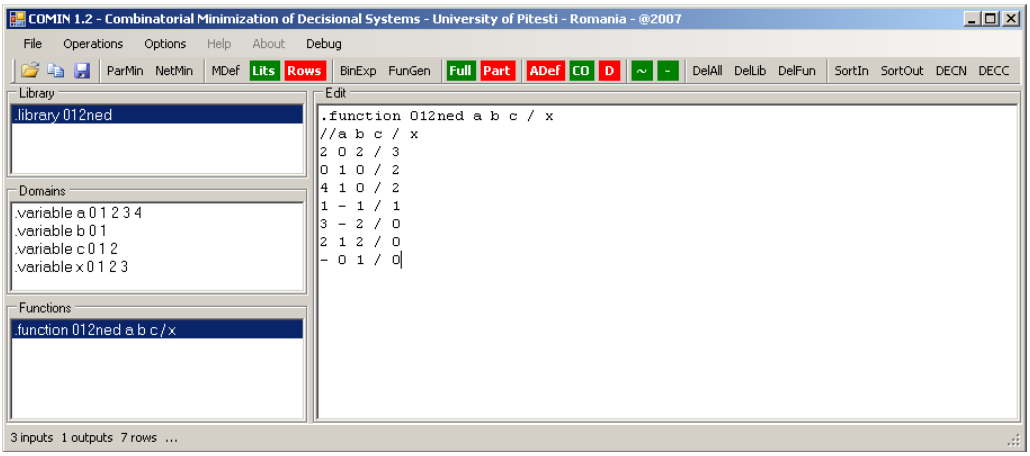


Fig. 9-1 Tabel de valori incomplet specificat cu intrări multivalente şi ieşire singulară multivalentă

9.2 Studii de caz

Exemplele ce urmează au fost alese pentru a evalua rezultatele minimizării utilizând metoda dezvoltată în comparaţie cu metoda MVSIS, singura metodă clasică ce asigură suport pentru sisteme decizionale multivalente. Cele 10 exemple au şi scopul de a prezenta gradual modul de utilizare al aplicaţiei COMIN.

Exemplul 1

Am definit două sisteme multivalente cu 3 intrări şi o ieşire singulară, unul determinist (**012det**) şi unul nedeterminist (**012ned**).

.library 012 .variable a 0 1 2 3 4 .variable b 0 1 2 3 4 .variable c 0 1 2 3 4 .variable x 0 1 2 3 4	.function 012det a b c / x //a b c / x 2 0 1 / 0 2 1 2 / 0 3 - 2 / 0 1 - 1 / 1 0 1 0 / 2 4 1 0 / 2 2 0 2 / 3

Tab. 9-1 Specificarea funcţiei **012det**

Am minimizat funcţia **012det** cu aplicaţia COMIN şi am obţinut un rezultat format din 5 literale şi 3 rânduri şi un rezultat format din 6 literale şi 3 rânduri pentru minimizarea cu MVSIS.

COMIN	MVSIS
.function 012det a b c / x //a b c / x 1 - - / 1 - - 0 / 2 2 0 2 / 3 / 0	.table a b c / x //a b c / x 1 - 1 / 1 - - 0 / 2 2 0 2 / 3 / 0
5 literale, 3 rânduri, 1 default	6 literale, 3 rânduri, 1 default

Tab. 9-2 Rezultatele minimizării obţinute cu COMIN şi MVSIS pentru **012det**

Diferenţa dintre cele două rezultate (COMIN a obţinut cu 16% mai puţine literale la acelaşi număr de cuburi) provine de la algoritmul de eliminare al combinaţilor de intrare comune apărute în urma determinării flexibilităţii maxime în metoda MVSIS).

Exemplul 2

Cel de al doilea sistem prezentat este o variantă uşor modificată a sistemului 01det astfel încât să prezinte o nedeterminare (neunivocitate implicită) evidenţiată de valoarea marcată din tabelul de mai jos:

.library 012 .variable a 0 1 2 3 4 .variable b 0 1 2 3 4 .variable c 0 1 2 3 4 .variable x 0 1 2 3 4	.function 012ned a b c / x //a b c / x - 0 1 / 0 2 1 2 / 0 3 - 2 / 0 1 - 1 / 1 0 1 0 / 2 4 1 0 / 2 2 0 2 / 3

Tab. 9-3 Specificarea funcţiei **012ned**

Rândul 1 conţine valoarea DC pe prima poziţie în loc de valoarea 2. Combinaţia 1 0 1 este acoperită atât de rândul 1 (- 0 1) cât şi de rândul 4 (1 - 1). O procesare cu semantică deterministă trebuie să prezinte la ieşirea sistemului valorile de ieşire $\{0,1\}$ pentru combinaţia de intrare (1 0 1), iar o procesare cu semantică nedeterministă trebuie să prezinte oricare dintre valorile 0 sau 1, sau ambele valori simultan. Dacă nu sunt admise ambele valori simultan atunci alegerea uneia dintre valorile regăsite este suficientă pentru a elimina nedeterminismul din sistem.

Rezultatele obţinute cu cele două sisteme de minimizare sunt:

COMIN	MVSIS
.function 012ned_k/n_NMC_x_Def a b c / x **//a b c / x** **1 - - / 1** **- - 0 / 2** **2 0 2 / 3** **/ 0**	**.table a b c x** **.default 0** **1 (1,2,3,4) 1 1** **- - 0 2** **2 0 2 3**
5 literale, 3 rânduri, 1 default	7 literale (10 valori), 3 rânduri, 1 default

Tab. 9-4 Rezultatele minimizării obţinute cu COMIN şi MVSIS pentru **012ned**

Modul de afişare al rezultatelor implică interpretarea mulţimii de valori $\{1,2,3,4\}$ ca un singur literal. În acest caz este posibilă utilizarea unui singur operator de decizie (diferit de 0) care să determine dacă variabila b face parte din această mulţime, dar nu în toate cazurile este posibil acest lucru. Propriu-zis, chiar dacă sunt numărate 7 literale de fapt sunt necesari 10 operatori de echivalenţă. Numărul acestor operatori poate scădea pentru situaţii particulare în care sunt puse în evidenţă anumite particularităţi ale sistemului.

Suplimentar COMIN permite minimizarea directă cu semantică deterministă. Rezultatul obţinut este:

COMIN
.function 012ned_n/n_NMC_x_Def a b c / x **//a b c / x** **- 0 1 / 0** **2 1 - / 0** **3 - - / 0** **1 - - / 1** **- - 0 / 2** **/ 3**
7 literale, 5 rânduri, 1 default

Tab. 9-5 Rezultatul minimizării deterministe cu COMIN pentru **012ned**

Rezolvarea sistemului fără a considera o valoare de ieşire implicită este:

COMIN

```
.function 012ned_n/n_NMC_x_Def a b c
/ x
//a b c / x
- 0 1 / 0
2 1 - / 0
3 - - / 0
1 - - / 1
- - 0 / 2
2 0 2 / 3
```

10 literale, 6 rânduri, fără default

Tab. 9-6 Rezultatul minimizării deterministe cu COMIN pentru **012ned** fără valoare de ieşire implicită

Alegerea valorii de ieşire 3 ca valoare implicită a fost dictată de următoarele premise:

- combinaţia de intrare 1 0 1 trebuie să conducă la ieşirea $\{0,1\}$. Ieşirea 0 este dată de primul rând din rezultat (- 0 1 / 0) iar ieşirea 1 este dată de al patrulea rând (1 - - / 1). Alegerea oricăreia dintre valorile 0 sau 1 ca valoare de ieşire implicită ar fi condus la denaturarea sistemului (soluţia nu ar fi păstrat integral specificaţia iniţială). De exemplu, alegerea valorii 0 transformă sistemul în Tab. 9-7, în care combinaţia de intrare (1 0 1) nu poate determina decât ieşirea 1. Această combinaţie, fiind acoperită de primul rând din rezultat nu mai are cum să implice şi ieşirea 0.
- singurele valori de ieşire valide pentru a fi alese să reprezinte o ieşire implicită sunt 2 şi 3. Alegerea uneia dintre acestea este determinată de numărul de literale pe care le elimină din tabel. Cum valoarea 3 se determină cu ajutorul a 3 literale, iar valoarea 2 cu ajutorul unui singur literal alegerea valorii 3 implică cea mai mare reducere de literale.

```
.function 012ned_n/n_NMC_x_Def a b c
/ x
//a b c / x
1 - - / 1
- - 0 / 2
2 0 2 / 3
/ 0
```

10 literale, 6 rânduri, fără default

Tab. 9-7 Alegerea eronată a unei ieşiri implicite denaturează sistemul decizional (combinaţia 1 0 1 are numai ieşirea 1) pentru funcţia **012ned**

O altă tehnică de minimizare pusă la dispoziţie de aplicaţie este cea a minimizării în reţea pentru sistemele cu ieşire singulară pe baza valorilor variabilei de ieşire.

Funcţia de generare a specificaţiei cu ieşiri binare care evidenţiază corespondenţa dintre combinaţiile de intrare şi valorile variabilelor de ieşire conduce la:

```
.function 012ned_BE a b c / x(0) x(1) x(2) x(3)
x(4)
//a b c / x(0) x(1) x(2) x(3) x(4)
0 0 1 / 1 0 0 0 0
0 1 0 / 0 0 1 0 0
1 0 1 / 1 1 0 0 0
1 1 1 / 0 1 0 0 0
1 2 1 / 0 1 0 0 0
1 3 1 / 0 1 0 0 0
1 4 1 / 0 1 0 0 0
2 0 1 / 1 0 0 0 0
2 0 2 / 0 0 0 1 0
2 1 2 / 1 0 0 0 0
3 0 1 / 1 0 0 0 0
3 - 2 / 1 0 0 0 0
4 0 1 / 1 0 0 0 0
4 1 0 / 0 0 1 0 0
```

10 literale, 6 rânduri, fără default
Tab. 9-8 Expandarea binară a funcţiei **012ned**

Se poate observa că singura combinaţie de intrare care activează 2 ieşiri este (1 0 1).

Rezultatul minimizării în reţea a funcţiei expandate binar este:

COMIN

```
.function 012ned_BE_k/n_NM_x(4)_sol  / x(4)
// / x(4)
/ 0

.function 012ned_BE_k/n_NM_x(2)_sol c / x(2)
//c / x(2)
0 / 1
/ 0

.function 012ned_BE_k/n_NM_x(1)_sol a / x(1)
//a / x(1)
1 / 1
/ 0

.function 012ned_BE_k/n_NM_x(3)_sol a b c / x(3)
//a b c / x(3)
2 0 2 / 1
/ 0
```

```
.function 012ned_BE_k/n_NM_x(0)_sol b c x(3) /
x(0)
//b c x(3) / x(0)
0 1 - / 1
- 2 0 / 1
/ 0
```

9 literale, 5 rânduri

Tab. 9-9 Minimizarea în reţea cu expandare binară cu COMIN pentru **012ned**

Exemplul 3

În cadrul acestui exemplu este prezentată minimizarea unei funcţii (**vectorned**) neunivoce, incomplet specificată cu ieşire multiplă. Specificaţia iniţială este:

```
.library vectorned
.variable x 0 1
.variable y 0 1
.variable z 0 1
.variable a 0 1 2
.variable b 2 3 4
```

```
.function vectorned x y z / a b
//x y z / a b
0 0 0 / 1 2
0 1 1 / 1 3
0 1 1 / - 4
0 1 1 / 2 4
1 0 0 / 2 2
1 1 1 / 0 -
```

Tab. 9-10 Specificarea funcţiei **vectorned**

Rezultatul minimizării obţinut cu aplicaţiile COMIN şi MVSIS este:

COMIN

```
.function vectorned_k/n_NM_a_Def x y /
a
//x y / a
0 - / 1
1 0 / 2
/ 0
.function vectorned_k/n_NM_b_Def y / b
//y / b
0 / 2
/ {3, 4}
```

4 literale, 3 rânduri, 2 default

MVSIS

```
.table x y z a
.default 1
1 1 1  0
1 - 0  2
.table y z b
.default 2
0 0  0
```

7 literale, 3 rânduri, 2 default

Tab. 9-11 Rezultatele minimizării obţinute cu COMIN şi MVSIS pentru **vectorned**

Valoarea „**în rest**” $\{3,4\}$ obţinută pentru ieşirea b semnifică faptul că poate fi aleasă oricare dintre cele două valori pentru ieşirea implicită a sistemului. Pentru acest sistem, rezultatul minimizării obţinut cu COMIN este cu 42% mai bun decât cel obţinut cu aplicaţia MVSIS.

Sistemul, fiind neunivoc poate fi minimizat şi cu semantică deterministă cu ajutorul aplicaţiei COMIN. Rezultatul obţinut este:

COMIN

```
.function vectorned_n/n_NM_a_Def x y
/ a
//x y / a
0 - / 1
0 1 / {1, 2}
1 0 / 2
/ 0
.function vectorned_n/n_NM_b_Def y /
b
//y / b
0 / 2
/ {3, 4}
```

6 literale, 4 rânduri, 2 default

Tab. 9-12 Rezultatul minimizării deterministe cu COMIN pentru **vectorned**

Pentru combinaţiile de intrare $(01-)$ ieşirea a are valoarea multiplă $\{1,2\}$ iar pentru combinaţiile de intrare $(--\neq 0)$ ieşirea b a sistemului are valoarea $\{3,4\}$. Combinaţia de ieşire (011) va genera, în sistemul minimizat, rezultatul $\{1,2\}\times\{3,4\}$. Conform specificaţiei iniţiale combinaţia (011) nu permitea ieşirea (23), dar, nefiind corelate ieşirile, aceasta se regăseşte în rezultatul obţinut ($(2,3)\in\{1,2\}\times\{3,4\}$).

Aplicaţia permite şi minimizarea cu corelarea ieşirilor, rezultatul fiind:

COMIN

```
.function vectorned_n/n_NMC_a_Def x y b
/ a
//x y b / a
1 1 - / 0
0 - 4 / 2
1 0 - / 2
/ 1
.function vectorned_n/n_NM_b_Def y / b
//y / b
0 / 2
/ {3, 4}
```

7 literale, 4 rânduri, 2 default

Tab. 9-13 Rezultatul minimizării deterministe cu corelarea ieşirilor pentru **vectorned**

În cadrul acestui rezultat, ieşirea a este dependentă de ieşirea b, ceea ce a permis eliminarea valorii de ieşire (23) pentru intrarea (011). Dacă $b=3$ atunci valoarea lui a nu poate fi 2 decât dacă $x=1$ ceea ce elimină valoarea de ieşire (23) pentru intrarea (011).

Rezultatul minimizării în paralel este identic cu cel obţinut pentru minimizarea în reţea deoarece:

- strategia de minimizare în paralel pierde implicit corelările ieşirilor;
- rezultatul minimizării în reţea fără corelări nu a condus la o minimizare înlănţuită (nu există ieşiri care depind de alte ieşiri).

În cadrul pachetului de strategii de minimizare dezvoltate în această lucrare intră şi minimizarea pe valorile variabilelor de ieşire. Sistemul iniţial este transformat în:

COMIN

```
.function vectorned_BE x y z / a(0) a(1) a(2) b(2) b(3)
b(4)
//x y z / a(0) a(1) a(2) b(2) b(3) b(4)
0 0 0 / 0 1 0 1 0 0          // 2 valori de ieşire
0 1 1 / 0 1 1 0 1 1          // 3 valori de ieşire
1 0 0 / 0 0 1 1 0 0          // 2 valori de ieşire
1 1 1 / 1 0 0 - - -
```

Tab. 9-14 Expandarea binară a sistemului **vectorned**

Această reprezentare permite evidenţierea neunivocităţilor. Primele 3 combinaţii de intrare au câte 2,3 şi respectiv 2 valori de ieşire.

Rezultatul minimizării în reţea fără corelarea ieşirilor este:

COMIN

```
.function vectorned_BE_n/n_PM_a(0)_Def x y /
a(0)
//x y / a(0)
1 1 / 1
/ 0

.function  vectorned_BE_n/n_PM_a(1)_Def  x  /
a(1)
//x / a(1)
1 / 0
/ 1

.function vectorned_BE_n/n_PM_a(2)_Def x y /
a(2)
//x y / a(2)
0 1 / 1
1 0 / 1
/0

.function  vectorned_BE_n/n_PM_b(2)_Def  y  /
b(2)
//y / b(2)
0 / 1
/ 0
```

```
.function  vectorned_BE_n/n_PM_b(3)_Def  y  /
b(3)
//y / b(3)
0 / 0
/ 1

.function  vectorned_BE_n/n_PM_b(4)_Def  y  /
b(4)
//y / b(4)
0 / 0
/ 1
```

10 literale, 7 rânduri, 6 default

Tab. 9-15 Rezultatul minimizării pe valorile de ieşire pentru **vectorned**

Activarea funcţiei de corelare conduce la modificarea rezultatului pentru $a(2)$:

COMIN

```
…
.function  vectorned_BE_n/n_NMC_a(2)_Def  x  y
a(0) / a(2)
//x y a(0) / a(2)
0 0 - / 0
- - 1 / 0
/ 1
```

9 literale, 7 rânduri, 6 default

Tab. 9-16 Rezultatul minimizării pe valori de ieşire corelate pentru **vectorned**

Modificarea rezultatului constă în blocarea valenţei 2 pentru ieşirea *a* dacă a fost activată valenţa 0 pentru aceeaşi variabilă.

Pentru a obţine minimizarea pe valenţe cu ajutorul aplicaţiei MVSIS am modificat corespunzător fişierul de intrare:

Date de intrare MVSIS	Rezultat minimizare MVSIS
.table x y z -> a0	.table a2 b4 a0
0 0 0 0	.default 0
0 1 1 0	0 1 1
1 0 0 0	
1 1 1 1	
.table x y z -> a1	.table x a2 b4 a1
0 0 0 1	.default 1
0 1 1 1	1 - - 0
1 0 0 0	- 0 1 0
1 1 1 0	
.table x y z -> a2	.table x y z a2
0 0 0 0	.default 1
0 1 1 1	0 - 0 0
1 0 0 1	- 0 1 0
1 1 1 0	1 1 - 0

```
.table x y z -> b2  .table b4 b2
0 0 0 1             .default 1
0 1 1 0             1  0
1 0 0 1
.table x y z -> b3  .table b4 b3
0 0 0 0             .default 0
0 1 1 1             1  1
1 0 0 0
.table x y z -> b4  .table y z b4
0 0 0 0             .default 0
0 1 1 1             - 1  1
1 0 0 0             1 -  1
```

Fişier de intrare modificat 15 literale, 10 rânduri, 6 default

Tab. 9-17 Rezultatul minimizării pe valorile de ieşire cu MVSIS pentru **vectorned**

Rezultatul obţinut cu aplicaţia dezvoltată în această lucrare este cu 33% mai redus decât cel obţinut cu aplicaţia MVSIS ca număr de literale şi cu 30% mai redus ca număr de cuburi.

Exemplul 4 – Convertor BCD 7 segmente

Funcţia convertorului **BCD 7 segmente** este:

COMIN

```
.l bcd7
.v A, B, C, D 0 1
.v a, b, c, d, e, f, g 0 1
.f bcd7 A B C D / a b c d e f g
0 0 0 0 / 1 1 1 1 1 1 0
0 0 0 1 / 0 1 1 0 0 0 0
0 0 1 0 / 1 0 1 1 0 1 1
0 0 1 1 / 1 0 0 1 1 1 1
0 1 0 0 / 0 1 0 0 1 1 1
0 1 0 1 / 1 1 0 1 1 0 1
0 1 1 0 / 1 1 1 1 1 0 1
0 1 1 1 / 1 0 0 0 1 1 0
1 0 0 0 / 1 1 1 1 1 1 1
1 0 0 1 / 1 1 0 1 1 1 1
```

MVSIS

```
.model bcd7
.inputs A B C D
.outputs a b c d e f g
.table A B C D -> a b c d e f g
0 0 0 0 1 1 1 1 1 1 0
0 0 0 1 0 1 1 0 0 0 0
0 0 1 0 1 0 1 1 0 1 1
0 0 1 1 1 0 0 1 1 1 1
0 1 0 0 0 1 0 0 1 1 1
0 1 0 1 1 1 0 1 1 0 1
0 1 1 0 1 1 1 1 1 0 1
0 1 1 1 1 0 0 0 1 1 0
1 0 0 0 1 1 1 1 1 1 1
1 0 0 1 1 1 0 1 1 1 1
.end
```

Tab. 9-18 Specificarea convertorului **BCD 7 segmente** în COMIN şi MVSIS

Rezultatele minimizării cu cele două aplicaţii de minimizare sunt:

COMIN

```
.function bcd7_n/n_NM_a_Def C d / a
//C d / a
0 0 / 0
/ 1
```

2 literale, 1 rând

MVSIS

```
.table A D d e a
.default 0
- - 1 -  1
1 - - -  1
- 1 - 1  1
```

4 literale, 3 rânduri

`.function bcd7_n/n_NM_b_Def B C D / b`
`//B C D / b`
`- 0 - / 1`
`1 - 0 / 1`
`/ 0`
	`.table A C f b`
	`.default 0`
	`- - 0 1`
	`- 0 - 1`
	`1 - - 1`

3 literale, 2 rânduri	3 literale, 3 rânduri

`.function bcd7_n/n_NM_c_Def B D f d / c`
`//B D f d / c`
`0 - 0 - / 1`
`- 0 - 1 / 1`
`/ 0`
	`.table A B C D d g c`
	`.default 0`
	`1 1 - - - - 1`
	`1 - 1 - - - 1`
	`- - - 0 1 - 1`
	`- - 0 - - 0 1`

4 literale, 2 rânduri	8 literale, 4 rânduri

`.function bcd7_n/n_NM_d_Def B f / d`
`//B f / d`
`0 1 / 1`
`1 0 / 1`
`/ 0`
	`.table A B f g d`
	`.default 0`
	`1 - - - 1`
	`- - 0 1 1`
	`- 0 1 - 1`

4 literale, 2 rânduri	5 literale, 3 rânduri

`.function bcd7_n/n_NM_e_Def D b c / e`
`//D b c / e`
`0 1 - / 1`
`- - 0 / 1`
`/ 0`
	`.table A B C D f e`
	`.default 0`
	`1 - - - - 1`
	`- 1 - - - 1`
	`- - 0 - 1 1`
	`- - - 1 1 1`

3 literale, 2 rânduri	6 literale, 4 rânduri

`.function bcd7_n/n_NM_f_Def A C D b / f`
`//A C D b / f`
`- 0 0 - / 1`
`- - - 0 / 1`
`1 - - - / 1`
`/ 0`
	`.table A B C D g f`
	`.default 0`
	`1 - - - - 1`
	`- 0 - - 1 1`
	`- - 0 0 - 1`
	`- - 1 1 - 1`

4 literale, 4 rânduri	7 literale, 4 rânduri

`.function bcd7_n/n_NM_g_Def A B C b / g`
`//A B C b / g`
`0 0 0 - / 0`
`- 1 - 0 / 0`
`/ 1`
	`.table A B C D g`
	`.default 1`
	`- 1 1 1 0`
	`0 0 0 - 0`

5 literale, 2 rânduri	6 literale, 2 rânduri
Total: 25 literale, 14 randuri	Total: 39 literale, 23 rânduri

Tab. 9-19 Rezultatele minimizării în reţea pentru convertorului **BCD 7 segmente**

Soluţia obţinută cu aplicaţia COMIN este mai redusă cu 35% literale şi 39% rânduri.

Varianta de minimizare în paralel, proprie implementărilor cu adâncime uniformă conduce la rezultatul:

COMIN

```
.function bcd7_a A B C D / a
1 - - - / 1
- 1 - 1 / 1
- - 1 - / 1
- 0 - 0 / 1
/ 0
```

6 literale, 4 rânduri

```
.function bcd7_e A B C D / e
1 - - - / 1
- 1 - - / 1
- - 0 0 / 1
- - 1 1 / 1
/ 0
```

6 literale, 4 rânduri

```
.function bcd7_ b B C D / b
- 0 - / 1
1 - 0 / 1
/ 0
```

3 literale, 2 rânduri

```
.function bcd7_f A B C D / f
- 1 1 0 / 0
0 - 0 1 / 0
/ 1
```

6 literale, 2 rânduri

```
.function bcd7_c A B C D / c
1 - - 1 / 0
- 1 0 - / 0
- - 1 1 / 0
/ 1
```

6 literale, 3 rânduri

```
.function bcd7_g A B C D / g
- 1 1 1 / 0
0 0 0 - / 0
/ 1
```

6 literale, 2 rânduri

```
.function bcd7_d A B C D / d
- 1 0 0 / 0
- 1 1 1 / 0
0 0 0 1 / 0
/ 1
```

10 literale, 3 rânduri

Total: 43 literale, 20 rânduri

Tab. 9-20 Rezultatul minimizării pentru implementare cu adâncime uniformă a convertorului **BCD 7 segmente** în aplicaţia COMIN

Utilizând această strategie de minimizare rezultatul este cu 10% mai slab din punct de vedere al numărului de literale şi cu 13% mai bun din punct de vedere al numărului de rânduri.

Exemplu 5

În acest exemplu prezint rezultatele minimizării pentru o funcţie complet specificată multivalentă univocă cu ieşire singulară:

COMIN

```
.library fdet
.variable a 0 1 2 3 4
.variable b 0 1 2 3 4
.variable c 0 1 2 3 4
.variable x 0 1 2 3 4
```

MVSIS

```
.model fdet
.inputs a b c
.outputs x
.mv a 5 0 1 2 3 4
.mv b 5 0 1 2 3 4
.mv c 5 0 1 2 3 4
.mv x 5 0 1 2 3 4
```

```
.function fdet a b c / x
//a b c / x
2 3 4 / 4
0 4 1 / 3
4 4 1 / 3
1 0 3 / 1
3 1 1 / 0
0 2 0 / 0
4 4 2 / 0
/ 2
```

```
.table a b c -> x
.default 2
2 3 4 4
0 4 1 3
4 4 1 3
1 0 3 1
3 1 1 0
0 2 0 0
4 4 2 0
.end
```

Tab. 9-21 Secificarea sistemului nultivalent univoc cu ieşire singulară **fdet**

Rezultatele minimizărilor cu aplicaţiile COMIN şi MVSIS sunt

COMIN

```
.function fdet_n/n_x_Def a b c / x
//a b c / x
1 - - / 1
- 4 1 / 3
2 - - / 4
/ 0
```

4 literale, 3 rânduri, 1 default

COMIN

```
.table a b c x
.default 0
1 0 3  1
(0,4) 4 1  3
2 3 4  4
.end
```

9 literale, 3 rânduri, 1 default

Tab. 9-22 Rezultatul minimizării pentru **fdet**

Cu aplicaţia COMIN se obţine, suplimentar, minimizarea pentru valenţele variabilelor de ieşire:

COMIN

```
.function fdet_BE_n/n_PM_x(0)_Def a b c / x(0)
//a b c / x(0)
- - 2 / 1
3 - - / 1
- 2 - / 1
/ 0

.function fdet_BE_n/n_PM_x(1)_Def a / x(1)
//a / x(1)
1 / 1
/ 0

.function fdet_BE_n/n_PM_x(2)_Def  / x(2)
// / x(2)
/ 0

.function fdet_BE_n/n_PM_x(3)_Def b c / x(3)
//b c / x(3)
4 1 / 1
/ 0
```

```
.function fdet_BE_n/n_PM_x(4)_Def a / x(4)
//a / x(4)
2 / 1
/ 0
```

7 literale, 6 rânduri, 5 default
Tab. 9-23 Rezultatul minimizării pentru **fdet**

Rezultatul obţinut cu COMIN este superior cu 55% în cazul minimizării fără expandarea funcţiei şi cu 22% în cazul funcţiei expandate.

Exemplul 6

În acest exemplu prezint minimizarea comparativă (COMIN şi MVSIS) pentru un automat sincron:

COMIN

```
.library aut_sincron
.variable x, y, A, B, C 0 1
.variable A*, B*, C*, Ja, Ka, Jb, Kb, Jc, Kb, Ta, Tb, Tc 0 1

.function aut_sincron x y A B C / A* B* C* Ja Ka Jb Kb Jc Kc Ta
Tb Tc
// synchronous automata
// 2 inputs (x, y)
// 3 state variables *A,B,C
// implementations with JK, D and T flip-flops
// A*, B*, C* are the next states of A, B, C,
//            and the values of Da, Db, Dc
//x y A B C / A* B* C* Ja Ka Jb Kb Jc Kc Ta Tb Tc
0 - 0 0 0 / 1 0 1 1 - 0 - 1 - 1 0 1
1 0 0 0 0 / 0 0 0 0 - 0 - 0 - 0 0 0
0 0 1 0 1 / 1 1 1 - 0 1 - - 0 0 1 0
- 1 1 0 1 / 1 0 1 - 0 1 - - 0 0 0 0
1 0 1 0 1 / 0 0 1 - 1 0 - - 0 1 0 0
- 1 0 0 1 / 0 0 0 0 - 0 - - 1 0 0 1
0 0 0 0 1 / 0 0 1 0 - 0 - - 0 0 0 0
1 0 1 1 1 / 0 0 0 - 1 - 1 - 1 1 1 1
0 1 1 1 1 / 0 0 1 - 1 - 1 - 1 1 1 0
0 0 1 1 1 / 1 1 1 - 0 - 0 - 1 0 0 0
0 1 1 0 0 / 1 0 0 - 0 1 - 0 - 0 0 0
1 1 1 0 0 / 1 1 1 - 0 1 - 1 - 0 1 1
- 0 1 0 0 / 0 0 1 - 1 0 - 1 - 1 0 1
```

Tab. 9-24 Specificare COMIN pentru sistemul **aut_sinc**

Tabele separate pe grupe de ieşire, obţinute cu ajutorul funcţiei **FunGen** sunt:

COMIN

.function aut_sincron_1 x y A B C / A* B* C*	.function aut_sincron_2 x y A B C / Ja Ka Jb Kb Jc Kc	.function aut_sincron_3 x y A B C / Ta Tb Tc
//x y A B C / A* B* C*	//x y A B C / Ja Ka Jb Kb Jc Kc	//x y A B C / Ta Tb Tc
- 1 0 0 1 / 0 0 0	0 0 1 1 1 / - 0 - 0 - 1	- 1 1 0 1 / 0 0 0
1 0 0 0 0 / 0 0 0	- 1 1 0 1 / - 0 1 - - 0	0 0 0 0 1 / 0 0 0
1 0 1 1 1 / 0 0 0	0 0 1 0 1 / - 0 1 - - 0	0 0 1 1 1 / 0 0 0
- 0 1 0 0 / 0 0 1	0 1 1 0 0 / - 0 1 - 0 -	0 1 1 0 0 / 0 0 0
0 0 0 0 1 / 0 0 1	1 1 1 0 0 / - 0 1 - 1 -	1 0 0 0 0 / 0 0 0
0 1 1 1 1 / 0 0 1	0 1 1 1 1 / - 1 - 1 - 1	- 1 0 0 1 / 0 0 1
1 0 1 0 1 / 0 0 1	1 0 1 1 1 / - 1 - 1 - 1	0 0 1 0 1 / 0 1 0
0 1 1 0 0 / 1 0 0	1 0 1 0 1 / - 1 0 - - 0	1 1 1 0 0 / 0 1 1
- 1 1 0 1 / 1 0 1	- 0 1 0 0 / - 1 0 - 1 -	1 0 1 0 1 / 1 0 0
0 - 0 0 0 / 1 0 1	0 0 0 0 1 / 0 - 0 - - 0	- 0 1 0 0 / 1 0 1
0 0 1 0 1 / 1 1 1	- 1 0 0 1 / 0 - 0 - - 1	0 - 0 0 0 / 1 0 1
0 0 1 1 1 / 1 1 1	1 0 0 0 0 / 0 - 0 - 0 -	0 1 1 1 1 / 1 1 0
1 1 1 0 0 / 1 1 1	0 - 0 0 0 / 1 - 0 - 1 -	1 0 1 1 1 / 1 1 1

Tab. 9-25 Separarea pe grupe de ieşiri în COMIN pentru sistemul **aut_sinc**

Rezultatele obţinute minimizării în reţea obţinute cu COMIN şi cu MVSIS (după ce am efectuat gruparea în fişiere separate pentru MVSIS) sunt:

COMIN	MVSIS
.function aut_sincron_A* x y A B C B* / A*	.table A B C x y A*
0 - 0 - 0 - / 1	.default 0
- - - - - 1 / 1	1 0 - - 1 1
- 1 1 0 - - / 1	1 - 1 0 0 1
/ 0	0 0 0 0 - 1
7 literale, 6 rânduri	11 literale, 3 rânduri
.function aut_sincron_B* x y A C / B*	.table C x y A* B*
//x y A C / B*	.default 1
0 0 1 1 / 1	1 - 1 - 0
1 1 - 0 / 1	- 0 1 - 0
/ 0	0 - 0 - 0
	- - - 0 0
8 literale, 3 rânduri	7 literale, 4 rânduri
.function aut_sincronC x y A B C B* A* / C*	.table A B C x y A* C*
//x y A B C B* A* / C*	.default 1
- - 1 - 0 0 1 / 0	0 1 - - - - 0
- 1 0 - 1 - - / 0	- 1 0 - - - 0
1 - 0 - - - - / 0	1 - 0 0 - 1 0
1 - - 1 - - - / 0	- 1 - 1 - - 0
/ 1	0 - - 1 - - 0
	0 - - - 1 0 0
11 literale, 4 rânduri	15 literale, 6 rânduri

```
.function aut_sincron_Ja x C / Ja
0 0 / 1
/ 0
```

```
.table A B C x Ja
.default 1
- - - 1  0
- - 1 -  0
- 1 - -  0
1 - - -  0
```

2 literale, 1 rând | 4 literale, 4 rânduri

```
.function aut_sincron_Ka B Kb Jb / Ka
//B Kb Jb / Ka
0 - 0 / 1
1 1 - / 1
/ 0
```

```
.table A B C x y Ka
.default 0
0 - - - -  1
- 1 - - 1  1
- - 0 - 0  1
- - - 1 0  1
```

4 literale, 2 rânduri | 7 literale, 4 rânduri

```
.function aut_sincron_Jb x y A C / Jb
0 - 1 1 / 1
- 1 1 - / 1
/ 0
```

```
.table B Ka Jb
.default 1
- 1  0
1 -  0
```

5 literale, 2 rânduri | 2 literale, 2 rânduri

```
.function aut_sincron_Kb x y / Kb
0 0 / 0
/ 1
```

```
.table y Ka Kc Kb
.default 0
- - 0  1
- 1 -  1
1 - -  1
```

2 literale, 1 rând | 3 literale, 3 rânduri

```
.function aut_sincron_Jc x A Ja Jb / Jc
0 - - 1 / 0
- 0 0 - / 0
/ 1
```

```
.table A B C x Ja Ka Jc
.default 1
- - 1 - - -  0
- 1 - - - -  0
- - - 0 - 0  0
0 - - - 0 -  0
```

4 literale, 2 rânduri | 6 literale, 4 rânduri

```
.function aut_sincron_Kc y A B / Kc
- - 1 / 1
1 0 - / 1
/ 0
```

```
.table A B C x y Ka Kc
.default 0
- - 0 - - -  1
- 1 - - - -  1
- - - - 1 1  1
0 - - 1 - -  1
```

3 literale, 2 rânduri | 6 literale, 4 rânduri

```
.function aut_sincron_Ta x y A B C / Ta
0 0 - - 1 / 0
- - 0 - 1 / 0
- 1 1 0 - / 0
1 - 0 - - / 0
/ 1
```

```
.table A B C x y Ta
.default 0
- 1 - - 1  1
0 1 - - -  1
0 - 0 - 1  1
1 - - 1 0  1
- - 1 1 0  1
- - 0 0 0  1
```

10 literale, 4 rânduri | 16 literale, 6 rânduri

```
.function aut_sincron_Tb y A B Ta Tc /
Tb
- 0 - - - / 0
- - 0 1 - / 0
- - 1 0 - / 0
1 - 0 - 0 / 0
/ 1
```

8 literale, 4 rânduri

```
.table A B C x y Ta
Tb
.default 0
- - 0 1 1 -  1
- 1 - - - 1  1
0 - 1 - - 1  1
1 0 - - 0 0  1
```

12 literale, 4 rânduri

```
.function aut_sincron_Tc x y A B C Ta /
Tc
0 1 1 - - - / 0
- 0 - - - 0 / 0
- - 1 0 1 - / 0
/ 1
```

8 literale, 3 rânduri

```
.table A B C x y Ta
Tc
.default 0
0 - - - 1 -  1
0 - - - - 1  1
- - 0 - - 1  1
- 1 - 1 - -  1
- - 0 1 1 -  1
```

11 literale, 5 rânduri

Total: 72 literale, 34 rânduri

Total: 100 literale, 49 rânduri

Tab. 9-26 Rezolvarea sistemului **aut_sincron** cu aplicaţiile COMIN şi MVSIS

Rezultatele obţinute cu COMIN sunt mai bune cu 28% din punct de vedere al numărului de literale şi cu 30% din punct de vedere al numărului de rânduri

Exemplul 7

În acest exemplu prezint un sistem pentru conversia din Z2 în Z3 a valorilor de la 0 la 9 exprimate pe 4 biţi. Sistemul are 4 intrări binare şi 3 ieşiri ternare. Funcţia este univocă incomplet specificată. Reprezentarea acesteia este:

COMIN

```
.library z23
.variable A 0 1
.variable B 0 1
.variable C 0 1
.variable D 0 1
.variable a 0 1
.variable b 0 1 2
.variable c 0 1 2
.function z23 A B C D / a b c
0 0 0 0 / 0 0 0
0 0 0 1 / 0 0 1
0 0 1 0 / 0 0 2
0 0 1 1 / 0 1 0
0 1 0 0 / 0 1 1
0 1 0 1 / 0 1 2
0 1 1 0 / 0 2 0
```

MVSIS

```
.model z23
.inputs A B C D
.outputs a b c
.mv A, B, C, D 2 0 1
.mv a, b, c 3 0 1 2

.table A B C D -> a b c
0 0 0 0 0 0 0
0 0 0 1 0 0 1
0 0 1 0 0 0 2
0 0 1 1 0 1 0
0 1 0 0 0 1 1
0 1 0 1 0 1 2
0 1 1 0 0 2 0
0 1 1 1 0 2 1
1 0 0 0 0 2 2
```

```
0 1 1 1 / 0 2 1                1 0 0 1 1 0 0
1 0 0 0 / 0 2 2                .end
1 0 0 1 / 1 0 0
```

Tab. 9-27 Sistem de conversie **z23** din Z2 în Z3

Rezultatele minimizării cu cele aplicaţiile COMIN şi MVSIS sunt:

COMIN reţea	MVSIS

`.function z23_k/n_NM_a_sol A D / a`
`1 1 / 1`
`/ 0`
	`.table A B C D a`
	`.default 1`
	`0 - - -  0`
	`- 0 0 0  0`

2 literale, 1 rând	4 literale, 2 rânduri

`.function z23_k/n_NM_b_sol A B C D a / b`
`- 0 1 1 - / 1`
`- 1 0 - - / 1`
`- 1 1 - - / 2`
`1 - - - 0 / 2`
`/ 0`
	`.table B C D a c b`
	`.default 2`
	`0 1 0 - -  0`
	`0 0 - - (0,1)  0`
	`- - - (1,2) -  0`
	`- 1 - (1,2) -  1`
	`- 1 1 - (0,2)  1`
	`1 0 - - -  1`

9 literale, 4 rânduri	14 literale, 6 rânduri

`.function z23_k/n_NM_c_sol A B C D b / c`
`0 - - 1 0 / 1`
`- - - 0 1 / 1`
`- - - 1 2 / 1`
`- - 1 - 0 / 2`
`- 1 0 1 - / 2`
`1 - - 0 - / 2`
`/ 0`
	`.table A B C D a c`
	`.default 0`
	`- 1 1 1 0  1`
	`0 1 0 0 0  1`
	`- 0 0 1 0  1`
	`- 1 0 1 -  2`
	`1 - - 0 -  2`
	`- 0 1 0 -  2`

14 literale, 6 rânduri	21 literale, 6 rânduri
Total: 25 literale, 11 rânduri	Total: 39 literale, 14 rânduri

Tab. 9-28 Rezolvarea sistemului **z23** cu aplicaţiile COMIN şi MVSIS

Rezultatul obţinut cu aplicaţia COMIN este mai bun cu 35% ca număr de literale şi cu 21% ca număr de rânduri.

Minimizarea în paralel (fără conectarea în reţea a ieşirilor) conduce la rezultatul:

COMIN paralel	COMIN reţea

`.function z23_k/n_PM_a A D / a`
`1 1 / 1`
`/ 0`
	`.function z23_k/n_NM_a A D / a`
	`1 1 / 1`
	`/ 0`

2 literale, 1 rând	2 literale, 1 rând

`.function z23_k/n_PM_b A B C D/b`
`- 0 1 1 / 1`
`- 1 0 - / 1`
`- 1 1 - / 2`
`1 - - 0 / 2`
`/ 0`
	`.function z23_k/n_NM_b A B C D a/b`
	`- 0 1 1 - / 1`
	`- 1 0 - - / 1`
	`- 1 1 - - / 2`
	`1 - - - 0 / 2`
	`/ 0`

9 literale, 4 rânduri	9 literale, 4 rânduri

```
.function z23_k/n_PM_c A B C D
/ c
0 0 0 1 / 1
- 1 0 0 / 1
- 1 1 1 / 1
- 0 1 0 / 2
- 1 0 1 / 2
1 - - 0 / 2
/ 0
```

18 literale, 6 rânduri
Total: 29 literale, 11 rânduri

```
.function z23_k/n_NM_c A B C D
b/c
0 - - 1 0 / 1
- - - 0 1 / 1
- - - 1 2 / 1
- - 1 - 0 / 2
- 1 0 1 - / 2
1 - - 0 - / 2
/ 0
```

14 literale, 6 rânduri
Total: 25 literale, 11 rânduri

Tab. 9-29 Rezolvarea sistemului **z23** cu aplicaţia COMIN paralel şi în reţea

Rezultatul minimizării în paralel este cu 16% mai slab decât rezultatul minimizării în reţea (minimizările au fost efectuate cu aplicaţia COMIN)

Minimizarea pe valenţele variabilelor de ieşire porneşte de la tabelul expandat:

```
.function z23_BE A B C D / a(0) a(1) b(0) b(1) b(2) c(0) c(1)
c(2)
//A B C D / a(0) a(1) b(0) b(1) b(2) c(0) c(1) c(2)
0 0 0 0 / 1 0 1 0 0 1 0 0
0 0 0 1 / 1 0 1 0 0 0 1 0
0 0 1 0 / 1 0 1 0 0 0 0 1
0 0 1 1 / 1 0 0 1 0 1 0 0
0 1 0 0 / 1 0 0 1 0 0 1 0
0 1 0 1 / 1 0 0 1 0 0 0 1
0 1 1 0 / 1 0 0 0 1 1 0 0
0 1 1 1 / 1 0 0 0 1 0 1 0
1 0 0 0 / 1 0 0 0 1 0 0 1
1 0 0 1 / 0 1 1 0 0 1 0 0
```

Tab. 9-30 Expandarea sistemului **z23** pe valenţele variabilelor de ieşire

Rezultatele obţinute cu COMIN pentru minimizările în paralel şi în reţea sunt:

COMIN paralel

```
.f z23_BE_PM_a(0) A D / a(0)
1 1 / 0
/ 1
```

2 literale, 1 rând

```
.f z23_BE_PM_a(1) A D / a(1)
1 1 / 1
/ 0
```

2 literale, 1 rând

```
.f z23_BE_PM_b(0) A B C D /
b(0)
- 1 - - / 0
1 - - 0 / 0
- - 1 1 / 0
/ 1
```

5 literale, 3 rânduri

COMIN reţea

```
.f z23_BE_NM_a(0) A D / a(0)
1 1 / 0
/ 1
```

2 literale, 1 rând

```
.f z23_BE_NM_a(1) a(0) / a(1)
0 / 1
/ 0
```

1 literal, 1 rând

```
.f z23_BE_NM_b(0) B C D b(2) /
b(0)
- - - 1 / 0
- 1 1 - / 0
1 - - - / 0
/ 1
```

4 literale, 3 rânduri

.f z23_BE_PM_b(1) B C D / b(1) 1 0 - / 1 0 1 1 / 1 / 0	.f z23_BE_NM_b(1) b(2) b(0) / b(1) 0 0 / 1 / 0
5 literale, 2 rânduri	2 literale, 1 rând
.f z23_BE_PM_b(2) A B C D / b(2) - 1 1 - / 1 1 - - 0 / 1 / 0	.f z23_BE_NM_b(2) A B C a(0) / b(2) - 1 1 - / 1 1 - - 1 / 1 / 0
4 literale, 2 rânduri	4 literale, 2 rânduri
.f z23_BE_PM_c(0) A B C D / c(0) - 1 1 0 / 1 - 0 1 1 / 1 0 0 0 0 / 1 1 - - 1 / 1 / 0	.f z23_BE_NM_c(0) C b(0) c(1) / c(0) 0 0 - / 0 - - 1 / 0 1 1 - / 0 / 1
12 literale, 4 rânduri	5 literale, 3 rânduri
.f z23_BE_PM_c(1) A B C D / c(1) - 1 1 1 / 1 0 0 0 1 / 1 - 1 0 0 / 1 / 0	.f z23_BE_NM_c(1) D a(0) b(1) / c(1) 0 - 1 / 1 1 1 0 / 1 / 0
10 literale, 3 rânduri	5 literale, 2 rânduri
.f z23_BE_PM_c(2) A B C D / c(2) - 0 - 1 / 0 - 1 1 - / 0 0 - 0 0 / 0 / 1	.f z23_BE_NM_c(2) c(1) c(0) / c(2) 0 0 / 1 / 0
7 literale, 3 rânduri	2 literale, 1 rând
Total: 43 literale, 19 rânduri	Total: 14 literale, 14 rânduri

Tab. 9-31 Rezultatele minimizărilor în paralel şi în reţea cu COMIN pentru **z23** pe valorile variabilelor de ieşire

Minimizarea în reţea conduce la un sistem redus cu 67% pentru literale şi cu 26% pentru rânduri faţă de minimizarea în paralel. Minimizarea în reţea expandata este mai bună cu 56% din punct de vedere al literalelor şi mai slabă cu 21% din punct de vedere al numărului de rânduri decât expandarea în reţea fără expandare pe valenţele variabilelor.

Exemplul 8

Acest exemplu conţine o specificare de unitate aritmetica şi logică cu operanzi din domeniul Z_4. Specificarea în COMIN este:

```
.library alu4
.variable a       0 1 2 3
.variable b       0 1 2 3
.variable and     0 1 2 3
.variable or      0 1 2 3
.variable xor     0 1 2 3
.variable sum1    0 1 2 3
.variable sum     0 1 2 3
.variable out     0 1 2 3
.variable control 0 1 2 3

.variable carryin  0 1
.variable carryout 0 1

.function  sel  control  or
and xor sum / out
0 1 - - - / 1
0 2 - - - / 2
0 3 - - - / 3
1 - 1 - - / 1
1 - 2 - - / 2
1 - 3 - - / 3
2 - - 1 - / 1
2 - - 2 - / 2
2 - - 3 - / 3
3 - - - 1 / 1
3 - - - 2 / 2
3 - - - 3 / 3
/ 0
```

```
.function calc a b carryin / or and
xor sum1 sum carryout
0 0 0 / 0 0 0 0 0 0
0 1 0 / 1 0 1 1 1 0
0 2 0 / 2 0 2 2 2 0
0 3 0 / 3 0 3 3 3 0
1 0 0 / 1 0 1 1 1 0
1 1 0 / 1 1 0 2 2 0
1 2 0 / 2 1 2 3 3 0
1 3 0 / 3 1 3 0 0 1
2 0 0 / 2 0 2 2 2 0
2 1 0 / 2 1 2 3 3 0
2 2 0 / 2 2 0 0 0 1
2 3 0 / 3 2 3 1 1 1
3 0 0 / 3 0 3 3 3 0
3 1 0 / 3 1 3 0 0 1
3 2 0 / 3 2 3 1 1 1
3 3 0 / 3 3 0 2 2 1
0 0 1 / 0 0 0 0 1 0
0 1 1 / 1 0 1 1 2 0
0 2 1 / 2 0 2 2 3 0
0 3 1 / 3 0 3 3 0 1
1 0 1 / 1 0 1 1 2 0
1 1 1 / 1 1 0 2 3 0
1 2 1 / 2 1 2 3 0 1
1 3 1 / 3 1 3 0 1 1
2 0 1 / 2 0 2 2 3 0
2 1 1 / 2 1 2 3 0 1
2 2 1 / 2 2 0 0 1 1
2 3 1 / 3 2 3 1 2 1
3 0 1 / 3 0 3 3 0 1
3 1 1 / 3 1 3 0 1 1
3 2 1 / 3 2 3 1 2 1
3 3 1 / 3 3 0 2 3 1
```

Tab. 9-32 Specificare în COMIN pentru **ALU4**

Rezultatele obţinute cu COMIN pentru minimizările în paralel şi în reţea sunt:

COMIN reţea

```
.function  calc_n/n_NMC_or  a  b
/ or
0 0 / 0
0 1 / 1
1 0 / 1
1 1 / 1
- 3 / 3
3 - / 3
/ 2
```

10 literale, 6 rânduri

MVSIS

```
.table a b sum1 xor or
.default 3
0 (0,1,2) (0,2,3) (0,1,3)  0
(0,1,2) (0,1,2) (1,2,3) (0,1,3)
1
- - - (2,3)  2
- 2 (0,2,3) -  2
```

11 literale, 4 rânduri

```
.function calc_n/n_NMC_and a b
/ and
0 - / 0
- 0 / 0
2 2 / 2
2 3 / 2
3 2 / 2
3 3 / 3
/ 1
```

10 literale, 6 rânduri

```
.table a b or sum1 xor and
.default 0
- 1 - - (0,2,3)  1
1 - - (0,2,3) -  1
(0,2,3) (2,3) - (0,1) -  2
- (0,2,3) (0,1,3) 2 -  3
```

10 literal, 4 rânduri

```
.function  calc_n/n_NMC_xor  or
and / xor
1 0 / 1
2 0 / 2
2 1 / 2
3 0 / 3
3 1 / 3
3 2 / 3
/ 0
```

12 literale, 6 rânduri

```
.table a b sum1 xor
.default 0
(0,1,3) (0,1,3) 1  1
(0,2) (0,1,2) (2,3)  2
(0,1,2) (0,2) (2,3)  2
```

9 literale, 3 rânduri

```
.function
calc_n/n_NM_carryout_Def
carryin or and sum1 / carryout
//carryin   or   and   sum1   /
carryout
- - 2 - / 1
- - 3 - / 1
- 3 1 - / 1
1 - - 3 / 1
/ 0
```

6 literale, 4 rânduri

```
.table and carryin or sum1 xor
carryout
.default 0
(1,2,3) - (0,2,3) - (0,1,3)  1
- 1 - (1,3) (0,2,3)  1
```

6 literale, 2 rânduri

```
.function  calc_n/n_NMC_sum1  a
b or and xor / sum1
//a b or and xor / sum1
- - - - 1 / 1
- - 3 2 - / 1
- - - 3 - / 2
- - 2 0 - / 2
1 1 - - - / 2
- - 2 1 - / 3
- - 3 0 - / 3
/ 0
```

12 literale, 7 rânduri

```
.table a b sum1
.default 0
0 1  1
1 0  1
2 3  1
3 2  1
0 2  2
1 1  2
3 3  2
2 0  2
0 3  3
1 2  3
2 1  3
3 0  3
```

24 literale, 12 rânduri

```
.function      calc_n/n_NMC_sum
carryin sum1/sum
0 1 / 1
1 0 / 1
0 2 / 2
1 1 / 2
0 3 / 3
1 2 / 3
/ 0
```

12 literale, 6 rânduri

```
.table carryin sum1 sum
.default 0
0 1  1
1 0  1
0 2  2
1 1  2
0 3  3
1 2  3
```

12 literale, 6 rânduri

```
.function      sel_n/n_NMC_out
control or and xor sum / out
//control or and xor sum / out
0 1 - - - / 1
1 - 1 - - / 1
2 - - 1 - / 1
3 - - - 1 / 1
0 2 - - - / 2
1 - 2 - - / 2
2 - - 2 - / 2
3 - - - 2 / 2
0 3 - - - / 3
1 - 3 - - / 3
2 - - 3 - / 3
3 - - - 3 / 3
/ 0
```

24 literale, 12 rânduri

```
.table and control or sum xor
out
.default 0
1 1 - - -  1
(0,1,3) 0 1 - -  1
- 3 - 1 -  1
- (0,2) - - (1,3)  1
2 1 - - -  2
- 3 - 2 -  2
- 0 2 - -  2
- (0,2) - - (2,3)  2
3 (0,1) - - -  3
- 0 3 - -  3
- 3 - 3 -  3
```

23 literale, 11 rânduri

Total: 86 literale, 47 rânduri | Total: 95 literale, 42 rânduri

Tab. 9-33 Rezultatele minimizărilor în paralel şi în reţea cu COMIN pentru **ALU4**

Rezultatul obţinut cu COMIN este mai bun cu 9,4% din punct de vedere al literalelor şi mai slab cu 12% din punct de vedere al rândurilor decât MVSIS.

Tabelul de mai jos conţine rezultatul minimizării în reţea pe valenţele variabilelor de ieşire cu COMIN:

COMIN în reţea pe valorile variabilelor de ieşire

```
.function
calc_BE_n/n_NMC_or(0)  a  b  /
or(0)
0 0 / 1
/ 0
```

2 literale, 1 rând

```
.function
calc_BE_n/n_NMC_and(0)  a  b  /
and(0)
0 - / 1
- 0 / 1
/ 0
```

2 literale, 2 rânduri

```
.function
calc_BE_n/n_NMC_or(1)    a    b
or(0) / or(1)
- - 1 / 0
- 2 - / 0
- 3 - / 0
2 - - / 0
3 - - / 0
/ 1
```

5 literale, 5 rânduri

```
.function
calc_BE_n/n_NMC_and(1)   and(3)
and(2) and(0) / and(1)
- - 1 / 0
- 1 - / 0
1 - - / 0
/ 1
```

3 literale, 3 rânduri

```
.function
calc_BE_n/n_NMC_or(2)    a    b
xor(2) / or(2)
- - 1 / 1
2 2 - / 1
/ 0
```

3 literale, 2 rânduri

```
.function
calc_BE_n/n_NMC_and(2)   a    b
and(3) / and(2)
0 - - / 0
- 0 - / 0
- - 1 / 0
- 1 - / 0
1 - - / 0
/ 1
```

5 literale, 5 rânduri

```
.function
calc_BE_n/n_NMC_or(3)  a  b  /
or(3)
- 3 / 1
3 - / 1
/ 0
```

2 literale, 2 rânduri

```
.function
calc_BE_n/n_NMC_and(3) a  b  /
and(3)
3 3 / 1
/ 0
```

2 literale, 1 rând

```
.function
calc_BE_n/n_NMC_xor(0)   xor(1)
xor(2) xor(3) / xor(0)
- - 1 / 0
- 1 - / 0
1 - - / 0
/ 1
```

3 literale, 3 rânduri

```
.function
calc_BE_n/n_NMC_sum1(0)   a   b
or(0) and(1) or(3) / sum1(0)
- - - 1 1 / 1
- - 1 - - / 1
2 2 - - - / 1
/ 0
```

5 literale, 3 rânduri

```
.function
calc_BE_n/n_NMC_xor(1)   a   b
or(1) / xor(1)
- - 0 / 0
1 1 - / 0
/ 1
```

3 literale, 2 rânduri

```
.function
calc_BE_n/n_NMC_sum1(1)  xor(1)
and(2) or(2) / sum1(1)
- 1 0 / 1
1 - - / 1
/ 0
```

3 literale, 2 rânduri

```
.function
calc_BE_n/n_NMC_xor(2)    a    b
and(2) / xor(2)
- 2 0 / 1
2 - 0 / 1
/ 0
```

4 literale, 2 rânduri

```
.function
calc_BE_n/n_NMC_sum1(2)
sum1(1)   sum1(3)   sum1(0)   /
sum1(2)
- - 1 / 0
- 1 - / 0
1 - - / 0
/ 1
```

3 literale, 3 rânduri

```
.function
calc_BE_n/n_NMC_xor(3)   and(3)
or(3) / xor(3)
0 1 / 1
/ 0
```

2 literale, 1 rând

```
.function
calc_BE_n/n_NMC_sum1(3)  and(0)
xor(2) or(3) / sum1(3)
0 1 - / 1
1 - 1 / 1
/ 0
```

4 literale, 2 rânduri

```
.function
calc_BE_n/n_NMC_sum(0)  carryin
sum1(3) sum1(0) / sum(0)
0 - 1 / 1
1 1 - / 1
/ 0
```

4 literale, 2 rânduri

```
.function
calc_BE_n/n_NMC_sum(2)  carryin
sum1(1) sum1(2) / sum(2)
0 - 1 / 1
1 1 - / 1
/ 0
```

4 literale, 2 rânduri

```
.function
calc_BE_n/n_NMC_sum(1)  carryin
sum1(1) sum1(0) / sum(1)
0 1 - / 1
1 - 1 / 1
/ 0
```

4 literale, 2 rânduri

```
.function
calc_BE_n/n_NMC_sum(3)   sum(0)
sum(2) sum(1) / sum(3)
- - 1 / 0
- 1 - / 0
1 - - / 0
/ 1
```

3 literale, 3 rânduri

```
.function
calc_BE_n/n_NMC_carryout(0)
and(3)   or(1)   and(0)   or(3)
sum(3) / carryout
0 - - - 1 / 0
- - 1 0 - / 0
- 1 - - - / 0
/ 1
```

5 literale, 3 rânduri

Total: 71 literale, 51 rânduri

Tab. 9-34 Rezultatele minimizărilor cu COMIN în reţea pe valenţele variabilelor de ieşire pentru **ALU4**

Pentru aceleaşi funcţii minimizarea cu MVSIS a condus la un rezultat format din 72 de literale şi 31 de rânduri. În acest caz COMIN este mai bun decât MVSIS cu 1,3 ca număr de literale şi mai slab cu 13% ca număr de rânduri (fără a considera funcţia de selecţie **control**). Numărul de literale apropiat din cele două soluţii este justificat de faptul că sistemul decizional este complet specificat

Exemplul 9

Specificaţia de mai jos conţine o funcţie binară cu 50 de intrări şi 10 ieşiri binare a cărei corespondenţă este dată prin 50 de rânduri.

```
.library 50x10x50
.variable i1, i2, i3, i4, i5, i6, i7, i8, i9, i10, i11, 12, i13,
i14, i15, i16, i17, i18, i19, i20, i21, i22, i23, i24, i25, i26,
i27, i28, i29, i30, i31, i32, i33, i34, i35, i36, i37, i38, i39,
i40, i41, i42, i43, i44, i45, 46, i47, i48, i49, i50 0 1
.variable o1, o2, op3, o4, o5, o6, o7, o8, o9, o10 0 1

.function 50x10x50 i1 i2 i3 i4 i5 i6 i7 i8 i9 i10 i11 i12 i13
i14 i15 i16 i17 i18 i19 i20 i21 i22 i23 i24 i25 i26 i27 i28 i29
i30 i31 i32 i33 i34 i35 i36 i37 i38 i39 i40 i41 i42 i43 i44 i45
i46 i47 i48 i49 i50 / o1 o2 o3 o4 o5 o6 o7 o8 o9 o10
```

Tab. 9-35 Specificarea funcţiei **50x10x50** în COMIN

Rezultatele minimizării în reţea cu aplicaţiile COMIN şi MVSISsunt:

COMIN	MVSIS
.f 50x10x50_n/n_NM_o1 i2 i4 i13 i24 i26 i31 i34 i40 i44 / o1 //i2 i4 i13 i24 i26 i31 i34 i40 i44 / o1 0 - - - - - - - 0 / 0 - - 0 1 - - - - - / 0 - - - - 0 1 - - - / 0 - - - - - - - 0 1 - / 0 - 1 - - 0 - - - - / 0 / 1	.table i17 i23 i30 i4 i6 i7 i8 i9 o1 .default 0 - 1 - - 0 1 - - 1 - 0 0 - 0 0 1 - 1 - 0 - - - 0 1 1 1 - 1 - - - 1 0 - 1 - 1 - - 0 - 0 0 1 - 1 - 0 - - 0 0 1 1 - - - - 1 0 0 1
10 literale, 5 rânduri	27 literale, 7 rânduri
.f 50x10x50_n/n_NM_o2 i8 i12 i16 i27 i28 i33 i38 i41 i42 i49 / o2 //i8 i12 i16 i27 i28 i33 i38 i41 i42 i49 / o2 0 - - - - 0 0 - - - / 1 0 - - 1 - - - 1 - - / 1 - - 1 - - - - - 1 1 / 1 - 1 - - 0 - - - - - / 1 / 0	.table i2 i22 i28 i29 i32 i44 i7 i8 i9 o2 .default 0 0 0 - 0 - - - - - 1 - 0 0 - - - - - 0 1 - - 1 0 1 - - - - 1 1 1 - - - 1 - - 0 1 0 - 0 - - - 0 - - 1 0 1 - 1 - - - - 1 1 1 1 - - - - - 0 1 1 1 - 0 - 1 - - 0 1 1
11 literale, 4 rânduri	29 literale, 8 rânduri
.f 50x10x50_n/n_NM_o3 i2 i5 i6 i16 i20 i22 i39 i41 i43 i49 o1 o8 / o3 //i2 i5 i6 i16 i20 i22 i39 i41 i43 i49 o1 o8 / o3 - - - - - - 0 - 1 - 0 - / 1 - - - 1 - 1 - - - 0 - - / 1 - 1 0 - 0 - - - - - - - / 1 1 - - - - - - 0 - - - 1 / 1 / 0	.table i1 i43 i45 i47 i50 i6 i8 i9 o3 .default 0 - 0 - 1 1 0 - - 1 0 0 - 1 - - 0 0 1 - - 1 0 0 - - - 1 - 1 - - - 0 1 0 1 - - - 0 1 - 1 - 1 - 1 - 0 - - 1 - 1 - 1 - - 1 - - 1 1 - 1 - - - - 0 1 1
12 literale, 4 rând	28 literale, 8 rânduri
.f 50x10x50_n/n_NM_o4 i9 i17 i20 i22 i28 i32 i33 i36 i40 / o4 //i9 i17 i20 i22 i28 i32 i33 i36 i40 / o4 0 - - - 1 - - 1 - / 1 - 0 - 0 - - - - 1 / 1 - - - - - - 0 1 - - / 1 1 - 1 - - - - - - - / 1 / 0	.table i15 i20 i29 i32 i38 i6 i8 i9 o4 .default 0 - 1 1 1 1 - - - 1 - 1 - - - - - - 1 1 1 - - 0 - - - - 1 - 0 - 0 - 1 - - 1 - 0 - 1 - 0 - - 1 - - 0 1 0 0 - - 1 - - - 0 1 - 0 - 1 - 0 - 1 - - 1 0 1
10 literale, 4 rând	25 literale, 8 rânduri

```
.f 50x10x50_n/n_NM_o5 i2 i4 i10
i13 i23 i41 i43 i46 o4 o1 o7 / o5
//i2 i4 i10 i13 i23 i41 i43 i46
o4 o1 o7 / o5
- - 0 - - 1 1 - - - - / 1
- - - 1 - - - 0 - 0 - / 1
- 1 - - 1 - - - - - 0 / 1
1 0 - - - - - - - 1 - - / 1
/ 0
```

12 literale, 4 rânduri

```
.table i2 i21 i22 i35 i43 i47
i49 i6 i7 i8 o5
.default 1
- - 1 0 - - 1 - - -  0
- 0 - - - 0 - - 0 1  0
1 - - - - - - - 1 0 1  0
- - 0 - - - - - 1 1  0
- 1 1 - 1 - - - - 0  0
- 1 - - - - - - 0 0 0  0
0 0 - - - - - - - - 0  0
0 - - - - 0 - - - 0  0
0 - - 0 - - 0 - - 0  0
- 0 - - - - - - - 1 0  0
```

35 literale, 10 rânduri

```
.f 50x10x50_n/n_NM_o6 i9 i10 i11
i15 i17 i20 i23 i33 i36 i43 i45
o9/o6
//i9 i10 i11 i15 i17 i20 i23 i33
i36 i43 i45 o9 / o6
- 0 1 1 - - - - - - - - / 1
- - - - 0 - - 1 - - - 1 / 1
- - - - - 1 - - - 0 1 - / 1
1 - - - - - 0 - 0 - - - / 1
/ 0
```

12 literale, 4 rânduri

```
.table i19 i20 i22 i30 i32 i4
i45 i47 i7 i8 i9 o6
.default 1
1 - 1 - 1 - - - - - -  0
0 1 0 - 1 - - - - - -  0
0 1 1 - - - 0 - - - -  0
- 0 - 1 0 - - - - - -  0
- 0 0 - 0 1 - - - - -  0
- 0 - 1 - 0 - - - - -  0
- - - 0 - 1 0 1 - - -  0
1 - - 0 - 1 - 1 - - -  0
0 - - 0 - 0 - - 1 - -  0
1 - 1 - - - - - - 0 -  0
- - - 1 - 1 - 0 - - 1  0
```

40 literale, 11 rânduri

```
.f 50x10x50_n/n_NM_o7 i4 i6 i17
i20 i22 i33 i34 i35 i36 i49 o9 /
o7
//i4 i6 i17 i20 i22 i33 i34 i35
i36 i49 o9 / o7
- 0 - - - - - 0 - 0 - / 0
- - - 0 - - 0 - - - 0 / 0
- - 1 - - 1 - - - - - / 0
1 - - - 0 - - - 0 - - / 0
/ 1
```

10 literale, 4 rânduri

```
.table i12 i20 i21 i22 i24
i26 i46 i8 i9 o7
.default 1
1 1 0 1 - - - - - -  0
- - - 0 - - 0 0 -  0
1 - - 1 - - - 0 1  0
0 - - 0 - - - - 1  0
1 0 1 - - - - 1 0  0
0 - 1 - - - - 0 0  0
0 0 - 1 - - - - 0  0
- - - - 1 - - 0 0  0
- 0 - - - 1 - 0 0  0
```

34 literale, 9 rânduri

```
.f 50x10x50_n/n_NM_o8 i6 i18 i20
i22 i25 i27 i28 i34 i35 i49 o5 /
o8
//i6 i18 i20 i22 i25 i27 i28 i34
i35 i49 o5 / o8
0 - - - 0 - - - 0 - - / 1
- - 1 - - - 1 - - - 1 / 1
```

```
.table i20 i21 i24 i26 i31
i46 i8 i9 o8
.default 1
0 - - 0 - - 1 -  0
1 - - - 1 1 1 -  0
1 1 0 - - - 0 -  0
- - 0 - 0 - 0 -  0
```

- 1 - 0 - - - - - - 1 - / 1 1 - - - - 0 - 0 - - - / 1 / 0	0 - - 1 - - 0 - 0 - 1 - - - - - 0 1 0 0 0 - - - - - - 0 0 1 1 - 1 - - 1 0 0
12 literale, 4 rânduri	28 literale, 8 rânduri
.f 50x10x50_n/n_NM_o9_sol i9 i25 i34 i36 i40 i41 i43 i48 o2 / o9 //i9 i25 i34 i36 i40 i41 i43 i48 o2 / o9 0 - - 1 - - - 0 - / 0 - 0 - - 1 - - - 0 / 0 - - 0 - - 1 0 - - / 0 / 1	.table i19 i20 i21 i23 i36 i48 i6 i7 i8 i9 o9 .default 1 1 - 0 - 1 - - - - 0 0 - - - 0 1 1 - - - 1 0 - - 0 - - - 0 1 - - 0 - - - - 0 - 1 1 - 1 0 - - 1 - - - 0 - 1 1 0 - 1 1 - - - 1 - 1 0 0 0 - 0 - - - 0 - 1 0 0 - - - - - 0 1 - 1 0 0 - - - - 1 - 0 0 1 0 0 - - - - 1 - 0 - 0 1 0 - - - - 1 - 1 - 0 0 0 - - - - 1 - - 1 0 0 0
9 literale, 3 rânduri	50 literale, 12 rânduri
.f 50x10x50_n/n_NM_o10 i5 i11 i12 i30 i34 i35 i41 i45 i49 i50 o2 o7/o10 //i5 i11 i12 i30 i34 i35 i41 i45 i49 i50 o2 o7 / o10 0 1 - - - - - - - - 0 - - / 1 - - - - - - - 1 - 0 - - 0 / 1 - - - 1 - 0 - 0 - - - - / 1 - - 1 - 0 - - - - - 1 - / 1 / 0	.table i13 i19 i22 i29 i3 i38 i45 i6 i8 i9 o10 .default 0 0 1 - 0 - 0 - - - - 1 0 - - - 1 1 - - - 0 1 1 0 1 - - 0 - - - 0 1 - 1 - 1 - 1 - - - 0 1 1 - - - 0 1 - - - - 1 - 0 - - 0 0 - - - 0 1 1 - - - 1 0 - 0 - - 1 - 1 - - 0 - - - - 1 1 1 - - - - - - 0 - - 1 1 1 - - 1 - - - - 0 1 1
12 literale, 4 rânduri Total: 110 literale, 40 rânduri	38 literale, 10 rânduri Total: 337 literale, 91 rânduri

Tab. 9-36 Rezultatul minimizării cu COMIN în reţea a specificaţiei **50x10x50**

Rezultatul obţinut cu COMIN are un număr de literale redus cu 67%% faţă de rezultatul obţinut cu MVSIS şi un număr de cuburi redus cu 56% faţă de rezultatul obţinut cu MVSIS.

Exemplul 10

În acest exemplu este minimizată o funcţie care calculează maximul dintre 2 numere reprezentate pe 8 biti fiecare (un total de 16 intrări). Sistemul este complet specificat. Rezultatul este reprezentat pe 8 biţi.

```
.library max8
.variable i1,i2,i3,i4,i5,i6,i7,i8,i9,i10,i11,i12,i13,i14,i15,i16
0 1
.variable o1, o2, o3, o4, o5, o6, o7, o8 0 1

.function                                                    max8
i1i2i3i4i5i6i7i8i9i10i11i12i13i14i15i16/o1o2o3o4o5o6o7o8
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 / 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 / 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 / 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 / 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 / 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 / 0 0 0 0 0 1 0 1
.....
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 / 1 1 1 1 1 1 1 1
```

Rezultatele obţinute cu COMIN şi MVSIS pentru minimizările în paralel sunt:

COMIN	MVSIS

`.function maxim_n/n_NM_o1 i1 i9 / o1`
`0 0 / 0`
`/ 1`
	`.table i1 i9 o1`
	`.default 1`
	`0 0  0`

2 literale, 2 rânduri	2 literale, 1 rând

`.function maxim_n/n_NM_o2 i1 i2 i9 i10 / o2`
`0 - - 1 / 1`
`- - 1 1 / 1`
`- 1 0 - / 1`
`1 1 - - / 1`
`/ 0`
	`.table i1 i2 i9 i10 o2`
	`.default 1`
	`1 0 0 -  0`
	`- 0 - 0  0`
	`0 - 1 0  0`

8 literale, 4 rând	8 literale, 3 rânduri

`.function maxim_n/n_NM_o3 i1 i2 i3 i9 i10 i11 o1 o2 / o3`
`0 - - 1 - 0 - - / 0`
`- 0 - - - 0 - 1 / 0`
`- - 0 0 - - 1 - / 0`
`- - 0 - 0 - - 1 / 0`
`- - 0 - - 0 - - / 0`
`/ 1`
	`.table i1 i2 i3 i9 i10 i11 o1 o2 o3`
	`.default 1`
	`- - 0 - - 0 - -  0`
	`- - 0 0 - - 1 -  0`
	`0 - - - - 0 1 -  0`
	`- 0 - - - 0 - 1  0`
	`- - 0 - 0 - - 1  0`

14 literale, 5 rânduri	14 literale, 5 rânduri

`.function maxim_n/n_NM_o4 i1 i2 i3 i4 i9 i10 i11 i12 o2 o3 / o4`
`0 - - - 1 - - 0 - - / 0`
`- 0 - - - - - 0 1 - / 0`
`- - 0 - - - - 0 - 1 / 0`
`- - - 0 - 0 - - 1 - / 0`
`- - - 0 - - 0 - - 1 / 0`
`- - - 0 - - - 0 - - / 0`
`1 - - 0 0 - - - - - / 0`
`/ 1`
	`.table i1 i2 i3 i4 i9 i10 i11 x11 o1 o2 o3 o4`
	`.default 1`
	`- - - 0 - - - 0 - - -  0`
	`- - - 0 0 - - - 1 - -  0`
	`0 - - - - - - - 0 1 - -  0`
	`- 0 - - - - - - 0 - 1 -  0`
	`- - - 0 - 0 - - - 1 -  0`
	`- - 0 - - - - 0 - - 1  0`
	`- - - 0 - - 0 - - - 1  0`

20 literale, 7 rânduri	20 literale, 7 rânduri

`.function maxim_n/n_NM_o5 i1 i2 i3 i4 i5 i9 i10 i11 i12 i13 o1 o2 o3 o4 / o5`
`0 - - - - - - - - 0 1 - - / 0`
`- 0 - - - - - - - 0 - 1 - - / 0`
`- - 0 - - - - - - 0 - - 1 - / 0`
`- - - 0 - - - - - 0 - - - 1 / 0`
`- - - - 0 0 - - - - 1 - - - / 0`
`- - - - 0 - 0 - - - - 1 - - / 0`
`- - - - 0 - - 0 - - - - 1 - / 0`
`- - - - 0 - - - 0 - - - - 1 / 0`
`- - - - 0 - - - - 0 - - - - / 0`
`/ 1`
	`.table i1 i2 i3 i4 i5 i9 i10 i11 x11 i13 o1 o2 o3 o4 o5`
	`.default 1`
	`- - - - 0 - - - - 0 - - - -  0`
	`- - - - 0 0 - - - - 1 - - -  0`
	`0 - - - - - - - - 0 1 - - -  0`
	`- 0 - - - - - - - 0 - 1 - -  0`
	`- - - - 0 - 0 - - - - 1 - -  0`
	`- - 0 - - - - - - 0 - - 1 -  0`
	`- - - - 0 - - 0 - - - - 1 -  0`
	`- - - 0 - - - - - - 0 - - - 1  0`
	`- - - - 0 - - - 0 - - - -  1  0`

26 literale, 9 rânduri	26 literale, 9 rânduri

```
.function maxim_n/n_NM_o6 i1 i2 i3 i4 i5 i6 i9 i10
i11 i12 i13 i14 o1 o2 o3 o4 o5 / o6
0 - - - - - 1 - - - - 0 - - - - - / 0
- 0 - - - - - - - - - - 0 - 1 - - - / 0
- - 0 - - - - - - - - - 0 - - 1 - - / 0
- - - 0 - - - - - - - - 0 - - - 1 - / 0
- - - - 0 - - - - - - - 0 - - - - 1 / 0
- - - - - 0 0 - - - - - - 1 - - - - / 0
- - - - - 0 - 0 - - - - - 1 - - - / 0
- - - - - 0 - - 0 - - - - - 1 - - / 0
- - - - - 0 - - - 0 - - - - - 1 - / 0
- - - - - 0 - - - - 0 - - - - - 1 / 0
- - - - - 0 - - - - - 0 - - - - - / 0
/ 1
```

32 literale, 11 rânduri

```
.table i1 i2 i3 i4 i5 i6 i9 i10 i11 x11 i13 i14 o1
o2 o3 o4 o5 o6
.default 1
- - - - - 0 - - - - - 0 - - - - -  0
- - - - - 0 0 - - - - - 1 - - - -  0
0 - - - - - - - - - - - 0 1 - - - -  0
- 0 - - - - - - - - - - 0 - 1 - - -  0
- - 0 - - - - - - - - - 0 - - 1 - -  0
- - - - - 0 - - 0 - - - - - 1 - -  0
- - - 0 - - - - - - - - 0 - - - 1 -  0
- - - - - 0 - - - 0 - - - - - 1 -  0
- - - - 0 - - - - - - - 0 - - - - 1  0
- - - - - 0 - - - - - 0 - - - - - 1  0
```

32 literale, 11 rânduri

```
.function maxim_n/n_NM_o7 i1 i2 i3 i4 i5 i6 i7 i9 i10
i11 i12 i13 i14 i15 o1 o2 o3 o4 o5 o6 / o7
0 - - - - - - - - - - - 0 1 - - - - - / 0
- 0 - - - - - - - - - - - 0 - 1 - - - - / 0
- - 0 - - - - - - - - - - 0 - - 1 - - - / 0
- - - 0 - - - - - - - - - 0 - - - 1 - - / 0
- - - - 0 - - - - - - - - 0 - - - - 1 - / 0
- - - - - 0 - - - - - - - 0 - - - - - 1 / 0
- - - - - - 0 0 - - - - - - 1 - - - - - / 0
- - - - - - 0 - 0 - - - - - - 1 - - - - / 0
- - - - - - 0 - - 0 - - - - - - 1 - - - / 0
- - - - - - 0 - - - 0 - - - - - - 1 - - / 0
- - - - - - 0 - - - - 0 - - - - - - 1 - / 0
- - - - - - 0 - - - - - 0 - - - - - - 1 / 0
- - - - - - 0 - - - - - - 0 - - - - - - / 0
/ 1
```

38 literale, 13 rânduri

```
.table i1 i2 i3 i4 i5 i6 i7 i8 i9 i10 i11 x11 i13 i14
i15 o1 o2 o3 o4 o5 o6 o7
.default 1
- - - - - - - 0 - - - - - - - 0 - - - - - - -  0
- - - - - - - 0 0 - - - - - - - 1 - - - - - -  0
0 - - - - - - - - - - - - - - - 0 1 - - - - -  0
- 0 - - - - - - - - - - - - - - 0 - 1 - - - -  0
- - 0 - - - - - - - - - - - - - 0 - - 1 - - -  0
- - - 0 - - - - - - - - - - - - 0 - - - 1 - -  0
- - - - 0 - - - - - - - - - - - 0 - - - - 1 -  0
- - - - - 0 - - - - - - - - - - 0 - - - - - 1  0
- - - - - - 0 - - - - - - - - - - 0 - - - - -  0
- - - - - - - 0 0 - - - - - - - - 1 - - - - -  0
- - - - - - - 0 - 0 - - - - - - - 1 - - - - -  0
- - - - - - - 0 - - 0 - - - - - - - 1 - - - -  0
- - - - - - - 0 - - - 0 - - - - - - - 1 - - -  0
- - - - - - - 0 - - - - 0 - - - - - - - 1 - -  0
- - - - - - - 0 - - - - - 0 - - - - - - - 1 -  0
- - - - - - - 0 - - - - - - 0 - - - - - - - 1  0
1 - - - - - - 0 0 - - - - - - - - - - - - - -  0
/ 1
```

38 literale, 13 rânduri

```
.table i1 i2 i3 i4 i5 i6 i7 i8 i9 i10 i11 x11 i13
i14 i15 i16 o1 \
 o2 o3 o4 o5 o6 o7 o8
.default 1
- - - - - - - 0 - - - - - - - 0 - - - - - - -  0
- - - - - - - 0 0 - - - - - - - 1 - - - - - -  0
0 - - - - - - - - - - - - - - - 0 1 - - - - -  0
- 0 - - - - - - - - - - - - - - 0 - 1 - - - -  0
- - 0 - - - - - - - - - - - - - 0 - - 1 - - -  0
- - - 0 - - - - - - - - - - - - 0 - - - 1 - -  0
- - - - 0 - - - - - - - - - - - 0 - - - - 1 -  0
- - - - - 0 - - - - - - - - - - 0 - - - - - 1  0
- - - - - - 0 0 - - - - - - - - 1 - - - - - -  0
- - - - - - 0 - 0 - - - - - - - 1 - - - - - -  0
- - - - - - 0 - - 0 - - - - - - - 1 - - - - -  0
- - - - - - 0 - - - 0 - - - - - - - 1 - - - -  0
- - - - - - 0 - - - - 0 - - - - - - - 1 - - -  0
- - - - - - 0 - - - - - 0 - - - - - - - 1 - -  0
- - - - - - 0 - - - - - - 0 - - - - - - - 1 -  0
- - - - - - 0 - - - - - - - 0 - - - - - - - 1  0
1 - - - - - 0 0 - - - - - - - - - - - - - - -  0
- - - - - - 0 - - - - - - - - 0 - - - - - - -  0
- - - - - - - 0 - - - - - - - 0 - - - - - - -  0
```

45 literale, 15 rânduri
Total: 185 literale, 65 rânduri

45 literale, 15 rânduri
Total: 185 literale, 65 rânduri

Tab. 9-37 Rezultatele minimizărilor în paralel şi în reţea cu COMIN pentru **z23** pe valorile variabilelor de ieşire

Pentru acest sistem soluţiile obţinute cu cele două sisteme sunt identice.

9.3 Concluzii

În tabelul de mai jos am prezentat rezultatele minimizărilor pentru exemplele 1-10 obţinute cu aplicaţiile MVSIS (**M**) şi COMIN (**C**).

Nr.	Sistem	Specificare	Literale			Rânduri		
			M	C	%	M	C	%
1	012det	Incompletă	6	5	16%	3	3	0%
2	012ned	Incompletă	7	5	28%	3	3	0%
3 a	vectorned	Incompletă	7	4	42%	7	3	0%
3 b	**vectorned** pe valori		15	9	33%	10	7	30%
4 a	BCD7	Incompletă	39	25	35%	23	14	39%
4 b	BCD7 (COMIN paralel)		39	43	-10%	23	20	13%
5 a)	fdet	Incompletă	9	4	55%	3	3	22%
5 b)	**fdet** (COMIN expandat)		9	7	22%	3	3	-100%
6	aut_sincron	Incompletă	100	72	28%	49	34	30%
7	z23	Incompletă	39	25	35%	14	11	21%
8 a)	alu4	Completă	95	88	7,3%	42	48	-14%
8 b)	**alu4** (COMIN expandat)		72	71	1,3%	42	31	-64%
9	50x10x50	Incompletă	337	110	67%	91	40	56%
10	Max8	Completă	185	185	0%	65	65	0%

Tab. 9-38 Tabel comparativ cu rezultatele minimizărilor

Rezultatele minimizărilor în reţea ale celor 10 exemple au cumulat pentru MVSIS:

- **824** literale şi
- **300** rânduri

şi pentru COMIN:

- **523** literale şi
- **224** rânduri.

Din rezultatele de mai sus rezultă o îmbunătăţire a rezultatelor utilizând aplicaţia COMIN cu

- **36%** pentru literale şi
- **25%** pentru rânduri.

După cum se poate observa, exemplele complet specificate (8 şi 10) nu înregistrează diferenţe semnificative. În schimb, pentru funcţiile incomplet specificate s-au cumulat pentru MVSIS:

- **544** literale şi
- **193** rânduri

şi pentru COMIN:

- **250** literale
- **111** rânduri.

Rezultă o îmbunătăţire a rezultatelor minimizării cu COMIN pentru funcţiile incomplet specificate cu

- **54%** pentru literale şi
- **42%** pentru rânduri.

Aceste îmbunătăţiri se traduc direct în reducerea costurilor de implementare datorată utilizării unui număr mai mic de componente (atât în echimpament hardware cât şi în instrucţiuni corespunzătoare implementărilor software).

10 Concluzii

Obiectivul acestei lucrări a fost crearea unui sistem unitar, bazat pe principiul discriminării, pentru minimizarea funcţiilor multivalente complet sau incomplet specificate, deterministe sau nedeterministe, cu ieşiri corelate sau necorelate şi care să pună la dispoziţia utilizatorului diverse strategii ce pot fi selectate în concordanţă cu particularităţile fiecărei probleme în parte.

Demersul efectuat în acestă lucrare este structurat pe mai multe etape de cercetare:

- descrierea generală a sistemelor decizionale şi a terminologiei utilizate în specificarea acestora, în cadrul căreia am adăugat o subclasificare în funcţie de valoarea de ieşire a sistemelor: univoce (cu semantică implicit determinstă), neunivoce cu semantică determinstă şi neunivoce cu semantică nedeterministă (capitolul 2);
- analiza stadiului actual al domeniului, care include prezentarea metodelor clasice, a metodei discirminării şi o serie de algoritmi pentru procesări auxiliare necesare pentru selectarea implicanţilor soluţiei (capitolele 3 şi 4);
- crearea de modele algoritmice pentru operaţiile utilizate în minimizare (capitolul 5);
- studierea şi analiza comparativă a unui set de abordări în concordanţă cu principiul discriminării, care a condus la alegerea unei metode adecvate de procesare (capitolul 6);
- enunţarea unui set de principii şi a unui set de strategii pentru minimizare în conformitate cu metoda aleasă (capitolul 7);
- dezvoltarea metodei pentru sisteme nedeterministe şi adăugarea opţiunilor de procesare pentru minimizarea pe valori de ieşire, pentru sisteme cu ieşiri corelate, pentru determinarea valorii default în cazul sistemelor neunivoce deterministe şi nedeterministe (capitolul 8);
- implementarea aplicaţiei şi analiza rezultatelor pentru diverse sisteme decizionale (cazuri de utilizare), cu scopul de a evalua performanţele metodei (capitolul 9);
- prezentarea concluziilor asupra muncii depuse, a contribuţiilor personale şi determinarea direcţiilor viitoare de cercetare (capitolul 10).

Structurile de logică vectorială, cu particularizarea „logică algoritmică”, optimizabile prin discriminare, deschid perspectiva obţinerii unui salt în viteza de prelucrare a informaţiei prin „hardificarea” software-lui sau direct prin optimizarea structurilor decizionale ale algoritmilor.

Metodele de minimizare pot fi clasificate în funcţie de diverse criterii:
- tipul metodei;
- domeniul variabilelor;
- forma de reprezentare a datelor;
- tipul de ieşire al funcţiilor logice minimizate;
- tipul soluţiei obţinute.

Metoda de minimizare	Metodă		Domeniu variabile		Reprezentare		Ieşire		Soluţie	
	Grafică	Analitică	Binar	Multivalent	Tabelară	Diagramă de decizie	Singulară	Multiplă	Exactă	Euristică
Karnaugh	x	x	x		x		x			
Quine		x	x		x		x			
MINI		x	x		x			x		
ESPRESSO		x	x			x		x	x	x
MVSIS		x		x		x		x	x	x
FCMIN		x	x		x			x		x
CD-COVERAGE		x	x		x		x			x
BOOM I		x	x		x			x		x
BOOM II		x	x		x			x		x
Discriminării		x		x	x			x	x	

Tab. 10-1 Clasificarea metodelor şi algoritmilor de minimizare

Toate metodele de minimizare cunoscute, clasice sau moderne, metode precise sau euristice, enumerând aici: metoda analitică Quine-McCluskey, metoda grafică sau grafo-analitică a diagramei Karnaugh, metodele analitice euristice sau precise Espresso I şi II, Espresso MV, metoda SIS, sau MVSIS, indiferent dacă se referă la sisteme binare sau multivalente, cu ieşiri singulare sau multiple, au următoarele caracteristici comune:
- sunt în esenţă binare;
- se aplică sistemelor cu o singură ieşire binară (ieşirile multiple binare, sau singulare multivalente se reduc sau se descompun în subsisteme binare cu o singură ieşire);
- se aplică doar ieşirilor multivalente singulare, după descompunerea sistemelor cu ieşiri multiple în subsisteme cu ieşiri singulare, care, la rândul lor sunt codificate binar, tip 1 din N (adică fiecare din cele N valori multivalente se reprezintă printr-un vector binar cu N poziţii, având doar o valoare 1, restul componentelor vectorului fiind 0), iar ieşirea multiplă rezultată se tratează ca N ieşiri singulare separate;

Principiul de bază al acestor metode de minimizare constă în analizarea valorilor intrării pentru care funcţia are valoarea 1 (sau, alternativ, 0), folosindu-se teoremele de bază ale algebrei booleene (dintre care menţionez adiacenţa, factorul comun, consensul, teoremele lui De Morgan etc.) şi ceea ce este specific acestor metode, optimizarea folosind valorile intrării pentru care funcţia nu este specificată. Un asemenea principiu implică detalierea şi analizarea tuturor combinaţiilor de intrare nespecificate, operaţie care poate fi destul de laborioasă, în cazul unui mare număr de intrări, sau în cazul sistemelor multivalente.

Întrucât, de fiecare dată, analiza în vederea minimizării se realizează pe două submulţimi ale valorilor intrării:

- submulţimea *ON* , pentru care funcţia are valoarea 1 şi
- submulţimea *DC* (don't care), pentru care funcţia nu este specificată,

sau

- submulţimea *OFF* , pentru care funcţia are valoarea 0 şi
- submulţimea *DC* (don't care), pentru care funcţia nu este specificată,

rezultă o limitare a metodelor doar la optimizarea funcţiilor cu ieşiri binare singulare, urmărindu-se să se reducă celelalte cazuri la acesta, prin descompunerea lor în subsisteme cu o singură ieşire binară.

Metoda discriminării, spre deosebire de celelalte metode, foloseşte pentru minimizare submulţimile valorilor intrării, grupate pe valorile specificate ale funcţiei, neanalizându-se submulţimea valorilor intrării pentru care funcţia nu este specificată. Conform acestei metode minimizarea constă în determinarea şi reţinerea din vectorii de intrare aparţinând unei grupe valorice a ieşirii, a numărului minim de componente specificate care-l separă (îl discriminează) faţă de toţi ceilalţi vectori de intrare aparţinând altor grupe valorice ale funcţiei.

Principalele avantaje ale metodei discriminării prezentate în capitolul 4 pot fi enumerate succint astfel:

- abordare directă, naturală multivalentă, nefiind necesare nici un fel de conversii intermediare în binar;
- permite minimizarea tabelelor de valori având valori de intrare şi ieşire variabile binare sau multivalente;
- variabilele pot fi simple sau vectori;
- tratarea directă a ieşirilor multiple, simple sau vectoriale;
- tratare unitară a optimizării sistemelor decizionale nedeterministe, extinzând, în raport cu cercetările cunoscute în domeniu, gama de soluţii asupra optimizării sistemelor decizionale hardware şi software parţial optimizate, manifestând nedeterminări implicite, oferind totodată şi o soluţionare mai directă, faţă de abordările recente cunoscute, privind tratarea neunivocităţilor explicite, tratate – în cazul aplicării metodei discriminării – ca valori multivalente suplimentare ale ieşirii.

Metoda discriminării permite o gamă largă de aplicaţi care probează flexibilitatea, adaptabilitatea şi eficienţa metodei. Structurile de logică vectorială, cu particularizarea ei „logică algoritmică”, optimizabile prin discriminare, deschid perspectiva obţinerii unui salt în viteza de prelucrare a informaţiei, prin „hardificarea” software-lui, sau direct prin optimizarea structurilor decizionale ale algoritmilor.

Referitor la optimizarea algoritmilor, trebuie menţionat că numai într-o algebră multivalent-vectorială se pot exprima natural algoritmii, iar metoda discriminării oferă instrumentul practic al optimizării într-o astfel de algebră. Acestea evidenţiază disponibilitatea metodei discriminării de a soluţiona practic, mai direct, orice problemă de optimizare care implică minimizare, şi uneori oferind singura cale practică; se datorează – aşa cum s-a arătat - principiului de bază de minimizare al discriminării, principiu multivalent natural, diferit de principiul de bază, binar în esenţă, al tuturor metodelor cunoscute.

Metoda discriminării, aşa cum este formulată de autor, permite o serie de extensii care completează tipurile de sisteme logice ce pot fi minimizate. Prima problemă ridicată este cea a datelor de intrare. Un tabel de valori poate fi incomplet, ambiguu sau semioptimizat. Pentru un sistem logic univoc ambiguitatea tabelului de intrare reprezintă o deficienţă a proiectării care poate fi înlăturată în mod automat prin eliminarea ambiguităţilor sau prin reconsiderearea sistemului ca fiind univoc. Soluţia obţinută în urma minimizării poate prezenta hazard. Pentru o implementare software acesta nu constituie o problemă, dar pentru o implementare hardware trebuie luat în considerare.

Utilizarea extensiilor prezentate (pentru tratarea ambiguităţilor de proiectare a sistemelor logice univoce şi pentru tratarea hazardului) lărgeşte domeniul de aplicare al metodei discriminării atât din punct de vedere al datelor de intrare, cât şi din punct de vedere al utilizării. Acestea pot fi implementate analitic, lăsând în seama utilizatorului luarea deciziilor acolo unde acest lucru se impune. Utilizatorul poate alege eliminarea ambiguităţilor prin ştergerea rândurilor implicate sau prin reconsiderearea sistemului decizional ca fiind neunivoc. În ceea ce priveşte hazardul, dacă implementarea sistemului se face hardware, atunci hazardul poate fi anunţat de către metodă. Mai mult, metoda poate elimina o clasă de hazard (cel logic) şi anunţă sursa hazardului funcţional (pachetele de rânduri din tabelul minimizat care induc un astfel de hazard). În aceste condiţii utilizatorul trebuie doar să găsească soluţii tehnologice pentru eliminarea hazardului funcţional.

Modelul matematic prezentat acoperă problemele legate de logica multivalentă implicată în minimizare:

- specificarea multivalentă şi reprezentarea tabelară a datelor de intrare;
- domenii multivalente, variabile multivalente şi operatori specifici;
- vectori de variabile multivalente şi operatori specifici;
- matrice (liste de vectori) şi operatori specifici;

- operatori neomogeni (operatori cu operanzi vectori şi matrice).

Pe baza modelului matematic am conceput trei metode preliminare pentru pentru a testa diverse strategii de minimizare şi a determina o metodă viabilă de minimizare. Prima metodă prezentată a scos în evidenţă un sistem de analiză ce permite minimizarea informaţiei cunoscute (fără a utiliza informaţie suplimentară – componenta nespecificată a funcţiei). Comportamentul algoritmului corespunde unui motor de analiză a datelor în vederea obţinerii tuturor deducţiilor logice (motor de inferenţă). A doua metodă corespunde unui sistem de analiză al cărui motor funcţionează pe baza eliminării cazurilor nepermise („negative thinking"). A treia metodă este foarte eficientă din punct de vedere al calităţii rezultatelor, poate fi utilizată pe orice sistem de calcul, deoarece memoria utilizată este constantă, permite implementarea paralelă deoarece împarte spaţiul de căutare în zone disjuncte care pot fi procesate separat – este nevoie de foarte puţină informaţie de sincronizare, doar în momente bine determinate, când se analizează soluţii parţiale. Implementările algoritmilor prevăd posibilitatea conectării în reţea a nodurilor de ieşire după o strategie proprie, care, în condiţiile în care sunt folosite ca noduri de reţea chiar valorile ieşirilor multivalente, se obţine o minimizare mult mai puternică.

Din analiza metodelor de minimizare testate se desprinde un set de principii care pot sta la baza operaţiilor de minimizare şi un set de strategii ce se pot aplica în funcţie de semantica atribuită fiecărui tabel de valori:

- principiile minimizării
  - principiul discriminării
  - principiul discriminării în raport cu principiile minimizării „clasice"
  - principiul conservării integrităţii funcţionale a specificaţiei iniţiale
  - principiul minimizării paralele
  - principiul minimizării în reţea
  - principiul minimizării pe valorile variabilelor de ieşire
  - principiul minimizării cu ieşiri corelate
- opţiuni şi strategii de minimizare
  - strategii de minimizare în reţea
  - minimizarea neunivocităţilor explicite
  - minimizarea neunivocităţilor implicite
  - neunivocitate vs. nedeterminism
  - minimizarea tabelelor neunivoce cu semantică deterministă (sau)
  - minimizarea tabelelor neunivoce cu semantică nedeterministă
  - utilizarea opţiunii „în rest" cu valoarea „default"

Tehnica finală prezentată este o metodă proprie, originală de minimizare, în conformitate cu principiile enunţate. Principiul discriminării [14], [177], considerat ca punct de plecare al metodei, a deschis calea unor noi abordări, mai eficiente, în raport cu metodele cunoscute (clasice), ale

minimizării sistemelor decizionale, în general. Metoda minimizează ieşirile multiple ale sistemului decizional multivalent, determinist sau nedeterminist, complet sau incomplet specificat, cu ieşiri corelate sau necorelate.

Metodele clasice ale minimizării multivalente folosesc reducerea diagramei de decizii la calculul arborelui minim decizional pentru specificaţia dată. Dacă specificaţia dată este incomplet specificată, atunci rezultatul acestei abordări include tot spaţiul combinaţional al valorilor variabilelor de intrare. Metoda prezentată poate accesa doar în mod implicit partea nespecificată a spaţiului valorilor de intrare. Utilizarea acestei zone nu reprezintă un scop în sine pentru minimizare, ci o consecinţă a aplicării principiilor enunţate. Metoda determină elementele care discriminează combinaţiile de intrare cu ieşiri diferite [220]. Specificaţiile binare reprezintă un caz particular al clasei de probleme acoperite.

Algoritmul facilitează minimizarea paralelă şi în reţea. Minimizarea în reţea stabileşte implicit o ordine pentru calculul funcţiei, care poate conduce la rezultate remarcabile chiar şi pentru probleme clasice, binecunoscute (ex. BCD 7 segmente) [219]. Minimizarea în reţea se poate aplica şi unui tabel cu mai multe ieşiri multivalente şi chiar cu o singură ieşire binară sau multivalentă dacă se foloseşte expandarea pe valori a variabilelor de ieşire.

Soluţia este evaluată după numărul de literale. În principiu, există mai multe soluţii minimizate echivalente pentru acelaşi tabel iniţial. Metoda calculează şi dă ca rezultat numai una dintre soluţii, deoarece generarea tuturor variantelor de soluţii este nerealistă. Sunt cazuri în care numărul soluţiilor echivalente este foarte mare.

Metoda permite două tipuri de minimizare: una care acoperă toate specificaţiile de proiectare (fără pierderea vreunei informaţii din tabelul dat), urmând să acopere toate valorile de ieşire, şi alta care acoperă cel puţin o valoare de ieşire pentru fiecare combinaţie de intrare nedeterministă. Primul model corespunde tratării tabelelor deterministe, precum şi tabelelor nedeterministe interpretate ca tabele neunivoce cu semantică deterministă - opţiunea n/n. Al doilea model corespunde nedeterminismului pur, în care se înţelege că din specificarea iniţială se poate alege orice submulţime nevidă - opţiunea k/n - din mulţimea valorilor unei variabile de ieşire corespunzătoare părţii de intrare a unui rând din tabelul iniţial. Metoda permite o alegere optimă a acestei submulţimi, din punctul de vedere al minimizării. În acest sens, oferirea ca rezultat a unei submulţimi cu mai multe elemente (acolo unde rezultă) constituie un avantaj pentru implementare, deoarece foloseşte maximum de potenţial funcţional al tabelului iniţial, în condiţiile unei implementări minime.

Un tabel univoc este determinist şi un tabel nu poate fi nedeterminist decât dacă este neunivoc. Tabele neunivoce reprezintă doar o subclasă a tabelelor deterministe. Fixarea unei singure valori de ieşire pentru tabele nedeterministe reprezintă o constrângere care reduce clasa de soluţii pentru minimizarea tabelelor nedeterministe. Interpretarea cea mai naturală o reprezintă selectarea un subset optim din punct de vedere

al minimizării pentru valorile de ieşire corespunzătoare rândurilor neunivoce.

Complexitatea procesului de minimizare este determinată pe de o parte de cantitatea de date de intrare şi, pe de altă parte, de complexitatea logicii cuprinse în tabelul iniţial. Se pot da uşor exemple de perechi de tabele identice din punct de vedere al metricii, dar care implică un efort de procesare diferit.

Rezultatul se situează pe o treaptă superioară faţă de diagramele de decizie deoarece clasa de implementări posibile pentru tabelul de valori este mai largă decât cea rezultată utilizând diagrame de decizie.

Minimizarea în reţea, pe variabile de ieşire sau pe valorile variabilelor de ieşire, aşa cum este tratată în prezenta lucrare produce rezultate remarcabile, putând fi aplicată atât direct, pe variabile de ieşire, cât şi pe valorile variabilelor de ieşire. Întrucât în momentul întoarcerii unei variabile de ieşire pe intrare pentru minimizarea variabilelor de ieşire rămase este utilizat implicit tabelul de valori al variabilei întoarse – implicit şi „logica" integrată în el -, metoda oferă posibilitatea minimizării cu păstrarea corelaţiilor dintre variabilele de ieşire. Minimizarea în reţea are drept corespondent în implementările software utilizarea variabilelor temporare. O variabilă temporară înglobează un anumit volum de calcul care poate fi reprodus oricând doar prin simpla utilizare a acesteia. La fel, şi în cazul tabelelor de valori, întoarcerea unei variabile de ieşire pe intrare aduce, fără a modifica în vreun fel specificaţia iniţială, posibilităţi noi de selecţie.

În cadrul acestei lucrări am dezvoltat o strategie originală, pe baza unor criterii care permit identificarea submulţimilor valorilor de ieşire ce pot fi atribuite valorii „**default**". Aceste criterii permit selecţia unei valori „**default**" chiar şi în cazul specificărilor nedeterministe.

Contribuţii

Din demersul acestei teze subliniez următoarele contribuţii:

- investigarea stadiului actual al domeniului în care se încadrează obiectivul studiului realizat;
- elaborarea unui mecanism de minimizare alături de autorul metodei discriminării care a condus la finisarea şi validarea metodei;
- analiza comparativă dintre metoda discriminării şi metodele existente, finalizată cu câteva strategii propuse pentru extinderea acesteia.
- crearea unui model matematic pentru analiza sistemelor multivalente;
- crearea a trei modele preliminare şi a aplicaţiilor corespunzătoare pentru minimizare. Cele trei modele au contribuit la studiul algoritmilor posibili pentru minimizare şi la validarea modelului matematic, pe baza cărora au fost create;

- stabilirea unor principii şi strategii de minimizare. Acestea sunt generale, în sensul că acoperă toate strategiile investigate în cadrul studiului stadiului actual, metoda discriminării şi strategiile propuse pentru extinderea metodei discriminării;
- crearea unei metode de minimizare bazată pe setul de principii stabilit care implementează toate strategiile enumerate.

Lucrarea conţine o dezvoltare originală ca metodă şi teorie a tehnicilor de minimizare cunoscute, conducând la rezultate mai bune în comparaţie cu metodele existente. În acest sens, articolul „An efficient network strategy in deep minimizations of Multivalued decision systems" conţine prezentate studii comparative de aplicaţii concrete, clasice, din domeniu, atât binare cât şi multivalente. Din studiile comparative rezultă că tehnica prezentată conduce la o soluţie de minimizare, eficientă, mai bună, pur multivalentă. De exemplu, la aplicaţia convertor 50x10x50 este mai mică 67% ca număr de iterale şi cu 55% ca număr de rânduri faţă de soluţiile cunoscute. În plus, aplicarea tehnicii de minimizare dezvoltate şi implementate cu un concept original de conectare în reţea a unei funcţii multivalente conduce la minimizări suplimentare faţă de tot ce este obţinut până acum cu alte metode.

Rezultatele comparative prezentate în capitolul 9 demonstrează eficienţa metodei, care, în cazul multivalent, poate fi considerată prima tehnică ce permite minimizarea ghidată de semnatica sistemului: univocă, neunivocă deterministă şi nedeterministă. Metodele preliminare, aşa cum au fost concepute, conduc la minimizări eficiente ale tabelelor de valori, cu neunivocităţi explicite în condiţiile în care vectorii de intrare sunt disjuncţi.

Perspective

În prezent, soluţiile hardware rapide constau în paralelizarea calculului prin creşterea numărului de intrări binare. Implementările software fiind în esenţă secvenţiale, sunt la polul opus. Mai mult, utilizarea variabilelor cu domeniu de valori mare implică o serializare în timp a valorii transmise. În final s-a ridicat următoarea problemă: având la dispoziţie o metodă de sinteză multivalentă, există un optim din punct de vedere al implementării între numărul de intrări (codarea în lăţime) şi codarea în timp?

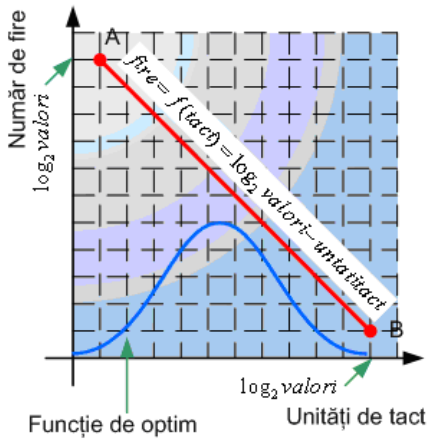


Fig. 10-1. Modalităţi de reprezentare a valorilor multivalente

De exemplu, o variabilă cu 256 de valori poate fi reprezentată pe 8 fire, sau în 8 unităţi de timp, sau poate fi reprezentată pe h fire şi pe $w = \left\lceil log_2 \left(\frac{256}{2^h} \right) \right\rceil = \left\lceil log_2 256 - log_2 2^h \right\rceil = 8 - h$ unităţi de tact. Este evident că din punct de vedere al vitezei optimul este în punctul A (numărul maxim de fire). Cu implementările realizate deja, în punctul B se poate obţine o complexitate mai redusă şi un consum de energie mai mic.

Metoda combinaţională prezentată poate fi aplicată direct în nodurile unei reţele secvenţiale pentru minimizarea acestora sau pot fi adăugate ca în exemplul cu automatul sincron semnale distincte pentru starea curentă şi starea următoare. Una dintre directiile viitoare o constituie studiul minimizării sistemelor sincrone [2][223][224][225][226][227][228][229][230][231][232] cu metoda propusă.

11 Bibliografie

[1] Jiang, J.-H.R.; Mischenko, A.; Brayton, R.K., "Reducing Multivalued Algebraic Operations To Binary", Design, Automation and Test in Europe Conference and Exhibition, 2003Volume , Issue , pp: 752 - 757, 2003

[2] Fan, Mo; Brayton, R.K. , "A Timing-Driven Module-Based Chip Design Flow", Proceedings of the 41st Design Automation Conference, pp: 67- 70, ISSN: 0738-100X, ISBN: 1-51183-828-8, 2004

[3] Michael Hutton, Khosrow Adibsamii, Andrew Leaver, "Adaptive Delay Estimation For Partitioning-Driven Pld Placement", IEEE Transactions onVolume 11, Issue 1, pp: 60–63,, 37653

[4] M. Karnaugh, "The Map Method for synthesis of combinatorial logic circuits", Transactions of the American Institute of Electrical Engineers Communications and Electronics, vol. 72, pp. 593-599, November, 1953

[5] Quine, Willard, "A way to simplify truth functions", American Mathematical Monthly, vol. 62, no. 9, pp. 627-631, November., 1955

[6] McCluskey Jr., Edward J., "Minimization of Boolean functions", Bell Szstems Technical Journal. Vol. 35, no. 6, pp. 1417-1444, 1956

[7] Brayton, R.K., G.D. Hachtel, C. McMullen, A.L Sangiovanni-Vincentelli, " Logic Minimization Algorithms for VLSI Synthesis ", Boston: Kluwer Academic Publisher, 1984

[8] Hachtel, Gary D., Fabio Somenzi., "Logic Synthesis and Verification Algorithms.", Boston: Kluwer Academic Publisher, 1984

[9] Ellen M. Sentovich, Kanwar Jit Singh, Luciano Lavagno, Cho Moon, Rajeev Murgai, Alexander Saldanha, Hamid Savoj, Paul R. Stephan, Robert K. Brayton, Alberto Sangiovanni-Vincentelli., "SIS: A System for Sequential Circuit Synthesis.", Univ. of California, Berkeley, CA 94720: Tech. Rep. UCB/ERL M92/41 Electronics Research Lab, May 1992

[10] Gao, M., J.-H. Jiang, Y. Jiang, Y. Li, S. Sinha, R. Bryton. , "Negative Thinking In Branch-And-Baund: The Case Of Unate Covering", Tahoe City: In the Note of the International Workshop on Logic Synthesis, June 2001.

[11] Chai, Donald, Jie-Hong Jiang, Yunjian Jiang, Yinghua Li, Alan Mischenko, Robert Brayton. , "MVSIS 2.0 User's Manual.", University of Berkeley CA 94720: Departament of Electrical Engineering and Computer Sciences, 2003.

[12] P., Fišer, J. Hlavička., "BOOM - A Heuristic Boolean Minimizer", Computers and Informatics, Vol. 22, No. 1 . 2003, pp. 19-51, Jan 2003

[13] Fiser, Petr, Hana Kubatova, "Flexible Two-Level Boolean Minimizer BOOM-II and Its Applications", Proceedings of the 9th EUROMICRO Conference on Digital System Design, 369 - 376., 2006

[14] Ion Ştefănescu, "DISCRIMINATION: A New Principle in the Efficient Minimization of the Binary and Multiple-Valued Functions", 1st International Conference on Electronics, Computers and Artificial Intelligence – ECAI 2005 – University of Piteşti, Romania, 1-2 July 2005.

[15] **Zafiu, Adrian**, Ion Ştefănescu., "Minimization aspects of the hardware and Software Systems – An Extension of the Discrimination Method.", 1st International Conference on Electronics, Computers and Artificial Intelligence – ECAI 2005 – University of Piteşti, , 2005

[16] Gheorghe M. Ştefan, "Funcţie şi structură în sistemele digitale", Editura Academiei Române, Bucureşti, România, 1991.

[17] Arthur D. Friedman, "Fundamentals of Logic Design and Switching Theory", Computer Science Press, Inc., U.S.A. , 1986.

[18] F. Martin McNeill, Ellen Thro, "Fuzzy Logic A Practical Approach", Library of Congress Cataloging-in-Publication Data, Academic Press Limited, London ISBN 0-12-485965-8 , 1994
[19] Huntington, E. V, "Sets of Independent Postulates for the Algebra of Logic", Trans. Am. Math. Soc., 5, 288-309, 1904.
[20] P. Fišer, J. Hlavička , "Efficient Minimization Method For Incompletely Defined Boolean Functions", Proc. 4th Int. Workshop on Boolean Problems, Freiberg (Germany), pp. 91-98 , Sept. 21-22, 2000
[21] Massimo Baleani, Frank Gennari, Yunjian Jiang, Yatish Patel, Robert K. Brayton, Alberto Sangiovanni-Vincentelli, "HW/SW Partitioning and Code Generation of Embedded Control Applications on a Reconfigurable Architecture Platform", Proceedings of the 10th International Symposium on Hardware/Software Codesign (CODES), Estes Park, Colorado, USA, pp.151 - 156 , May 2002
[22] Yunjian Jiang, Robert K. Brayton , "Software Synthesis From Synchronous Specifications Using Logic Simulation Techniques", Design Automation Conference, 2002. Proceedings. 39th Volume , Issue, pp. 319 - 324, 2002
[23] Elena Dubrova, Luca Macchiarulo , "A Comment On Graph-Based Algorithm For Boolean Function Manipulation", IEEE Transactions on Computers, Vol. 48, No. 11, pp. 1290-1292, November 2000
[24] R.E. Bryton, "Graph-Based Algorithms For Boolean Function Manipulation", IEEE Transactions on Computers, pp C-35(8): 667-691 , 31625
[25] Cristian-Győző Haba, "Contribuţii la Sinteza Structurilor Numerice De Comandã", Teză doctorat, Universitatea Tehnică "Gh.Asachi" Iaşi, Facultatea De Automatică Şi Calculatoare, 1999
[26] S. Mitra, N. Saxena, E.J. McCluskey, "Techniques For Estimation Of Design Diversity For Combinational Logic Circuits", IEEE Intl. Conf. Dependable Systems and Networks, pp. 25-34, 2001
[27] Valentina Ciriani, "A New Approach To Three-Level Logic Synthesis", Technical Report TR-02-03, Dipartimento di Informatica, Universita di Pisa, 37347
[28] Satrajit Chatterjee, Robert Brayton, "A New Incremental Placement Algorithm And Its Application To Congestion-Aware Divisor Extraction", IEEE/ACM International Conference on Volume , pp 541 – 548, 7-11 Nov. 2004
[29] R.M. Fuhrer, S.M. Nowick, M. Theobald ,N.K. Jha, B. Lin, L. Plana, "Minimalist An Environment For The Synthesis Veri Cation And Testability Of Burst Mode Asynchronous Machines", Technical Report TR CUCS-020-99, Columbia Univ., New York , July 1999
[30] Marsha Chechik, Arie Gurfinkel, Benet Devereux, Albert Lai, Steve Easterbrook, "Data Structures For Symbolic Multi-Valued Model-Checking", Formal Methods in System Design, Springer Netherlands, ISSN 0925-9856, pp. 295-344, 2006, 2004
[31] R. Jiang, R. Brayton , "Depth-bounded Communication Complexity For Distributed Computation", Proc. IWLS '03, pp. 51-58, 2003
[32] Amelia Huimin Sheen, "Probabilistic Representation And Manipulation Of Boolean Functions Using Free Boolean Diagrams", Thesis E.E. Ph.D , 1994
[33] Alan Mishchenko, Robert K. Brayton, "Sat-Based Complete Don't-Care Computation For Network Optimization", Proceedings of Design, Automation and Test in Europe, 2005, Vol. 1, pp. 412 – 417, 7-11 March 2005
[34] M. Gao, R. K. Brayton, "Semi-Algebraic Methods For Multi-Valued Logic", IEEE IWLS 2000, International Workshop on Logic Synthesis, Dana Point, CA, Workshop Notes 73-80 , May 31 - June 2, 2000
[35] Robert Mateescu, Rina Dechter , "A Comparison Of Time-Space Schemes", Proceedings of The Twentieth International Joint Conference on Artificial Intelligence, IJCAI-07, Hyderabad, India., 2006
[36] Alexander Saldanha, Tiziano Villa, Robert K. Brayton, Alberto L. Sangiovanni-Vincentelli, "A Framefork For Satisfying Input And Output Encoding Consttraints", 28th ACM/IEEE Design Automation Conference, 1991
[37] Rina Dechter, Robert Mateescu , "A Simple Insight Into Iterative Belief Propagation's Success", UAI, 2003

[38] Andres Martinelli, "Advances In Functional Decomposition: Theory And Applications", Department of Electronic, Computer, and Software Systems, School of Information and Communication Technology, Royal Institute of Technology (KTH), Stockholm, Sweden, 2006
[39] Robert Mateescu, Rina Dechter, "And/Or Cutset Conditioning", In proceedings of The Nineteenth International Joint Conference on Artificial Intelligence, IJCAI-05, Edinburgh, Scotland, 2005
[40] Robert Mateescu, Rina Dechter , "And/Or Search Spaces And The Semanticwidth Of Constraint Networks", ,
[41] Rina Dechter, Robert Mateescu, "And/Or Search Spaces For Graphical Models", Artificial Intelligence Journal, Vol. 171, No. 2-3, pp: 73-106, 2007
[42] Robert Mateescu, Rina Dechter, "Compiling Constraint Networks Into And/Or Multi-Valued Decision Diagrams (Aomdds)", Proceedings of Principles and Practice of Constraint Programming - CP 2006, 12th International Conference, CP 2006, Nantes, France, September 25-29, 2006
[43] Rina Dechter, Kalev Kask, Robert Mateescu, "Iterative Join-Graph Propagation", In Proc. Of 18th International Conference on Uncertainty in artificial Intelligence (UAI ,92), Edmonton, Canada (2002) 128-136, 2002
[44] Rina Dechter, Robert Mateescu , "Mixtures Of Deterministic-Probabilistic Networks And Their And/Or Search Space", In proceedings of The Twentieth Conference on Uncertainty in Artificial Intelligence, UAI-04, Banff, Canada, 2004
[45] Rina Dechter, Robert Mateescu, "The Impact Of AND/OR Search Spaces On Constraint Satisfaction And Counting", In proceedings of The Tenth International Conference on Principles and Practice of Constraint Programming, CP-04, Toronto, Canada. pp. 731--736, 2004
[46] Robert Mateescu, Rina Dechter, "The Relationship Between And/Or Search Spaces And Variable Elimination", ,
[47] Robert Mateescu, Rina Dechter, Kalev Kask, "Tree Approximation For Belief Updating", Eighteenth national conference on Artificial intelligence, Edmonton, Alberta, Canada, pp. 553 – 559, ISBN:0-262-51129-0, 2002
[48] Robert Mateescu, Rina Dechter, „A Comparison of Time-Space Schemes for Graphical Models”,Proceedings of The Twentieth International Joint Conference on Artificial Intelligence, IJCAI-07, Hyderabad, India., 2006
[49] J. P. Roth, ""A Calculus and An Algorithm for the Multiple-Output 2-Level Minimization Problem'", Research Report RC 2007, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, February, 1968
[50] S.J. Hong, R.G. Cain, D.L. Ostapko, "MINI: A Heuristic Approach for Logic Minimization", IBM Journal of Research and Development, Volume 18, No. 5, pp: 443-458, September, 1974
[51] C.Y. Lee, "Representation of Switching Circuits by Binary-Decision Programs", Bell System Technical Journal, Vol. 38, pp. 985-999, July, 1959
[52] S.B. Akers, "Binary Decision Diagrams", IEEE Transactions on Computers, Vol. C-27, No. 6, pp. 509-516, June , 1978
[53] John Harrison, , "Binary Decision Diagrams As A Hol Derived Rule", The Computer Journal, vol. 38, pp. 162-170, 1995
[54] Drechsler, R.; Thornton, M.; Wessels, D. Multiple-Valued Logic, "Mdd-Based Synthesis Of Multi-Valued Logic Networks", Proceedings. 30th IEEE International Symposium on Volume pp:41–46, 2000
[55] Alan Mishchenko, Robert Brayton, "Simplification Of Non-Deterministic Multi-Valued Networks", IEEE/ACM International Conference on Computer Aided Design, 2002, ICCAD 2002, pp. 557- 562, ISSN: 1092-3152 ISBN: 0-7803-7607-2 , 10-14 Nov. 2002
[56] C. Shanon, "A symbolic analysis of relay and switching circuits", PhD thesis, 1938
[57] Yunjian Jiang, Slobodan Matic, Robert K. Brayton, "Generalized Cofactoring For Logic Function Evaluation", Proceedings Design Automation Conference, 2003, pp: 155- 158, ISBN: 1-58113-688-9, 2-6 June 2003
[58] Vivek Chakravarthy Komaragiri, "Application Of Decision Diagrams For Information Storage And Retrieval", Thesis (M.S.)-Mississippi State University, 2002

[59] R.Jacobi,A.M.Trullemans, "Boolean Mapping with Binary Decision Diagrams ", IFIP Workshop on Logic and Architecture Synthesis, pp.19-38 INPG Grenoble, France, December, 1993
[60] N. Yevtushenko, T. Villa, R. K. Brayton, A. Mishchenko, A. L. Sangiovanni-Vincentelli, , "Composition Operators In Language Equations", Proceedings of IWLS '04, pp. 409-415, 2004
[61] Jin S. Zhang, Alan Mishchenko, Robert Brayton, Malgorzata Chrzanowska-Jeske, "Symmetry Detection For Large Boolean Functions Using Circuit Representation, Simulation And Satisfiability", Proc. DAC '06, pp. 510-515, 2006
[62] J.Goubault-Larrecq , "Implementing Tableaux by Decision Diagrams", August, 1996
[63] Necula, G. C., "Compiling With Proofs", Doctoral Thesis. UMI Order Number: AAI9918593, Carnegie Mellon University, 1998
[64] Nina Yevtushenko, Tiziano Villa, Robert K. Brayton, Alex Petrenko, Alberto L. Sangiovanni-Vincentelli , "Compositionally Progressive Solutions Of Synchronous Language Equations", Proceedings of the IEEE/ACM 12th International Workshop on Logic & Synthesis (IWLS'03). Laguna Beach, California, USA , May 28-30, 2003
[65] Alan Mishchenko, Robert Brayton, Roland Jiang, Tiziano Villa, Nina Yevtushenko , "Efficient Solution Of Language Equations Using Partitioned Representations", Proceedings of the Design, Automation and Test in Europe, 2005, Vol. 1, pp. 418 - 423 , 7-11 March 2005
[66] S. Sanner, D. McAllester, "Affine Algebraic Decision Diagrams (Aadds) And Their Application To Structured Probabilistic Inference", Proceedings of the 19th International Joint Conference on AI (IJCAI-05) , 2005
[67] Trevor Meyerowitz, Subarna Sinha, Robert Brayton, "Applying Edge Partitioning To Spfd's", EE219B Final Project Report, May 2000, 2000
[68] Javier Borge Holthoefer, "Algebra De Boole: Del Silogismo Aristotelico A Los Circuitos Integrados", Artículo de en la revista A parte rei, 2003
[69] Yunjian Jiang, Robert Brayton, "An Information Theoretic Approach To Logic Evaluation", International Workshop on Logic Synthesis (IWLS), Anaheim, June 2003
[70] Michael D. Hutton, Jonathan Rose, "Applications Of Clone Circuits To Issues In Physical-Design", Proceedings of the 1999 IEEE International Symposium on Circuits and Systems, ISCAS '99, vol.6, pp: 448-451, Orlando, FL, USA ISBN: 0-7803-5471-0, Jul 1999
[71] Jochen Bern, Christoph Meinel, Anna Slobodova , "Eficient OBDD Based Boolean Manipulation In Cad Beyond Current Limits", Trierer Forschungsberichte, Fachbereich IV - Mathematik / Informatik, Universitat Trier, ISSN 0944-0488 , 2006
[72] Chien-Chung Tsai, Malgorzata Marek-Sadowska, "Multilevel Logic Synthesis For Arithmetic Functions", Proc. IEEE/ACM Design Automation Conference, pp.242–247, June 1996
[73] H. Touati , "Implicit state enumeration of finite state machines using BDD's", Proceedings of the International Conference on Computer-Aided Design 1990, pp. 130-133, November1990
[74] Yunjian Jiang, Robert K. Brayton, "Don't Cares In Logic Minimization Of Extended Finite State Machines", Design Automation Conference, 2003. Proceedings of the ASP-DAC 2003. Asia and South Pacific, pp. 809- 815, ISBN: 0-7803-7659-5 , 21-24 Jan. 2003
[75] Michael L. Case, Alan Mishchenko, Robert K. Brayton, "Inductively Finding A Reachable State Space Over-Approximation", Proc. IWLS '06, pp. 172-179, 2006
[76] Donald E. Knuth, "Arta programării calculatoarelor, Volumul I - Algoritmi fundamentali", Editura Teora, 2000
[77] Gitanjali M. Swamy, "An Exact Logic Minimizer Using Implicit Binary Decision Diagram Based Methods", Masters Thesis - Department of Electrical Engineering and Computer Science - University of California, Berkeley, December 1993
[78] A.L. Oliveira, L.P. Carloni, T. Villa, A.L. Sangiovanni-Vincentelli, "An Implicit Formulation For Exact Bdd Minimization", VLSI: Integrated Systems on Silicon, Ricardo Reis and Luc Claesen editors, Chapman-Hall , 1997
[79] Ion Ştefănescu, "Celule de memorie digitală", Teză de doctorat, Institutul de Fizică Atomică, Bucureşti, 1974

[80] Gheorghe Ştefan, Virgil Bistriceanu, "Circuite digitale – probleme, proiectare", Casa de Editură Albastră, Cluj-Napoca, 2000
[81] John F. Wakerly, "Circuite digitale / Principiile şi practicile folosite în proiectare", Teora, România, 2002
[82] Yinghua Li, Alex Kondratyev, Robert K. Brayton , "Clockless Implementation Structure And Methodology For Dsm Implementation", Proceedings of IWLS '04, pp. 92-98, 2004
[83] Sasan Iman, Massoud Pedram, "Combinational Circuit Optimization", Kluwer Nato Advanced Science Institutes Series, pp. 267 – 320, ISBN:0-7923-4569-X, Kluwer Academic Publishers Norwell, MA, USA, 1997
[84] Randy H. Katz, "Contemporary Logic Design", The Benjamin/Cummings Publishing Company, Inc., USA, 1994.
[85] Yinghua Li, Alex Kondratyev, Robert K. Brayton , "Gaining Predictability And Noise Immunity In Global Interconnects", Proc. 5th Intl. Conf. on Application of Concurrency to System Design, Los Alamitos, CA: IEEE Computer Society, pp. 176-185, 2005
[86] SPFD-Based Wire Removal in Standard-Cell and Network-of-PLA Circuits, "Spfd-Based Wire Removal In Standard-Cell And Network-Of-Pla Circuits", IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems, Vol. 23, No. 7, July 2004
[87] Jean Goubault-Larrecq, "Implementing Tableaux By Decision Diagrams", Institut fur Logik, Komplexitat und Deduktionssysteme Universitat Karlsruhe, D-76128 Karlsruhe , August 29, 1996
[88] Sebastien Collette, Jean-Franc¸ois Raskin, Frederic Servais, "On The Symbolic Computation Of The Hardest Configurations Of The Rush Hour Game", In Proceedings of the 5th International Conference on Computers and Games (CG 2006), volume 4630 of LNCS, pp. 220-233, 2007
[89] Karsten Strehl , "Symbolic Methods Applied To Formal Verification And Synthesis In Embedded Systems Design", Thesis (doctoral), Swiss Federal Institute of Technology Zurich, , February 17, 2000
[90] N. Reus, "Tratarea simbolică a schemelor de comutaţie", Editura Academiei Republicii Socialiste Romania, 1971.
[91] Congguang Yang, Maciej Ciesielski, Vigyan Singhal, "Bds: A Bdd based Logic Optimization System", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on Volume 21, pp: 866 - 876 , July 2002
[92] Riera Baburés, Jordi, "Mapatge tecnològic orientat a generadors de mòduls - Capitol 6 Cobertura De La Xaraxa Booleana", Tesi, Universitat Autonoma de Barcelona, ISBN 84-699-9864-1, 35311
[93] Victor N. Kravets, Karem A. Sakallah , "Constructive Multilevel Logic Synthesis Under Properties Of Boolean Algebra", Proceedings of the 35th annual conference on Design automation, San Francisco, California, United States, pp: 336 – 341, ISBN:0-89791-964-5, 1998
[94] Alan Mishchenko, Satrajit Chatterjee, Robert Brayton, "Dag-Aware Aig Rewriting A Fresh Look At Combinational Logic Synthesis", Proc. DAC'06, pages 532–536 , 2006
[95] Yunjian Jiang, Robert K. Brayton , "Don't Cares And Multi-Valued Logic Network Minimization", International Conference on Computer Aided Design, ICCAD-2000, pp. 520-525, San Jose, CA, USA, ISBN: 0-7803-6445-7 , 2000
[96] Fan Mo, Robert Brayton, "Pla-Based Regular Structures And Their Synthesis", IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems, Vol. 22, No. 6, June 2003
[97] JieHong, R. Jiang, Robert K. Brayton, "Retiming And Resynthesis: A Complexity Perspective", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume: 25, Issue: 12, pp. 2674-2686, Sonoma, CA, USA, ISSN: 0278-0070, INSPEC Accession Number: 9311121, 39052
[98] J. Zhang, S. Sinha, A. Mishchenko, R. Brayton, M. Chrzanowska-Jeske, "Simulation And Satisfiability In Logic Sysnthesis", The IEEE International Workshop on Logic Synthesis, 2005
[99] Fan Mo, Robert K. Brayton, "Whirlpool Plas: A Regular Logic Structure And Their Synthesis", Proc. ICCAD '02, pp. 543-550, 2002
[100] Alan Mishchenko, Satrajit Chatterjee, Robert Brayton, Maciej Ciesielski, "An Integrated Technology Mapping Environment", Proceedings of the IWLS'05, pp. 383-390, 2005

[101] Subhasish Mitra and Edward J. McCluskey, "Combinational Logic Synthesis For Diversity In Duplex Systems", Test Conference, Proceedings. International, City, NJ, USA ISBN: 0-7803- 6546-1, 179-188, 2000
[102] Alan Mishchenko, Sungmin Cho, Satrajit Chatterjee, Robert Brayton , "Cutless Fpga Mapping", ERL Technical Report, EECS Dept., UC Berkeley, 2007
[103] Radhaselvi Venkatesan , "FPGA Implementation Of RNS Structures", Thesis, University of Windsor, Windsor, Ontario, , December, 1994
[104] Alan Mishchenko, Satrajit Chatterjee, Robert Brayton, "Improvements To Technology Mapping For Lut-Based Fpgas", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Sonoma, CA, USA, Volume: 26, Issue: 2, pp. 240-253, ISSN: 0278-0070, 39114
[105] Ahmad A. Al-Yamani, Nahmsuk Oh, Edward J. McCluskey, "Algorithm-Based Fault Tolerance: A Performance Perspective Based On Error Rate", Fast Abstract, International Symposium on Dependable Systmes and Networks (DSN'01),Gotenberg, Sweden, July 1-4, 2001
[106] Hutton, M.D. Rose, J.S. Corneil, D.G, "Automatic Generation Of Synthetic Sequential Benchmark Circuits", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems Volume 21, Issue 8, pp: 928-940, 37469
[107] Michael Hutton, J.P. Grossman, Jonathan Rose and Derek Corneil, "Characterization And Parameterized Generation Of Synthetic Combinational Benchmark Circuits", Proc. 33rd ACM/SIGDA Design Automation Conference (DAC), pp. 94-99, June 1996
[108] Michael Huttony, J.P. Grossmanz, Jonathan Rose , Derek Corneily, "Characterization And Parameterized Random Generation Of Digital Circuits", Proceedings of the 33rd annual conference on Design automation, Las Vegas, Nevada, United States, pp: 94 – 99, ISBN:0-89791-779-0, 1996
[109] Fan Mo, Robert K. Brayton, "Checkerboard: A Regular Structure And Its Synthesis", Proceedings of IWLS '03, pp. 7-13, 2003
[110] Michael Huttony, Jonathan Rosez, Derek Corneily, "Generation Of Synthetic Sequential Benchmark Circuits", Proceedings of the 1997 ACM fifth International Symposium on Field Programmable Gate Arrays, Monterey, California, United States Pages: 149-155, ISBN:0-89791-801-0, 1997
[111] Kalyana R. Kantipudi , "Minimizing N-Detect Tests For Combinational Circuits", Thesis Submitted to the Graduate Faculty of Auburn University in Partial Fulfillment of the Requirements for the Degree of Master of Science, Auburn, Alabama, May 10, 2007
[112] S. Dasgupta, C. H. Papadimitriou, U. V. Vazirani, "Algorithms", http://www.cs.berkeley.edu/~vazirani/algorithms.html, May 22, 2006
[113] F. Mo, R. Brayton, "An Integrated Standard-Cell Physical Design Algorithm", ERL Technical Report, EECS Dept., UC Berkeley, 2004
[114] Alan Mishchenko, Satrajit Chatterjee, Robert Brayton, Niklas Een, "Improvements To Combinational Equivalence Checking", International Conference on Computer-Aided Design, 2006. ICCAD '06. IEEE/ACM, San Jose, CA, pp. 36-843, ISSN: 1092-3152 ISBN: 1-59593-389-1 , Nov. 2006
[115] Alan Mishchenko, Robert Brayton, "Verification After Synthesis", Proc. IWLS '06, pp. 263-267, 2006
[116] S. Sinha, X. Wang, R. Brayton , "Comparing Two Rewiring Models", IWLS 2004, May, 2004
[117] Michael D. Hutton, Jonathan Rose , "Equivalence Classes Of Clone Circuits For Physical-Design Benchmarking", Proceedings of the 1999 IEEE International Symposium on Circuits and Systems, 1999. ISCAS apos 99. Volume 6, Jul 1999 pp. 428 - 431, 1999
[118] Adnan Aziz, Vigyan Singhal, Felice Balarin Robert K. Brayton, Alberto L. Sangiovanni-Vincentelli, O. Kupferman, M.Y. Vardi, "Equivalences For Fair Kripke Structures", Proc. 8th Int. Conference Computer Aided Verification, , volume 1102 of Lecture Notes in Computer Science, pages 372--382. Springer-Verlag, 1996
[119] Adnan Aziz.. , "Formal Methods in VLSI System Design", PhD thesis, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, , May 1996

[120] HoonSang Jin, Fabio Somenzi, "Circus: A Hybrid Satisfiability Solver", International Conference on Theory and Applications of Satisfiability Testing (SAT 2004), Vancouver, Canada, May 2004
[121] Aziz, F. Balarin, S.-T. Cheng, R. Hojati, T. Kam, S. C. Krishnan, R. K. Ranjan, T. R. Shiple, V. Singhal, S. Tasiran, H.-Y. Wang, R. K. Brayton, A. L. Sangiovanni-Vincentelli, "HSIS: A Bdd Based Environment For Formal Veri Cation ", Proceedings of the 31st Annual ACM IEEE Design Automation Conference DAC '94, San Diego, California, United States, pp. 454 – 459, ISBN:0-89791-653-0 , June 1994
[122] Alan Mishchenko, Satrajit Chatterjee, Jie-Hong Jiang, Robert Brayton, "Integrating Logic Synthesis, Technology Mapping, And Retiming", Proc. IWLS '05, 2005
[123] Jie-Hong R. Jiang, Robert K. Brayton, "On The Verification Of Sequential Equivalence", IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems, Vol. 22, No. 6, June 2003
[124] Franc Brglez, Jason A. Osborne, , "Performance Testing Of Combinatorial Solvers With Isomorph Class Instances", Workshop on Experimental Computer Science (San Diego, 13-14 June 2007
[125] Clarke, E. M.; Jha, S., "Symmetry And Induction In Model Checking", Lecture Notes in Computer Science, vol. 1000, Springer-Verlag, New York, 1996
[126] Alan Mishchenko, Robert K. Brayton, "A Boolean Paradigm in Multi-Valued Logic Synthesis", In the Notes of the International Workshop on Logic Synthesis, New Orleans, June 2002
[127] Rudiger Ebendt,Wolfgang Gunther, Rolf Drechsler, "Combination Of Lower Bounds In Exact Bdd Minimization", Design, Automation and Test in Europe Conference and Exhibition, pp. 758- 763, ISSN: 1530-1591, ISBN: 0-7695-1870-2, 2003
[128] Bernd Becker, Rolf Drechsler, Michael Theobald , "Minimization of 2-level AND\XOR Expressions using Ordered Kronecker Functional Decision Diagrams", ,
[129] Elena Dubrova, Yunjian Jiang, Robert Brayton, "Minimization of Multiple-Valued Functions in Post Algebra", In the Notes of the International Workshop on Logic Synthesis, Tahoe City, June 2001.
[130] Michael D. Hutton, "Upward Planar Drawing Of Single Source Acyclic Digraphs", Symposium on Discrete Algorithms, Proceedings of the second annual ACM-SIAM symposium on Discrete algorithms, San Francisco, California, United States, pp. 203 – 211, ISBN:0-89791-376-0, 1991
[131] Michael D. Hutton, Anna Lubiw , "Upward Planar Drawing Of Single Source Acyclic Digraphs", SIAM journal on computing (SIAM j. comput.), ISSN 0097-5397, vol. 25, no2, pp. 291-311 , 1996
[132] R. Rudell, "Multiple-Valued Minimization for PLA Synthesis", Master´s Report, University of California, Berkeley, or: IEEE Trans. CAD, vol. CAD-6. no. 5, 1987, pag. 727-750 , June 1986
[133] A. Mishchenko and R. K. Brayton, "Higher-Order Flexibilities In Multi-Valued Networks", ERL Technical Report, EECS Dept., UC Berkeley, May 2002
[134] D. Michael Miller, DavidY. Feinstein, Mitchell A. Thornton, "QMDD Minimization Using Shifting For Variable Reordering", Journal of Multiple-Valued Logic and Soft Computing, vol. 13, no. 4-6, pp. 537- 552, 2007
[135] S. Nagayama, T. Sasao, Y. Iguchi, and M. Matsuura, "Area-Time Complexities of Multivalues Decision Diagrams", IEICE Transactions on Fundamentals of Electronics, Vol.E87-A, No.5, pp. 1020-1028, May, 2004
[136] Bwolen Yang, Randal E. Bryant, David R. O'Hallaron, Armin Biere, Olivier Coudert, Geert Janssen, Rajeev K. Ranjan, Fabio Somenzi, "A Performance Study Of Bdd-Based Model Checking", Proceedings of the Second International Conference on Formal Methods in Computer-Aided Design pp: 255 - 289 ISBN:3-540-65191-8, 1998
[137] Randal E. Bryant, "Symbolic Boolean Manipulation With Ordered Binary Decision Diagrams", ACM Computing Surveys, Vol. 24, No. 3, pp. 293–318, September, 1992
[138] Jagesh V. Sanghavi, Rajeev K. Ranjan, Robert K. Brayton and Alberto Sangiovanni-Vincentelli, "High Performance BDD Package Based on Exploiting Memory Hierarchy", Proceedings of ACM/IEEE Design Automation Conference , June 1996

[139] Rajeev K. Ranjan, Jagesh V. Sanghavi, Robert. K. Brayton and Alberto Sangiovanni-Vincentelli, "Binary Decision Diagrams on Network of Workstations", Proceedings of IEEE/ACM International Conference on Computer Design, , Austin, TX, Oct. 1996
[140] Rajeev K. Ranjan, Wilsin Gosti, Robert. K. Brayton and Alberto Sangiovanni-Vincentelli, "Dynamic Reordering in a Breadth-First Manipulation Based BDD Package: Challenges and Solutions", Proceedings of IEEE/ACM International Conference on Computer Design Austin, TX, Oct. 1997
[141] J.Lind-Nielsen , "Verification Of Large State Event Systems Using Compositionality And Dependency Analysis", Tools and Algorithms for the Construction and Analysis of Systems '98 (TACAS'98). LICS 1384, Springer-Verlag , 1998
[142] Minxi Gao, Jie-Hong Jiang, Yunjian Jiang, Yinghua Li, Subarna Sinha, Robert Brayton, "MVSIS", Notes of the International Workshop on Logic Synthesis, Tahoe City, June 2001
[143] Donald Chai, Jie-Hong Jiang, Yunjian Jiang, Yinghua Li, Alan Mishchenko, Robert Brayton, "Mvsis 2.0 Programmer's Manual", http://embedded.eecs.berkeley.edu/Respep/Research/mvsis/doc/mvsis_20_prog.pdf,
[144] Yunjian Jiang, Robert Brayton, "MVSIS Code Generation Manual", http://embedded.eecs.berkeley.edu/mvsis/doc/genc_10_manual.pdf , June 2002
[145] Alan Mishchenko, Robert K. Brayton, "A Theory Of Non-Deterministic Networks", International Conference on Computer Aided Design, ICCAD-2003, pp: 709- 716, ISBN: 1-58113-762-1 , 2003
[146] Mischenko, Alan, Robert Bryton., "A theory of nondeterministic networks.", Vol. 25. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, June 2006
[147] M. Gao, J-H. Jiang, Y. Jiang, Y. Li, A. Mishchenko, S. Sinha, T. Villa, R. Brayton, "Optimization of Multi-Valued Multi-Level Networks ", In the Proceedings of the International Symposium on Multiple-Valued Logic, 2002.
[148] Robert K. Brayton (1) , Jie-Hong R. Jiang (1), Alan Mishchenko (1), Tiziano Villa (2), Nina Yevtushenko (3) , "BALM User's Manual – ", University of California, Berkeley, (2)University of Udine, Italy, (3)Tomsk State University, Russia., xxx
[149] Popel, Denis V., Rolf Drechsler., "Efficient Minimization of Multiple-valued Decision Diagrams.", Multiple-Valued Logic, 2003. Proceedings. 33rd International Symposium., May 2003
[150] ***, "Input file format for Espresso (PLA)- Espresso (5).", ,
[151] Robert K. Brayton, Sunil P. Khatri, "Multi-Valued Logic Synthesis", Proceedings On Twelfth International Conference Volume , pp. 196 - 205, 7-10 Jan 1999
[152] Jie-Hong Jiang, Yunjian Jiang, Robert K. Brayton, "An implicit method for multi-valued network encoding", In the Notes of the International Workshop on Logic Synthesis, Tahoe City, June 2001
[153] Bollig, B., I. Wegener, " Improving the variable ordering of OBDDs is NP-Complete ", IEEE Transactions on Computers, 993 - 1002, September 1996
[154] Quine, W. V., "The Problem of Simplifying Truth Functions.", Edited by Vol. 59, No. 8 The American Mathematical Monthly. pp: 521-531, Oct, 1952
[155] S. J. Hong, R. G. Cain, D. L. Ostapco, "MINI: A Heuristic Approach for Logic Minimization", IBM J. of Res. And Dev. , Sept. 1974.
[156] R. Rudell, A. Sangiovanni-Vincentelli, "Espresso-MV: Algorithms for Multiple-Valued Logic Minimization", Proc. Intl. Circ. Conf., Portland, May 1985.
[157] Oliveira, A.L.; Carloni, L.P.; Villa, T.; Sangiovanni-Vincentelli, A.L. , "Exact Minimization Of Binary Decision Diagrams Using Implicit Techniques", Transactions on Computers, Volume 47, pp: 1282-1296, ISSN: 0018-9340, Nov 1998
[158] R. Rudell, A. Sangiovanni-Vincentelli, "Exact Minimization of Multiple-Valued Functions for PLA Optimization", Intl. Conf. On Comp. Aided Design, Santa Clara, November 1986.
[159] Alan Mishchenko, Satrajit Chatterjee, Roland Jiang, Robert Brayton, "Fraigs: A Unifying Representation For Logic Synthesis And Verification", Proceedings of the 2006 ACM/SIGDA 14th International Symposium on Field Programmable Gate Arrays, Monterey, California, USA, pp: 41 – 49, ISBN:1-59593-292-5 , 2006

[160] Jie-Hong R. Jiang, Robert K. Brayton, "Functional Dependency For Verification Reduction", Proc. CAV '04, pp. 268-280, 2004
[161] R. K. Brayton, S. P. Khatri, "Multi-valued Logic Synthesis", International Conference on VLSI Design, Goa, India, Jan 1999.
[162] Y. Li, R. Bryton, "Multi-Valued Optimization On Post Logic Networks", 2003
[163] Luca P. Carloni, Evguenii I. Goldberg, Tiziano Villa, Robert K. Brayton, Alberto L. Sangiovanni-Vincentelli, "Combining negative thinking and Branch-And-Bound in unite covering problems", VLSI: Systems on a Chip, L.M. Silveira, R. Reis, S. Devadas editors, Kluwer , 1999
[164] Shivakumaraiah Lokesh, Mitchell A. Thornton, "Computation Of Disjoint Cube Representations Using A Maximal Binate Variable Heuristic", M.A. System Theory, 2002. Proceedings of the Thirty-Fourth Southeastern Symposium, 2002
[165] Aaron P. Hurst, Robert K. Brayton , "Computing Clock Skew Schedules Under Normal Process Variation", Proceedings of IWLS '05, 2005
[166] Jie-Hong Jiang, Yunjian Jiang, Yinghua Li, Alan Mishchenko, Subarna Sinha Tiziano Villa, Robert Brayton, "Negative Thinking By Incremental Problem Solving: Application To Unate Covering", http://web.cecs.pdx.edu/~mperkows/PerkowskiGoogle/mvsis_11_manual.pdf ,
[167] Goldberg, E.I., Carloni L.P., Villa T., Brayton R.K. ., "Negative Thinking by Incremental Problem Solving: Application to Unate Covering", Proceedings of IEEE International Conference on Computer Aided Design (ICCAD), San Jose, CA, USA, 9-13 Nov. 1997, pp. 91-99., 1997
[168] Evguenii, I. Goldberg, P. Carloni Luca, K. Bryton Robert, L. Sangiovanni-Vincentelli Alberto, " Negative Thinking in Branch-and-Bound: the Case of Unate Covering", IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems, vol. 19, no. 3 , 2000
[169] Alan Mishchenko, Robert K. Brayton, Tsutomu Sasao , "Exploring Multi-Valued Minimization Using Binary Methods", Proc. IWLS '03, pp. 278-285, 2003
[170] Satrajit Chatterjee, Alan Mishchenko, Robert Brayton, "Factor Cuts", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 26, Issue 2, pp:240 - 253, 39114
[171] Michael Theobald, Steven M. Nowick , "Fast Heuristic And Exact Algorithms For Two-Level Hazard-Free Logic Minimization", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 17, Issue 11, pp.1130 - 1147, Nov 1998
[172] Alan Mishchenko Jin S. Zhang, Subarna Sinha, Jerry R. Burch, Robert Brayton, Malgorzata Chrzanowska-Jeske, "Using Simulation And Satisfiability To Compute Flexibilities In Boolean Networks", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on Volume 25, Issue 5, pp: 743–755, May 2006
[173] Petr Fiser, Hana Kubatova, "Boolean Minimizer Fc-Min: Coverage Finding Process", Euromicro Symposium on Digital System Design. Piscataway: IEEE, 2004, p. 152-159. ISBN 0-7695-2203-3, 2004
[174] P. Fiser, J. Hlavicka, H. Kubatova , "Fc-Min: A Fast Multi-Output Boolean Minimizer", Proceedings Euromicro Symposium on Digital System Design, 2003
[175] Petr Fišer, Hana Kubátová, "Two-Level Boolean Minimizer Boom-Ii", Proc. 6th International Workshop, Freiberg University of Mining and Technology, Institute of Computer Science, pp. 221-228. ISBN 3-86012-233-9, 2004
[176] Ion Ştefănescu, "New Technical Aspects in Identifying and Eliminating of Logic and Essential Hazards of the Combinational Networks", 1st International Conference on Electronics, Computers and Artificial Intelligence – ECAI 2005 – University of Piteşti, Romania, 1-2 July 2005.
[177] STEFĂNESCU, Ion. , "Proiectarea logica a sistemelor decizionale hardware şi software – Aspecte de baza (Logic Design of Hardware and Software Decisional Systems – Basic Aspects). ", Bucuresti : Matrix Rom, 2006.
[178] Laurenţiu Ionescu, **Adrian Zafiu**, Ion Ştefănescu, "A Comparison between Traditional and Discriminaton Methods in the Synthesis of a Maximum and Minimum Circuit for Implementation on a Field Programmable Gate Array", 1st International Conference on Electronics, Computers and Artificial Intelligence – ECAI 2005 – University of Piteşti, Romania, 1-2 July 2005

[179] Anavai Ramesh, George Becker, Neil V. Murray , "CNF And DNF Considered Harmful For Computing Prime Implicants / Implicates", Proceedings of the 4th International Conference on Logic Programming and Automated Reasoning, St. Petersburg, Russia, July 13-20, 1993
[180] Christian Plessl, "Reconfigurable Accelerators For Minimum Covering Problems", Diploma Thesis DA-2001.10, ftp://ftp.tik.ee.ethz.ch/pub/people/plessl/plessl01_diplomathesis.pdf , 36965
[181] S. L. A. Czort, , "The Complexity Of Minimizing Dijunctive Normal Form Formulas", Master's Thesis, University of Aarhus, 1999
[182] Sinha, S.; Mishchenko, A.; Brayton, R.K. , "Topologically Constrained Logic Synthesis", IEEE/ACM International Conference on Computer Aided Design, ICCAD 2002, pp. 679-686, ISSN: 1092-3152, ISBN: 0-7803-7607-2 , 10-14 Nov. 2002
[183] S. R. Petrick, "A direct determination of the irredundant forms of a boolean function from the set of prime implicants", Technical Report AFCRC-TR-56-110, Air Force Cambridge Research Center, Cambridge, MA, April, 1956
[184] Raymond J. NELSON, "Weak Simplest Normal Truth Functions.", Journal of Symbolic Logic, vol. 20, no 3, pp. 232-234, 1955
[185] Raymond J. NELSON, "Simplest Normal Truth Functions.", Journal of Symbolic Logic, vol. 20, no 2, pp. 105-108, 1955
[186] B. Thelen, "Investigations of algorithms for computer-aided logic design of digital circuits, (in German)", PhD thesis, ITIV, Univ. of Karlsruhe, 1981
[187] Jacek Bieganowski, Andrei Karatkevich, "Heuristics For Thelen's Prime Implicant Method", Schedae Informaticae, Computer Edition, Volume 14, 2005
[188] Jacobson, H.M. Myers, C.J. , "Efficient Algorithms For Exact Two-Level Hazard-Free Logic Minimization", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 21, pp: 1269-1283, Nov 2002
[189] ANACHIP USA, Inc., "Hazard-Free Combinatorial Latches ", 2002
[190] Fuhrer, R.M.; Lin, B.; Nowick, S.M, "Symbolic Hazard-Free Minimization And Encoding Of Asynchronous Finite State Machines", IEEE/ACM International Conference on Computer-Aided Design, ICCAD-95, pp. 604 - 611, 5-9 Nov 1995
[191] **Zafiu, Adrian**, Ion STEFANESCU, Eduard FRANTI. , "A new method for combinational minimization of multi-valued decisional system specifications.", SERVICES and SOFTWARE ARCHITECTURES. Bucharest,: Romanian Academy, may 18, 2007. 28-37, 2007
[192] Olveira, Arlindo L., Luca P Carloni, Tiziano Villa, Alberto L. Sangiovanni-Vincentelli., "Exact Minimization of Binary Decision Diagrams.", IEEE Transaction Vol 47 Nr 11. 1998. 1282-1296., Nov 1998
[193] Drechsler, Rolf, Detlef Sieling, "Binary decision diagrams in theory and practice", University of Dortmund: Software Tools for Technology Transfer 3, Online , 2001
[194] Tomoya Kitai, Yusuke Oguro, Tomohiro Yoneda, Eric Mercer, Chris Myers, "Level Oriented Formal Model For Asynchronous Circuit Verification And Its Efficient Analysis Method", Dependable Computing, 2002. Proceedings. Pacific Rim International Symposium on 16-18 Dec. 2002, pp. 210- 218, ISBN: 0-7695-1852-4 , 37591
[195] Taylor L. Booth, "Sequential Machines and Automata Theory", John Wiley and Sons, Inc., USA, 1968.
[196] Andresen, Henrik Reif, "An Introduction to Binary Decision Diagrams", Technical University of Denmark: Lecture notes for Advanced Algorithms, October 1997
[197] , "Berkeley Logic Interchange Format (Blif)", University of California at Berkeley, July 28, 1992
[198] Kukimoto, Yuji., "BLIF-MV.", The VIS Group, University of California. Berkeley, May 31, 1996
[199] Giumale, Cristian A., "Introducere în analiza algoritmilor - Teorie şi aplicaţii.", Bucureşti: POLIROM,. 313-386, 2004
[200] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze, "An Introduction to Information Retrieval", Cambridge University Press. 2008 (the authors expect the book to appear in May or June of 2008.) , November 17, 2007
[201] Fan Mo, Robert K. Brayton, "River Plas: A Regular Circuit Structure", Design Automation Conference, 2002. Proceedings. 39th Volume , Issue , pp. 201 - 206, 2002

[202] Alan Mishchenko, Robert Brayton, "Scalable Logic Synthesis Using A Simple Circuit Structure", Proc. IWLS '06, pp. 15-22, 2003
[203] Arthur D. Friedman, Premachandran R. Menon, "Theory & Design of Switching Circuits", Computer Science Press, Inc., USA, 1975.
[204] M.C.A. Koster, „Decomposition of Graphs: Upper bounds, Lower bounds, and Exact methods to compute Treewidth Arie", Eighth International Workshop on Preferences and Soft Constraints (Soft-2006), Nantes, France, September 25th, 2006
[205] Stefano Bistarelli, Ugo Montanari, Francesca Rossi and Francesco Santini, „Modelling Multicast QoS Routing by using Best-Tree Search in AND-OR Graphs and Soft Constraint Logic Programming", Eighth International Workshop on Preferences and Soft Constraints (Soft-2006), Nantes, France, September 25th, 2006
[206] Philippe J´egou, Samba Ndojh Ndiaye and Cyril Terrioux , „Dynamic heuristics for branch and bound search on tree-decomposition of Weighted CSPs", Eighth International Workshop on Preferences and Soft Constraints (Soft-2006), Nantes, France, September 25th, 2006
[207] Nina Yevtushenko, Tiziano Villa, Robert K. Brayton, Alex Petrenko, Alberto L. Sangiovanni-Vincentelli, "Solution Of Parallel Language Equations For Logic Synthesis", IEEE/ACM International Conference on Computer Aided Design, ICCAD 2001, pp. 103-110, San Jose, CA, USA, ISBN: 0-7803-7247-6 , 2001
[208] Reiner Hahnle, Neil V. Murray, Erik Rosenthal, "Some Remarks On Completeness Connection Graph Resolution And Link Deletion", Lecture Notes in Computer Science, Springer Berlin / Heidelberg, ISSN 0302-9743, Volume 1397/1998, May 1998
[209] Michael Hutton, "The Theoretical Foundations Of Computer Science", , 1994
[210] Evgueni I. Goldberg, Mukul R. Prasad, Robert K. Brayton, "Using Problem Symmetry In Search Based Satisfiability Algorithms", Proceedings of Design, Automation and Test in Europe Conference and Exhibition, pp. 134 - 141, 2002
[211] Jie-Hong R. Jiang, Alan Mishchenko, Robert K. Brayton, "On Breakable Cyclic Definitions", Computer Aided Design, 2004. ICCAD-2004. IEEE/ACM International Conference on Volume , Issue , 7-11 Nov. 2004, pp: 411 – 418, ISSN: 1092-3152, ISBN: 0-7803-8702-3, 2004
[212] Renee-Louis Vallee, "Analyse binaire – 1ere partie – Definitions et traitements des fonctions binaires", Raport CEA-R$-3534 (I), C.E.A. Saclay, France, 1968
[213] Renee-Louis Vallee, "Analyse binaire – 2eme partie – Applications et fonctions de transcodage", Raport CEA-R$-3534 (2), C.E.A. Saclay, 1969
[214] Edward J. McCluskey, "Logic Design Principles with Emphasis on Testable Semicustom Circuits", Prentice/Hall, N.J., USA, 1986.
[215] R. Brayton, G. Hatchel, C. McMullen, A. Sangivanni-Vincentelli, " Logic Minimization Algorithms for VLSI Synthesis", Kluwer Academic Publishers, 1984.
[216] Saeyang Yang, "Logic Synthesis and Optimization Benchmarks User Guide Version 3.0", Technical Report, Microelectronics Center of North Carolina (MCNC), Jan. 1991
[217] Presentation of TOSMANA Adding Multi-Value Variables and Visual Aids to QCA, "Presentation Of Tosmana", Paper prepared for presentation at the COMPASSS Launching Conference in Louvain La Neuve and Leuven, Lasse Cronqvist, Institute for Political Science, University of Marburg, Germany, 16-17 Sept. 2003
[218] Jiang, Y., Robert K Brayton. , "Don't Cares and Multi-Valued Logic Network Minimization.", Santa Clara: IEEE/ACM International Conference on CAD ICCAD 00, November 2000.
[219] ŞTEFĂNESCU Ion, **Adrian ZAFIU**. , "An Efficient Network Strategy in Deep Minimizations of Deterministic and Nondeterministic Multivalued Decisional Systems - Principles and Comparative Results.", Electronics, Computers and Artificial Intelligence – ECAI'07. Ed. University of Pitesti. Pitesti, pg. 7-16., June 2007
[220] **Zafiu, Adrian**. , "An efficient generative algorithm developed for the minimization of the multi-valued specifications. ", Electronics, Computers and Artificial Intelligence, ECAI, June 2007
[221] **Zafiu, Adrian**, Ion Ştefănescu, Eduard Franti , "An algorithm to determine the multivalued implicant vectors with a guaranteed minimum number of specified entries.", Tenerife, Canary Islands, Spain: WSEAS Transaction on Computers Research Issue 2, Volume 1, 272-278., December 2006

[222] **Zafiu, Adrian**, Ion Ştefănescu, Holban Stefan , "An exact method to compute maximal implicants in a multivalued logic.", Tenerife, Canary Islands, Spain: 5th WSEAS Int. Conf. on System Science and Simulation in Engineering (ICOSSE '06), pp 49-52., December 2006

[223] Sentovich E. M., Singh K. J., Lavagno L., Moon C., Murgai R., Saldanha A., Savoj H., Stephan P. R., Brayton R. K., Sangiovanni-Vincentelli A. L.: , "A System for Sequential Circuit Synthesis.", Univ. of California, Berkeley, CA 94720: Tech. Rep. UCB/ERL M92/41 Electronics Research Lab, May 1992

[224] Tiberiu Mureşan, Aurel Gontean, Mircea Băbăiţă, Petru Demian, "Circuite integrate numerice – Aplicaţii şi proiectare", Editura de Vest, Timişoara, România, 2000

[225] P. Naslin, "Circuite logice şi automatizări secvenţiale", Editura Tehnică, Bucureşti, 1967

[226] Gheorghe Ştefan, "Circuite şi sisteme digitale", Editura tehnică, Bucureşti, România, 2000

[227] Nina Yevtushenko, Tiziano Villa, Robert K. Brayton, Alex Petrenko, Alberto L. Sangiovanni-Vincentelli , "Equisolvability Of Series Vs. Controller's Topology In Synchronous Language Equations", Design, Automation and Test in Europe Conference and Exhibition, 2003, pp. 1154 - 1155, 3-7 March 2003

[228] Vigyan Singhal, Carl Pixley, Adnan Aziz, Shaz Qadeer, Robert Brayton, , "Sequential Optimization In The Absence Of Global Reset", ACM Transactions on Design Automation of Electronic Systems (TODAES), Volume 8I, SSN:1084-4309, 37712

[229] Robert Mack Fuhrer , "Sequential Optimization Of Asynchronous And Synchronous Finite State Machines Algorithms And Tools", Kluwer Academic Publishers Norwell, MA, USA, ISBN:0792374258, 2001

[230] S. Sinha, A. Kuehlmann, R.K.Brayton, "Sequential SPFDs", IEEE/ACM International Conference on Computer Aided Design, 2001. ICCAD 2001. Volume , Issue , pp. 84 - 90, 2001

[231] N. Yevtushenko, T. Villa, Robert K. Brayton, A. Petrenko, Alberto L. Sangiovanni-Vincentelli, "Sequential Synthesis By Language Equation Solving", Technical Report No. UCB/ERL M03/9, University of California, Berkeley , 2003

[232] G. Wang, A. Mishchenko, R. Brayton, A. Sangiovanni-Vincentelli , "Sequential Synthesis With Co-Büchi Specifications", ERL Technical Report, EECS Dept., UC Berkeley, 38808

[233] H.-J.Mathony , "Universal logic design algorithm and its application to the synthesis of two-level switching circuits", IEE Proceedings of Computers and Digital Techniques, Vol 136, pp. 171-177, May , 1989

[234] **Adrian Zafiu**, Ion Stefanescu, Ionescu Laurentiu, Constantin Ghita, Eduard Franti, "An exact method to compute maximal implicants in a multivalued logic", 5th WSEAS Int. Conf. on System Science and Simulation in Engineering (ICOSSE '06), ISSN 1790-5117, ISBN 960-8457-57-2 Tenerife, Canary Islands, Spain, December 16-18, 2006, pg 249-254

[235] **Adrian Zafiu**, Monica Dascălu, Eduard Franti, Annely-Mihaela Zafiu, „Optimizing prosthesis design by using virtual environments", The 7th WSEAS International Conference on APPLIED INFORMATICS AND COMMUNICATIONS (AIC'07), Athens, September 2007,

[236] **Adrian Zafiu**, Stefan Holban, „A top-down minimization algorithm using discrimination principle for non-deterministic multivalued systems", The 7th WSEAS International Conference on APPLIED INFORMATICS AND COMMUNICATIONS (AIC'07), Athens, September 2007

[237] **Adrian Zafiu**, Ion Stefanescu, „A new framework for non-deterministic multivalued system minimization – techniques of optimization and minimization", The 7th WSEAS International Conference on APPLIED INFORMATICS AND COMMUNICATIONS (AIC'07), Athens, September 2007

[238] **Adrian Zafiu**, "New Algorithms and Specific Strategies for Minimization of Multivalued Nondeterministic Decisional Systems", International Journal of Internet Technology and Secured Transactions (IJITST), INDERSCIENCE Publishers, submitted July 2007

[239] Ion Stefanescu, **Adrian Zafiu**, "Network Minimization of Deterministic and Nondeterministic Multivalued Decisional Systems – Principles and Results of a New Approach", International Journal of Internet Technology and Secured Transactions (IJITST), submitted July 2007

[240] **Adrian Zafiu**, Monica Dascălu, Eduard Franţi, Annely Mihaela Zafiu, Bogdan Oancea, "Virtual Enviroments for Prosthesis Design", 6th WSEAS International Conference on COMPUTATIONAL INTELLIGENCE, MAN-MACHINE SYSTEMS and CYBERNETICS (CIMMACS '07), pp. 572-579, Tenerife, Canary Islands, Spain, December 14-16, 2007

[241] **Adrian Zafiu**, Ilie Popa, "A new approach of irredundant covering selection", Advances in Applied Mathemathics, Systems, Communications and Computers, pp. 68-72, Attica, Greece, June 1-3, 2008, ISBN 978-960-6766-69-5

[242] Ion Stefanescu, **Adrian Zafiu**, "A new approach to deterministic and non-deterministic decisional systems minimization – principles and results", Advances in Applied Mathemathics, Systems, Communications and Computers, pp. 240-251, Attica, Greece, June 1-3, 2008, ISBN 978-960-6766-69-5

[243] **Adrian Zafiu**, Constantin Ghita, "An efficient model for traffic simulation", Advances in Applied Mathemathics, Systems, Communications and Computers, pp. 265-269, Attica, Greece, June 1-3, 2008, ISBN 978-960-6766-69-5