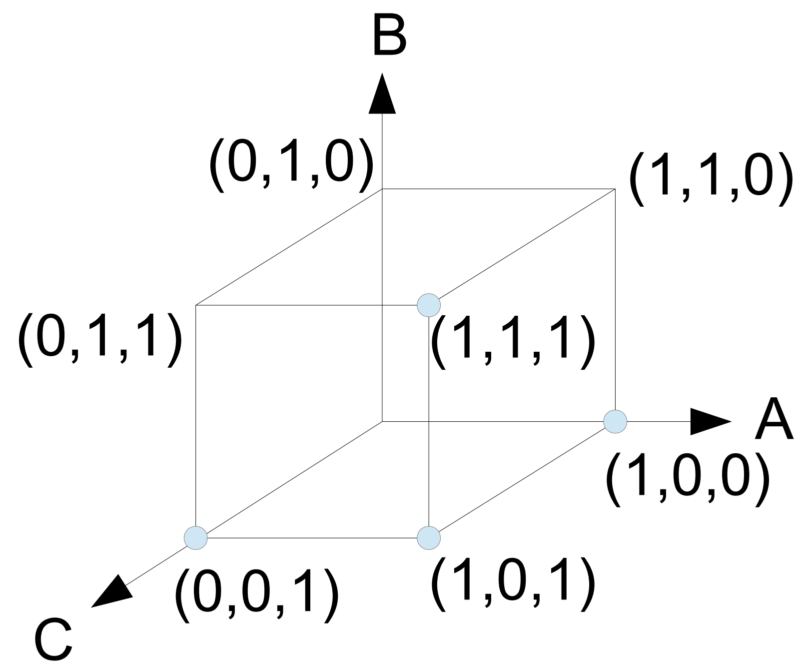
Reprezentarea funcţiilor logice

Expresii algebrice

- forma normală (canonică) disjunctivă:
  - $f(x_1, x_2, ..., x_n) = A_1 + A_2 + ... A_n$, unde A_i este minterm
  - $Y(A,B,C) = \overline{A} * \overline{B} * C + A * \overline{B} * \overline{C} + A * \overline{B} * C + A * B * C$
- forma echivalenţilor zecimali a mintermilor
  - $f(x_1, x_2, ..., x_n) = \{i \mid A_i = 1\}$
  - $f(A,B,C) = \{1,4,5,7\}$
- forma normală (canonică) conjunctivă:
  - $f(x_1, x_2, ..., x_n) = B_1 * B_2 * ... B_n$, unde B_i este maxterm
  - $Y(A,B,C) = (A+B+\overline{C}) * (\overline{A}+B+C) * (\overline{A} + B + \overline{C}) * (\overline{A} + \overline{B} + \overline{C})$

Cuburi booleene

- Greu de utilizat pentru funcţii cu mai multe variabile
- Reprezentare grafică, important din punct de vedere istoric – a condus la diagrame Karnaugh
- Exemplu:
  - O funcţie Y, cu trei variabile A, B şi C



Tabel de adevăr

- Coloane cu combinaţiile posibile ale valorilor + o coloană cu valoarea funcţiei
- Exemplu:
  - O funcţie Y, cu trei variabile de intrare A, B şi C

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

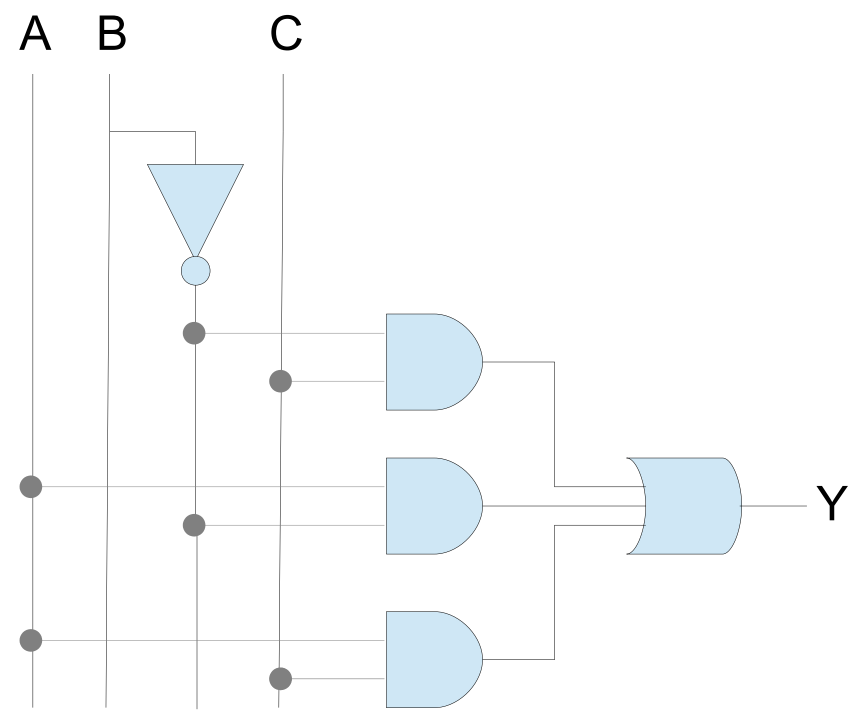
Diagrame Karnaugh

- O reprezentare tabelară a proiecţiei unui cub boolean
- “sistem de coordonate”, pe axe valorile sunt aranjate in codare Gray
- Exemplu:
  - O funcţie Y, cu trei variabile de intrare A, B şi C

A \ BC	0 0	0 1	1 1	1 0
0	$\overline{ABC}$	$\overline{AB}C$	$\overline{A}BC$	$\overline{A}B\overline{C}$
1	$A\overline{BC}$	$A\overline{B}C$	ABC	$AB\overline{C}$

Circuite logice

- O reprezentare garfică, uşor de obţinut atât din reprezentări algebrice, cât şi din diagrame Karnaugh
- Exemplu:
  - O funcţie Y, cu trei variabile de intrare A, B şi C



Minimizarea funcțiilor logice

Minimizarea funcţiilor logice

- Obiectivul minimizării logice este reducerea mărimii reprezentării funcţiilor Booleene în oricare din formele de reprezentare sume de produse ori produse de sume.
- Se poate remarca faptul că oricare din cele două forme ale funcţiilor Booleene poate fi dedusă din cealaltă cu ajutorul legilor De Morgan iar această transformare păstrează numărul de termeni şi de literali.
- Categorii
  - Pe două nivele (minimizarea Karnaugh, Quine-McCluskey, minimizarea euristică)
  - Multinivel (metode simbolice)

Definiții

- Literal: variabilă sau forma negată a variabilei
- Implicant: este un grup ce se poate încercui într-o diagrama Karnaugh, un termen (chiar un minterm) care face parte din funcție logică
- Implicanţi prim: scoaterea unui literal din implicant, grupul rezultat nu mai acopera doar valori de 1, ci conţine şi valori de 0
- Acoperire: o acoprire a unei funcții booleane este o multime de implicanți care conține toți termenii a funcției
- Acoperire minimă: acoperire
 cu număr minim de implicanți

A \ BC	0 0	0 1	1 1	1 0
0	1	1	0	0
1	1	1	1	0

Metode grafice (diagrame Karnaugh)

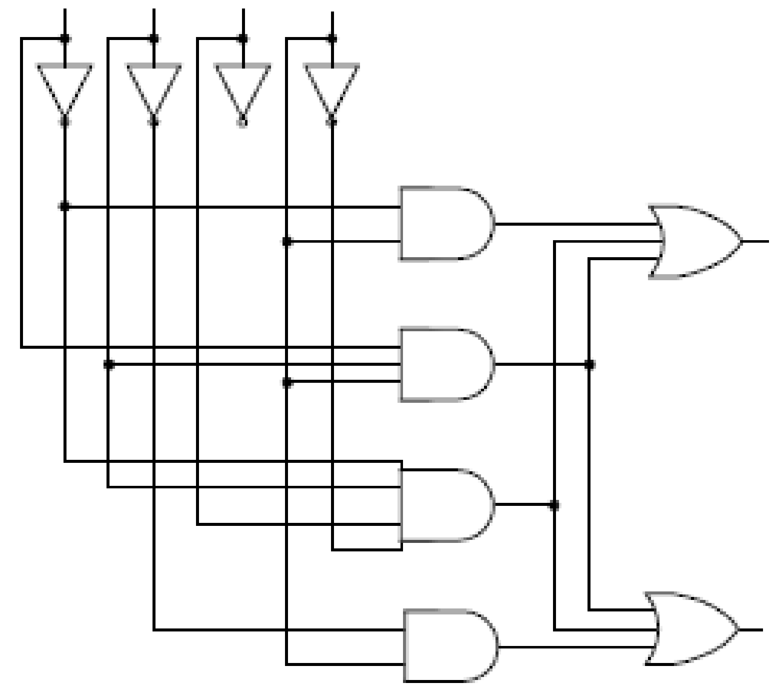
- Valorile în diagramele Karnaugh aşa sunt plasate, încât două poziţii diferă doar într-un singur bit
- Factorizarea mintermilor este mai uşor:
  - $A\overline{B}C + ABC = AC(\overline{B}+B)$

CD \ AB	0 0	0 1	1 1	1 0
0 0	0	0	0	0
0 1	1	1	1	0
1 1	1	1	1	0
1 0	0	1	0	0

CD \ AB	0 0	0 1	1 1	1 0
0 0	0	0	0	0
0 1	1	0	1	1
1 1	1	0	1	1
1 0	0	1	0	0

Metode grafice (diagrame Karnaugh)

- Minimizarea grafică a funcţiilor cu mai multe ieşiri:
  - se caută minimizarea unei perechi de funcţii logice:
    - $Y_1 = \overline{A} * D + B * D + \overline{A} * B * C$
    - $Y_2 = A * D + B * D + \overline{A} * B * C * \overline{D}$.
  - Realizările individuale costă 29 de unităţi:
    - Y_1 costă 1 poartă SAU, 3 porţi ŞI şi 14 fire de legătură
    - Y_2 costă 1 poartă SAU, 3 porţi ŞI şi 11 fire de legătură



Metode grafice (diagrame Karnaugh)

- Minimizarea grafică a funcţiilor cu mai multe ieşiri:
  - Indetificarea părţilor comune a celor două funcţii
    - $Y_1 = \overline{A} * D + A * B * D + \overline{A} * B * C * \overline{D}$
    - $Y_2 = A * D + A * B * D + \overline{A} * B * C * \overline{D}$
  - implementarea comună a funcţiilor necesită doar 2 porţi SAU, 4 porţi ŞI şi 17 fire, per total costul fiind 23

Metoda Quine – McCluskey

- Minimizarea prin această metodă presupune parcurgerea paşilor următoare:
  - ordonarea mintermilor
  - determinarea implicaţilor primi
  - determinarea tabelului de acoperire a funcţiei.
  - calculul formal de determinare a tuturor soluţiilor funcţiei
- Exemplu
  - F(A,B,C,D,E) = {1,3,5,7,8,11,12,17,19,21,23,24,28}

Metoda Quine – McCluskey

- Exemplu
  - Ordonarea termenilor

1.	ponderea
1 8	1
3 5 12 17 24	2
7 11 19 21 28	3
23	4

Metoda Quine – McCluskey

- Exemplu
  - determinarea implicaţilor primi
    - sunt comparate implicanţii care figurează în grupuri consecutive
    - toate elementele grupului cu pondere mai mică sunt comparate cu toate elementele grupului cu pondere mai mare

1.	2.
1✓	1,3 (2)
8	
3✓	
5	
12	
17	
24	
7	
11	
19	
21	
28	
23	

Metoda Quine – McCluskey

- Exemplu
  - determinarea implicaţilor primi
    - condiţiile de unificare a doi implicanţi:
      - diferenţa ponderilor este 1;
      - diferenţa valorilor decimale ale implicanţilor este egală cu o putere a lui 2 (adică 1, 2, 4, 8, 16, etc);
      - valoarea decimala a mintermului cu pondere mai mare trebuie sa fie mai mare;
      - toate aceste condiţii trebui să se indeplinească în aceeaşi moment.

1.	2.
1✓	1,3 (2)
8	
3✓	
5	
12	
17	
24	
7	
11	
19	
21	
28	
23	

Metoda Quine – McCluskey

- Exemplu
  - determinarea implicaţilor primi

1.	2.
1✓	1,3 (2)
8✓	1,5 (4)
3✓	1, 17 (16)
5✓	8, 12 (4)
12✓	8, 24 (16)
17✓	3, 7 (4)
24✓	3, 11 (8)
7✓	3, 19 (16)
11✓	5, 7 (2)
19✓	5, 21 (16)
21✓	12,28 (16)
28✓	17, 19 (2)
23✓	17, 21 (4)
	24, 28 (4)
	7, 23 (16)
	19, 23 (4)
	21,23 (2)

Metoda Quine – McCluskey

- Exemplu
  - determinarea implicaţilor primi

1.	2.	3.
1✓	1,3 (2)✓	1,3,5,7 (2,4)
8✓	1,5 (4)✓	1,3,17,19 (2,16)
3✓	1, 17 (16)✓	1,5,17,21 (4,16)
5✓	8, 12 (4)✓	8,12,24,28 (4,16)
12✓	8, 24 (16)✓	3,7,19,23 (4,16)
17✓	3, 7 (4)✓	5,7,21,23 (2,16)
24✓	3, 11 (8) (a)	17,19,21,23 (2,4)
7✓	3, 19 (16)✓	
11✓	5, 7 (2)✓	
19✓	5, 21 (16)✓	
21✓	12,28 (16)✓	
28✓	17, 19 (2)✓	
23✓	17, 21 (4)✓	
	24, 28 (4)✓	
	7, 23 (16)✓	
	19, 23 (4)✓	
	21,23 (2)✓	

Metoda Quine – McCluskey

- Exe
  - c
    i

1.	2.	3.	4.
1✓	1,3 (2)✓	1,3,5,7 (2,4)✓	1,3,5,7,17,19,21,23 (2,4,16). (c)
8✓	1,5 (4)✓	1,3,17,19 (2,16)✓	
3✓	1, 17 (16)✓	1,5,17,21 (4,16)✓	
5✓	8, 12 (4)✓	8,12,24,28 (4,16) (b)	
12✓	8, 24 (16)✓	3,7,19,23 (4,16)✓	
17✓	3, 7 (4)✓	5,7,21,23 (2,16)✓	
24✓	3, 11 (8) (a)	17,19,21,23 (2,4)✓	
7✓	3, 19 (16)✓		
11✓	5, 7 (4)✓		
19✓	5, 21 (16)✓		
21✓	12,28 (16)✓		
28✓	17, 19 (2)✓		
23✓	17, 21 (4)✓		
	24, 28 (4)✓		
	7, 23 (16)✓		
	19, 23 (4)✓		
	21,23 (2)✓		

Metoda Quine – McCluskey

- Exemplu
  - Determinarea unei acoperiri
  - Pe prima coloană se scriu implicanţii găsiți, iar pe prima linie termenii funcţiei.
  - Pentru fiecare implicant prim se notează în tabel (matricea de incidenta) care sunt termenii acoperiţi de el.

	1	3	5	7	8	11	12	17	19	21	23	24	28
a		x				x							
b					x		x					x	x
c	x	x	x	x				x	x	x	x		

Metoda Quine – McCluskey

- Exemplu
  - Determinarea unei acoperiri
  - Se poate construi o funcţie complementară X pentru a determina implicanţilor necesari:
  - X = (c) (a + c) (c) (c) (b) (a) (b) (c) (c) (c) (c) (b) (b)
  - X = a b c

	1	3	5	7	8	11	12	17	19	21	23	24	28
a		x				x							
b					x		x					x	x
c	x	x	x	x				x	x	x	x		

Metoda Quine – McCluskey

- Exemplu
  - Transcierea implicanţilor în variabilele funcţiei
  - Primul implicant a 3,11 (8).
    - Se scrie un minterm în formă disjunctivă 3 este $\overline{ABC}DE$
    - Diferenţa calculată încă la gruparea mintermilor, arată care dintre variabilă a funcţiei se şterge: diferenţa fiind 8, se şterge variabila cu ponderea 8, adică variabila B
    - Se obţine expresia implicantului: $\overline{AC}DE$.
  - Al doilea implicant 8,12,24,28 (4,16)
    - se scrie echivalentul mintermului 8: $\overline{A}B\overline{CDE}$.
    - se şterg variabilele cu ponderea 4 şi 16: $B\overline{DE}$.
  - A treilea implicant cu expresia 1,3,5,7,17,19,21,23 (2,4,16) va fi egal cu $\overline{B}E$.

Metoda Quine – McCluskey

- Exemplu
  - Rezultatul este: F = $\overline{ABC}DE + B\overline{DE} + \overline{B}E$.
- Metoda Quine-McCluskey se poate utiliza si cu termeni nedeterminati
  - In prima faza de cautare a implicantilor primi termenii nedeterminati sunt incluse in cautare (sunt considerate termeni obisnuiti)
  - In a doua faza de determinare a acoperiiri in matricea de incidenta lipsesc coloanele termenilor nedeterminati

Metoda Quine – McCluskey

- Metoda Quine-McCluskey cu termeni nedeterminati

$$f(A,B,C,D,E)=\sum m(\underbrace{5}_{2},\underbrace{7}_{3},\underbrace{11}_{3},\underbrace{12}_{2},\underbrace{27}_{4},\underbrace{29}_{4})+\sum d(\underbrace{14}_{3},\underbrace{20}_{2},\underbrace{21}_{3},\underbrace{22}_{3},\underbrace{23}_{4})$$

			5,7(2)
	5	√	5,21(16)
2 – 1s	12	√	12,14(2)
	20	√	20,21(1)
			20,22(2)
	7	√	7,23(16)
	11	√	11,27(16)
3 – 1s	14	√	21,23(2)
	21	√	21,29(8)
	22	√	22,23(1)
	23	√	
4 – 1s	27	√	X
	29	√	

Metoda Quine – McCluskey

- Metoda Quine-McCluskey cu termeni nedeterminati

2 – 1s	5 √ 12 √ 20 √	5,7(2) √ 5,21(16) √ $\boxed{12,14(2)}$ PI 20,21(1) √ 20,22(2) √	$\boxed{20,21,22,23(1,2)}$ PI $\boxed{5,7,21,23(2,16)}$ PI ~~20,22,21,23(2,1)~~ ~~5,21,7,23(16,2)~~
3 – 1s	7 √ 11 √ 14 √ 21 √ 22 √	7,23(16) √ $\boxed{11,27(16)}$ PI 21,23(2) √ $\boxed{21,29(8)}$ PI 22,23(1) √	
4 – 1s	23 √ 27 √ 29 √	X	

Metoda Quine – McCluskey

- Metoda Quine-McCluskey cu termeni nedeterminati

			√ 5	√ 7	√ 11	√ 12	√ 27	√ 29
Don't Cares Only	~~20,21,22,23~~	(1,2)						
EPI	5,7,21,23	(2,16)	$\boxed{X}$	$\boxed{X}$				
EPI	12,14	(2)				$\boxed{X}$		
EPI	11,27	(16)			$\boxed{X}$		$\boxed{X}$	
EPI	21,29	(8)						$\boxed{X}$

Metoda Quine – McCluskey

- Metoda Quine-McCluskey cu termeni nedeterminati

			16 A	8 B	4 C	2 D	1 E	Boolean
EPI	5,7,21,23	(2,16)	–	0	1	–	1	$\overline{B}CE$
EPI	12,14	(2)	0	1	1	–	0	$\overline{A}BC\overline{E}$
EPI	11,27	(16)	–	1	0	1	1	$B\overline{C}DE$
EPI	21,29	(8)	1	–	1	0	1	$AC\overline{D}E$

$$f(A,B,C,D,E) = \overline{B}CE + \overline{A}BC\overline{E} + B\overline{C}DE + AC\overline{D}E$$

Metoda Quine – McCluskey

- Utilizarea metodei in cazul minimizarii functiilor logice cu multiple iesiri
  - Se consideră funcţia

$$f_2 = m_1 + m_5 + m_6 + m_7;$$
$$f_1 = m_1 + m_4 + m_5 + m_6;$$
$$f_0 = m_0 + m_2 + m_5 + m_6 + m_7;$$

Metoda Quine – McCluskey

Generarea implicanţilor primi pentru această funcţie vectorială se face printr-un procedeu foarte puţin diferit în comparaţie cu cazul funcţiilor scalare.

Tabelul 4a

Iniţializarea metodei Quine-McCluskey, aplicată funcţiei vectoriale din Exemplul 7.

Implicanţii de ordinul 0			
Grupa	Indicii	$x_2x_1x_0$	$f_2f_1f_0$
0	0	000	001√
1	1	001	110√
	2	010	001√
	4	100	010√
2	5	101	111
	6	110	111
3	7	111	101√

Doi implicanţi ai unei funcţii vectoriale sunt adiacenţi dacă îndeplinesc condiţia de adiacenţă a implicanţilor funcţiilor scalare, aplicată părţilor de intrare, la care se adaugă condiţia de compatibilitate a părţii de ieşire.
Aceasta constă dintr-o intersecţie nevidă a respectivelor părţi de ieşire.

Astfel, spre exemplu, implicanţii de ordinul 0: (000 | 001) şi (001 | 110), au partea de intrare adiacentă dar funcţiile sunt disjuncte.
În concluzie, aceşti implicanţi de ordinul 0 nu sunt adiacenţi, aşa cum se poate remarca şi din tabelul 4a.

Metoda Quine – McCluskey

Tabelul 4b

Implicanţii de ordinul 1 ai funcţiei vectoriale din Exemplul 7.

Grupa	Indicii	$x_2x_1x_0$	$f_2f_1f_0$
0	0,2	0-0	001
	1,5	-01	110
	2,6	-10	001
1	4,5	10-	010
	4,6	1-0	010
2	5,7	1-1	101
	6,7	11-	101

Deîndată ce un implicant are un succesor de index superior, acesta este marcat (cu caracterul √).

Metoda Quine – McCluskey

Astfel, (010 | 001) şi (110 | 111) sunt adiacenţi dar este marcat doar primul

Tabelul 4c

Matricea de incidenţă a implicanţilor primi ai funcţiei vectoriale din exemplul 7.

Implicanţii primi			Termenii canonici												
			f_2				f_1				f_0				
Simbol	Binar	Indicii	1	5	6	7	1	4	5	6	0	2	5	6	7
p	(101 | 111)	5		*					*				*		
q	(110 | 111)	6			*					*				*	
r	(0-0 | 001)	0,2									*	*			
s	(-01 | 110)	1,5	*	*			*		*						
t	(-10 | 001)	2,6										*		*	
u	(10- | 010)	4,5						*	*						
v	(1-0 | 010)	4,6						*		*					
w	(1-1 | 101)	5,7		*		*							*		*
x	(11- | 101)	6,7			*	*								*	*

pentru că succesorul lor este în fapt doar succesorul primului, spre exemplu.

Deoarece, exceptând cei doi implicanţi din grupa 0, nici un alt implicant de ordin 0 nu are partea de ieşire complet reprezentată, rezultă că implicanţii grupei 0 nu au succesori si vor fi implicanţi primi, spre exemplu.

Metoda Quine – McCluskey

Implicanţii primi ai funcţiei ***f*** sunt:

(101 | 111), (110 | 111), (0-0 | 001), (-01 | 110), (-10 | 001), (10- | 010), (1-0 | 010), (1-1 | 101), (11- | 101).

Se ataşează fiecărui implicant prim din lista iniţială câte un simbol distinct:

p = (101 | 111), *q* = (110 | 111), *r* = (0-0 | 001), *s* = (-01 | 110), *t* = (-10 | 001), *u* = (10- | 010), *v* = (1-0 | 010), *w* = (1-1 | 101) şi *x* = (11- | 101).

Matricea de incidenţă a acestei funcţii vectoriale este prezentată în tabelul 4c.

Tabelul 4d

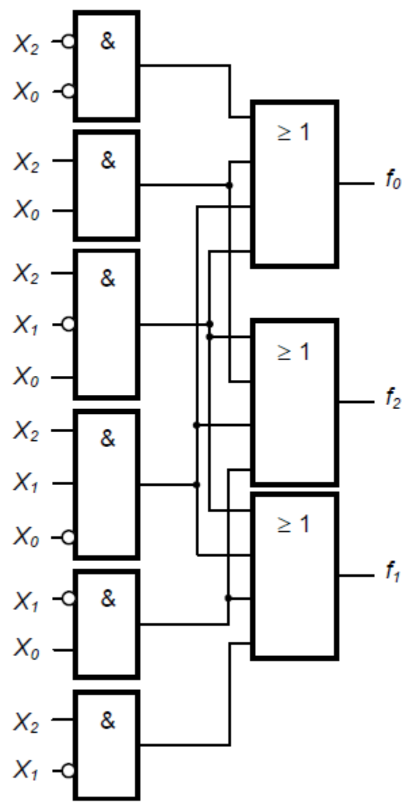
Matricea de incidenţă după îndepărtarea implicanţilor primi *p* şi *q* aparţinând celor trei funcţii f_2, f_1 şi f_0.

Implicanţii primi			Termenii canonici						
			f_2		f_1		f_0		
simbol	binar	indicii	1	7	1	4	0	2	7
r	(0-0 | 001)	0,2					*	*	
s	(-01 | 110)	1	*		*				
t	(-10 | 001)	2						*	
u	(10- | 010)	4				*			
v	(1-0 | 010)	4				*			
w	(1-1 | 101)	7		*					*
x	(11- | 101)	7		*					*

Metoda Quine – McCluskey

Raţionând după apartenenţa la funcţii, implicanţii primi p = (101 | 111), având indicele 5, şi q = (110 | 111), având indicele 6, aparţin intersecţiei celor trei funcţii $f_2 \cdot f_1 \cdot f_0$ şi vor avea ramificaţii spre cele trei linii de ieşire reprezentând f_2, f_1 şi f_0.
Cu alte cuvinte, implicanţii primi p şi q, aparţin oricărei soluţii de acoperire minimă primă care poate fi alcătuită prin setul actual de implicanţi primi.

Trebuie menţionat faptul că odată acoperiţi mintermii intersecţiei celor trei funcţii (m_5 şi m_6), coloanele acestora (5 şi 6) vor fi îndepărtate din matricea de incidenţă împreună cu liniile celor doi implicanţi primi.
Această transformare a matricei de incidenţă este ilustrată în tabelul 4d.



Metoda Quine – McCluskey

Considerând matricea de incidenţă rezultată după îndepărtarea implicanţilor primi *p* şi *q*, aparţinând celor trei funcţii f_2, f_1 şi f_0, se remarcă faptul că implicantul prim *s* aparţine intersecţiei funcţiilor f_2 şi f_1.

Acest implicant prim acoperă termenul canonic m_1, iar acest termen canonic este chiar intersecţia celor două funcţii, $m_1 = f_2 \cdot f_1$.

Implicantul prim *s* va avea ramificaţii atât spre linia de ieşire a funcţiei f_2, cât şi spre linia de ieşire a funcţiei f_1. În această manieră se dovedeşte că implicantul prim *s* va aparţine oricărei acoperiri minime prime a acestei funcţii vectoriale.

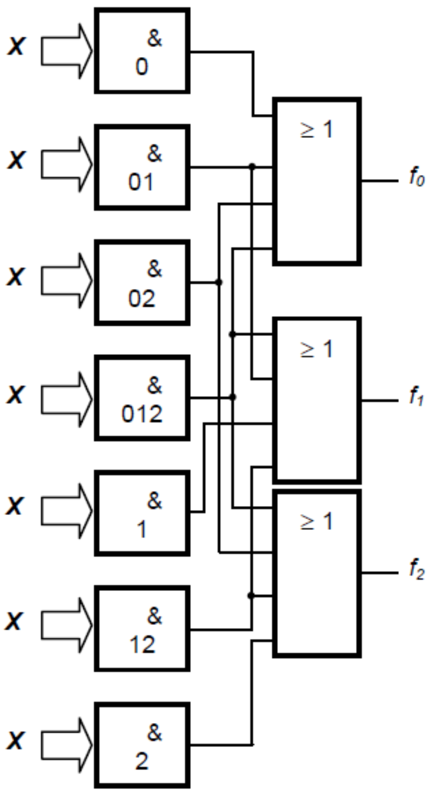
Tabelul 4e

Matricea de incidenţă după îndepărtarea implicantului prim *s* şi a coloanelor m_1.

Implicanţii primi			Termenii canonici				
			f_2	f_1		f_0	
simbol	binar	indicii	7	4	0	2	7
r	(0-0 | 001)	0,2			*	*	
t	(-10 | 001)	2				*	
u	(10- | 010)	4		*			
v	(1-0 | 010)	4		*			
w	(1-1 | 101)	7	*				*
x	(11- | 101)	7	*				*

Metoda Quine – McCluskey

Tabelul 4e prezintă matricea de incidenţă după excluderea implicantului prim *s* şi a coloanelor corespunzătoare termenului canonic m_1, atât din dreptul funcţiei f_2 cât şi din dreptul funcţiei f_1.
Matricea de incidenţă din tabelul 4e a scăzut considerabil în complexitate.
Astfel, termenul canonic m_7 aparţinând funcţiei f_2 şi funcţiei f_0 poate fi acoperit de implicantul prim *w* sau de implicantul prim *x*, unul dintre aceştia.
În mod analog, termenul canonic m_4, aparţinând funcţiei f_1, poate fi acoperit de implicantul prim *u* sau de implicantul prim *v*, oricare dintre aceştia.
Implicantul prim *r* domină implicantul prim *t*, în consecinţă acesta din urmă poate fi îndepărtat din matrice.
Acoperirea primă minimă a funcţiei vectoriale arată astfel:
(101|111), (110|111), (-01|110), (1-1|101), (10-|010), (0-0|001).
Implementarea prin circuite logice ŞI şi SAU a descrierii simbolice corespunzătoare acoperirii prime minime a acestei funcţii vectoriale este prezentată în figura 5.
◊



Metoda Quine – McCluskey

În cazul unei funcţii vectoriale, având trei componente scalare, de forma:

$$\boldsymbol{F}: \boldsymbol{B}^n \to \boldsymbol{B}^3,$$

structura minimizată generică a funcţiei $\boldsymbol{F}$ arată ca în figura 6.

Toate porţile *ŞI* sunt generice şi sunt indexate.

Indexul unei porţi *ŞI* este înscris, în figura 6, la baza simbolului porţii respective. Circuitele *ŞI* sunt generice în sensul că pot fi mai multe circuite având acelaşi index dar realizând fiecare, în parte, alt termen produs. Doi termeni produs, având acelaşi index, pot diferi atât prin paritatea variabilelor cât şi prin faptul că liniile lor de intrare pot defini două submulţimi distincte ale domeniului de definiţie.

Codul indexului unui circuit *ŞI* este determinat în raport cu ramificaţiile pe care le are linia de ieşire a respectivului circuit. Astfel, un circuit *ŞI* având indexul 012 va avea linia de ieşire ramificată spre circuitele *SAU* aparţinând liniilor funcţiilor scalare f_0, f_1 şi f_2.

Metoda Quine – McCluskey

În mod similar, o poartă *ŞI* având indexul 12 îşi ramifică liniile de ieşire spre porţile *SAU* ale funcţiilor scalare f_1 şi f_2. În timp ce o poartă ŞI cu indexul 0 se va conecta doar la poarta SAU a cărei linie de ieşire este etichetată prin f_0, spre exemplu.
Porţile indexate printr-o singură cifră (0, 1 sau 2, în cazul acesta) nu au linia de ieşire ramificată dar sunt generice. Aceste porţi se conectează, fiecare, doar intrarea unei porţi *SAU* a unei singure funcţii scalare (respectiv f_0, f_1 sau f_2).
Analog, sunt definibile structurile generice pentru funcţiile multi-ieşire având patru ori mai multe linii scalare de ieşire.

Minimizare euristică

- În cazul circuitelor reale, mărimea de ordin a numărul variabilelor de intrare a unei funcţii booleene poate să ajungă la un ordin de 1000 sau și mai mare.
- Minimizarea euristică nu caută o acoperire minimă, ci o acoperire destul de bună.
- Minimizarea funcţiilor se face iterativ, prin aplicarea operatorilor de expandare, neredundanţă şi reducere

Minimizare euristică

- Exemplu
- Se proneşte de la o acoperire oarecare, poate fi şi definiţia funcţiei

	00	01	11	10		abcd	F	implicant
00	1	1				0-0-	1	a
01	1	1	1	1		0-10	1	b
11			1	1		1001	1	c
10	1	1	1	1		1101	1	d
						1-1-	1	e

Minimizare euristică

- Exemplu
  - Expandare, acest operator calculează o acoperire primă şi minimală în raport cu conţinerea într-un singur implicant. Implicanţii care formează acoperirea sunt procesaţi rând pe rând. Fiecare implicant care nu este prim este expandat într- un implicant prim, adică este înlocuit printr-un implicant prim care-l conţine. Consecutiv, toţi implicanţii din acoperire (cei încă ne-procesaţi) care sunt conţinuţi în acest implicant prim sunt îndepărtaţi.

Minimizare euristică

- Exemplu
- După expandare

	00	01	11	10		abcd	F	implicant
00	1	1				0-0-	1	a
01	1	1	1	1		--10	1	b
11			1	1		--01	1	c
10	1	1	1	1		1-0-	1	d
						1-1-	1	e

Minimizare euristică

- Minimizare euristică
  - Exemplu
    - iredundant, are drept acţiune calculul unei acoperiri iredundante. Operatorul alege un subset al implicanţilor astfel încât nici un implicant al acestui subset nu este acoperit de ceilalţi implicanţi din subset.

Minimizare euristică

- Exemplu
- După iredundant

	00	01	11	10		abcd	F	implicant
00	1	1				0-0-	1	a
01	1	1	1	1		--10	1	b
11			1	1		~~--01~~	~~1~~	~~c~~
10	1	1	1	1		1-0-	1	d
						1-1-	1	e

Minimizare euristică

- Minimizare euristică
  - Exemplu
    - Reduce, este un operator care realizează transformarea unei acoperiri oarecare, într-o acoperire ne-primă dar de aceeaşi cardinalitate cu cea iniţială. Implicanţii sunt procesaţi unul câte unul. Acest operator încearcă să înlocuiască fiecare implicant printr-un altul care este conţinut în acesta, cu condiţia că implicanţii reduşi împreună cu cei rămaşi continuă să acopere funcţia.

Minimizare euristică

- Exemplu
- După reducere

	00	01	11	10		abcd	F	implicant
00	1	1				0-0-	1	a
01	1	1	1	1		0-10	1	b
11			1	1		~~--01~~	~~1~~	~~c~~
10	1	1	1	1		1-01	1	d
						1-1-	1	e

Minimizare euristică

- Exemplu
- Încă o operaţie de expandare

	00	01	11	10		abcd	F	implicant
00	1	1				0-0-	1	a
01	1	1	1	1		--10	1	b
11			1	1		~~--01~~	~~1~~	~~c~~
10	1	1	1	1		1--1	1	d
						1-1-	1	e

Minimizare euristică

- Exemplu
- Încă o operaţie de iredundant

	00	01	11	10		abcd	F	implicant
00	1	1				0-0-	1	a
01	1	1	1	1		--10	1	b
11			1	1		~~--01~~	~~1~~	~~c~~
10	1	1	1	1		1--1	1	d
						~~1-1-~~	~~1~~	~~e~~

Unelte pentru minimizarea funțiilor

Espresso

- Programul original Espresso este disponibil ca cod sursă C de la site-ul web al Universității din California, Berkeley.
- Algoritmul espresso consta in 3 pasi de baza:
  - Expandare
  - Iredundant
  - Reducere

Espresso

- Exemplul 1
  - Explicatia fisierului:
    - # comentariu
    - .i 4 # (variabile de intrare)
    - .o 1 # (varuabile de iesire)
    - .ilb A B C D # (Numele variabilelor de intrare)
    - .ob F # (Numele variabilei de iesire)
    - 0 0 0 0   0 # (tabelul de adevar, 4 intrari, o iesire
    - .e # (Marcheaza sfarsitul fisierului)

```
# example inp
.i 4
.o 1
.ilb A B C D
.ob F
0 0 0 0   0
0 0 0 1   0
0 0 1 0   0
0 0 1 1   0
0 1 0 0   0
0 1 0 1   0
0 1 1 0   1
0 1 1 1   1
1 0 0 0   1
1 0 0 1   1
1 0 1 0   0
1 0 1 1   0
1 1 0 0   0
1 1 0 1   0
1 1 1 0   0
1 1 1 1   0
.e
```

Espresso

- Exemplul 1
  - Rezultat
    - .p 2 (Indica faptul ca sunt doi termini in expresia de iesire)
    - 100- 1 (termenul este AB'C'. notatie: B' este B negat,
    - 011- 1 (termenul este A'BC)
    - Expresia logica este F = AB'C' + A'BC.

```
C:\espressso>espresso ex1.txt
# example input file 1 ex1.txt
.i 4
.o 1
.ilb A B C D
.ob F
.p 2
100- 1
011- 1
.e

C:\espressso>
```

Espresso

- Exemplul 2

```
# example file 2

.i 3
.o 3
.ilb A B C
.ob F G H
000 100
1-- 101
1-1 0-1
010 010
.e
```

- Rezultat

```
C:\espressso>espresso ex2.txt
# example file 2
.i 3
.o 3
.ilb A B C
.ob F G H
.p 3
010 010
-00 100
1-- 101
.e

C:\espressso>
```

Espresso

- Exemplul 3
  - f(a,b,c,d) = {4 ,5 ,6 ,8 9 ,0 ,13} , având următorii termeni canonici neprecizaţi: {0, 7,15}.

```
.i 4
.o 1
.ilb a b c d
.ob f
.p 10
0100 1
0101 1
0110 1
1000 1
1001 1
1010 1
1101 1
0000 -
0111 -
1111 -
.e
```

- Rezultat

```
C:\espressso>espresso ex3.txt
.i 4
.o 1
.ilb a b c d
.ob f
.p 3
1-01 1
10-0 1
01-- 1
.e

C:\espressso>
```

- f(a,b,c,d) = ac'd + ab'd' + a'b.

Espresso

- Exemplul 4

```
.i 4
.o 7
.ilb x8 x4 x2 x1
.ob a b c d e f g
.p 16
0000 1111110
0001 0110000
0010 1101101
0011 1111001
0100 0110011
0101 1011011
0110 1011111
0111 1110000
1000 1111111
1001 1110011
1010 - - - - - - -
1011 - - - - - - -
1100 - - - - - - -
1101 - - - - - - -
1110 - - - - - - -
1111 - - - - - - -
.e
```

- Rezultat

```
C:\espressso>espresso ex4.txt
.i 4
.o 7
.ilb x8 x4 x2 x1
.ob a b c d e f g
.p 9
-10- 0000001
-01- 0001001
-0-1 0110000
-101 1011010
--00 0110010
--11 1110000
-0-0 1101100
1--- 1000011
-110 1011111
.e

C:\espresso>
```

a = x4x2’x1 + x2x1 + x4’x1’ + x8 + x4x2x1’

b = x4’x1 + x2’x1’ + x2x1 + x4’x1’

c = x4’x1 + x4x2’x1 + x2’x1’ + x2x1 + x4x2x1’

d = x4’x2 + x4x2’x1 + x4’x1’ + x4x2x1’

e = x4’x1’ + x4x2x1’

f = x4x2’x1 + x2’x1’ + x8 + x4x2x1’

g = x4x2’ + x4’x2 + x8 + + x4x2x1’