Capitolo 4

Algebra Booleana e porte logiche

4.1 Calcolo funzionale di verità

4.1.1 Introduzione

Come anticipato brevemente nel capitolo 1, la logica Booleana e le corrispondenti regole di manipolazione dei suoi elementi definite nell'*Algebra Boolena*, sono lo strumento concettuale chiave per lo sviluppo di *tutta* la tecnologia microelettronica che ha portato ai computer moderni, alle loro reti, a Internet, alla televisione digitale, alla telefonia mobile e, in generale, a tutti quei dispositivi elettronici di elaborazione dei segnali finalizzata al monitoraggio, controllo e gestione, locale e a distanza, degli strumenti automatici che usiamo quotidianamente. Quando usiamo il computer, lo smartphone, il telecomando per il cancello elettrico, quando attiviamo l'allarme di casa, usiamo un lettore ottico al supermercato, paghiamo col bancomat o freniamo in emergenza guidando l'automobile, stiamo usando l'Algebra Booleana.
Fatte queste premesse, è evidente che essa è la colonna portante di tutto lo sviluppo tecnologico della nostra società dell'informazione e che sia assolutamente necessario comprendere in profondità il suo funzionamento.
La Logica Booleana nasce nel 1854 a opera del matematico inglese George Boole, ma rimane inizialmente confinata nell'ambito degli specialisti di matematica. Solo quasi cent'anni dopo viene sfruttata a livello applicativo, quando nel 1938 Claude Elwood Shannon ne adattò il simbolismo all'analisi dei circuiti di commutazione, uno strumento essenziale nel progetto dei circuiti logici, che sono la base costitutiva dei moderni microprocessori e di tutta la circuiteria a essi asservita. L'algebra Booleana, nella sua declinazione binaria, contempla solo due valori logici atti a rappresentare il *vero* e il *falso* in una proposizione e ci permette di studiare formalmente i problemi della *logica deduttiva*. Successivamente, con Shannon, tali valori logici vengono associati allo stato di apertura o di chiusura di un generico contatto, oppure alla presenza o all'assenza di un segnale in un particolare punto di un circuito. I due valori logici, che sono mutuamente escludentisi, sono chiamati *costanti logiche* o *costanti binarie* e sono indicati di solito con i simboli 0 e 1. Prima di addentrarci nello studio sistematico della Logica e dell'Algebra Booleana, facciamo un esempio introduttivo che ci fa capire fin da subito l'importante valore operativo di questa disciplina.

Problema Si voglia progettare un sistema di allarme per le cinture di sicurezza di un autoveicolo. Le specifiche di progetto sono:

1. si dispone della seguente serie di sensori atti a misurare le condizioni del sistema:
   - un sensore che attesta l'accensione del motore;
   - un sensore, piazzato sul cambio, per accorgersi se esso sia in folle o meno;

- un interruttore posto sotto ciascun sedile anteriore;
- un interruttore connesso con ciascuna cintura, che segnala se essa è allacciata o meno.

2. *Un cicalino deve suonare quando l'accensione è attivata, il cambio è innestato e almeno uno dei due sedili anteriori è occupato senza che la relativa cintura sia allacciata*

Anche se in questo caso il problema è sufficientemente semplice per poter tentare una risoluzione usando metodi di progetto semiempirici, è tuttavia necessario procedere a un'analisi formale delle specifiche assegnate, poiché essa è l'unica praticabile quando si ha a che fare con sistemi più complessi.
Le grandezze delle specifiche di progetto possono essere listate come segue:

suono del cicalino	C
accensione attivata	A
cambio innestato	G
sedile anteriore sinistro occupato	S_L
sedile anteriore destro occupato	S_R
cintura sinistra allacciata	B_L
cintura destra allacciata	B_R

A ciascuna grandezza viene associata una variabile, rappresentativa della grandezza stessa, che assume un valore di verità V o F (*Vero* o *Falso*) a seconda che la condizione espressa sia *vera* o *falsa*; chiamiamo tale variabile *variabile Booleana*. Ciò si può generalizzare a qualsiasi dichiarazione che possa essere classificata come vera o falsa; in tal caso la dichiarazione potrebbe essere costruita con l'apporto di più variabili Booleane opportunamente combinate.
Il metodo che permette di manipolare queste variabili e di assegnare a esse un valore di verità è conosciuto come *calcolo funzionale di verità* e non si limita alla manipolazione di grandezze semplici come quelle appena introdotte.
Si può innanzitutto osservare che per ogni affermazione assertiva esiste una corrispondente formulazione negativa, che viene espressa mediante una barra orizzontale sopra la variabile corrispondente. Ad esempio:

$$\textit{La cintura sinistra non è allacciata} \quad \rightarrow \quad \overline{B_L}$$

Poiché $\overline{B_L}$ è vera quando B_L è falsa e viceversa, essa viene chiamata la *negazione* di B_L, e viene indicata anche con NOT(B_L). Spesso la negazione di una variabile A viene indicata simbolicamente anche con $\sim A$ o con A'.
Si consideri ora la proposizione:

$$\textit{La cintura di sinistra non è allacciata e il sedile anteriore sinistro è occupato} \quad \rightarrow \quad \overline{B_L} \cap S_L$$

Quest'ultimo è un esempio di *funzione composta di verità* o *affermazione composta*, il cui valore di verità può essere determinato a partire dalle proposizioni componenti. Le esatte relazioni tra i valori di verità delle affermazioni componenti e il valore di verità dell'affermazione composta dipende dalla connessione (o connettivo) esistente tra le parti componenti. Nel caso in esame il connettivo e l'AND e indica che la proposizione $\overline{B_L} \cap S_L$ è vera *se e solo se* sono vere entrambe le proposizioni elementari $\overline{B_L}$ e S_L. L'AND è una relazione molto comune tra le proposizioni e viene rappresentata con il simbolo $\cap$.
Vi sono evidentemente solo quattro possibili combinazioni di valori di verità di due proposizioni A e B. Si può pertanto definire completamente la proposizione composta $A \cap B$ riportando i suoi valori di verità in una tabella di quattro righe, come illustrato nella terza colonna della tabella di figura 4.1; tale tabella prende il nome di *tavola di verità* (del connettivo AND, in questo caso).

		AND	OR	XOR	XNOR	NOR	NAND
A	B	$A \cap B$	$A \cup B$	$A \oplus B$	$A \equiv B$	$A \downarrow B$	$A \mid B$
F	F	F	F	F	V	V	V
F	V	F	V	V	F	F	V
V	F	F	V	V	F	F	V
V	V	V	V	F	V	F	F

Figura 4.1: Tavola di verità dei principali connettivi

La tabella definisce anche il connettivo $A \cup B$ (quarta colonna in figura 4.1) che simbolizza una proposizione che è vera quando l'una o l'altra o entrambe le proposizioni elementari sono vere. Un esempio della vita reale potrebbe essere rappresentato dalla frase

Se hai fame o sete ti offro qualcosa al bar

Il relativo connettivo viene chiamato OR; è evidente che in questo caso si va al bar se l'amico è assetato, affamato o entrambi (a maggior ragione). Ma l'uso comune della congiunzione "o" non è sempre in accordo col significato della proposizione $A \cup B$. Si consideri infatti l'affermazione:

Giorgio usa vestiti grigi o azzurri

E' evidente che tale affermazione non significa che Giorgio usi contemporaneamente vestiti grigi e azzurri. Pertanto, nel caso che si sta esaminando il connettivo prende il nome di OR ESCLUSIVO (o XOR, che è la contrazione di *eXclusive OR*) e viene rappresentato con $A \oplus B$. La relativa tavola di verità è riportata nella quinta colonna di figura 4.1. Per sottolineare la differenza con quest'ultimo caso, il connettivo $\cup$ viene per contrasto chiamato OR INCLUSIVO, ma molto spesso, in relazione al suo frequentissimo uso, lo si chiama semplicemente OR.
Prima di passare al calcolo funzionale di verità del problema che si sta esaminando, definiamo anche altri connettivi di uso frequente. Il primo è $A \equiv B$; esso sta a indicare che A e B hanno sempre lo stesso valore di verità, cioè A è vero *se e solamente se* B è vero. Poiché esso è la negazione dello XOR, viene anche chiamato XNOR. La relativa tavola di verità è rappresentata in sesta colonna della figura 4.1.
Gli ultimi due connettivi che introduciamo sono il connettivo NOR e il connettivo NAND, i cui simboli sono rispettivamente $\downarrow$ e $\mid$; essi rappresentano la negazione di OR e di AND. I valori di verità occupano la settima e l'ottava colonna della tabella di figura 4.1.

Si possono ora rappresentare le specifiche di progetto del sistema in esame con un'equazione funzionale di verità. E' tuttavia opportuno manipolare tali specifiche in termini di proposizioni semplici in modo da eliminare le ambiguità lessicali:

Specifiche - L'allarme suonerà (C) se e solo se valgono contemporaneamente queste tre condizioni:

1. l'accensione è attivata (A)
2. il cambio è innestato (G)
3. - il sedile anteriore sinistro è occupato (S_L) e la cintura sinistra non è allacciata ($\overline{B_L}$) oppure
   - il sedile anteriore destro è occupata (S_R) e la cintura destra non è allacciata ($\overline{B_R}$)

A partire da tali specifiche si può scrivere facilmente l'equazione funzionale di verità. Si ottiene:

$$C \equiv A \cap G \cap [(S_L \cap \overline{B_L}) \cup (S_R \cap \overline{B_R})] \tag{4.1}$$

Si osservi che la rigorosa struttura gerarchica data alle specifiche è utile a evitare ambiguità nella precisa dichiarazione delle stesse, in particolare nei connettivi logici che legano le varie dichiarazioni. E' infatti molto facile incappare in tali ambiguità quando si descrivono specifiche mediante proposizioni dichiarative, usando però un linguaggio colloquiale. Un esempio in tal senso potrebbe essere la frase

L'allarme suona quando la cintura destra non è allacciata

che è una frase di senso comune, ben comprensibile, ma che non rappresenta in modo adeguato la complessità e la struttura delle specifiche. Questa infatti non è una funzione composta di verità, poiché il suo valore di verità non può essere determinato unicamente a partire a partire dal valore di verità delle singole proposizioni componenti. Infatti, anche se ambedue le componenti sono vere, la proposizione composta può non essere vera in quanto il sedile destro potrebbe non essere occupato. In tal caso la parte destra dell'affermazione composta è vera, ma l'allarme potrebbe essere stato attivato dal conducente che non ha allacciato la cintura di sicurezza. Un'affermazione del tipo appena visto non può evidentemente essere manipolata tramite il calcolo funzionale di verità.

4.1.2 I connettivi binari

Nella tabella di figura 4.1 sono stati individuati sei connettivi logici, AND, OR, XOR e XNOR, NOR e NAND, che sono i più interessanti. Poiché per ciascuna coppia di valori delle variabili ci sono due possibilità, abbiamo 4 righe, e quindi $2^4 = 16$ connettivi logici. Vediamoli tutti:

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
			AND					XOR	OR	NOR	XNOR					NAND	
A	B		$\cap$					$\oplus$	$\cup$	$\downarrow$	$\equiv$		$\subset$		$\supset$	$\mid$	
F	F	F	F	F	F	F	F	F	F	V	V	V	V	V	V	V	V
F	V	F	F	F	F	V	V	V	V	F	F	F	F	V	V	V	V
V	F	F	F	V	V	F	F	V	V	F	F	V	V	F	F	V	V
V	V	F	V	F	V	F	V	F	V	F	V	F	V	F	V	F	V

Figura 4.2: Tavola di verità dei $2^4 = 16$ connettivi binari

Tra questi vari connettivi, veste particolare interesse quello corrispondente al numero 13, che si indica con

$$A \supset B$$

che corrisponde alla proposizione composta

se A è vero, allora B è vero if A then B

ed è chiamato *implicazione*. Questo connettivo è simile allo XNOR "*se e solamente se*" introdotto al paragrafo precedente (connettivo 9), eccetto per il fatto che B non è necessariamente falso se A è falso.
Si consideri ad esempio la seguente proposizione:

se piove allora ci sono nuvole in cielo

l'unica possibilità che tale affermazione sia falsa è quella di verificare che in un dato momento piova ma *non* ci siano nuvole in cielo. Supponendo che sia vera $A \supset B$, questa può anche essere espressa nei seguenti modi:

A è condizione sufficiente per B
B è condizione necessaria per A

Applicando tali modi all'esempio precedente rispetto al linguaggio comune possiamo affermare che condizione sufficiente perché in cielo ci siano nuvole è che piova; oppure che condizione necessaria perché piova è che in cielo ci siano nuvole.
Vediamo ora un altro esempio:

$$(2 > x > 1) \supset (x > 0) \tag{4.2}$$

Se $x = 0.5$ la proposizione $x > 0$ è certamente vera, mentre la proposizione $2 > x > 1$ è falsa. Non vi è tuttavia contraddizione con la proposizione 4.2 nel suo complesso, che è vera come è stato definito nella tabella di figura 4.2. Il valore di verità dell'implicazione (che spesso è anche detta if - then) può a prima vista apparire non molto naturale, ma questo concetto è necessario in molti argomenti matematici. E' tuttavia bene puntualizzare che la distinzione tra il connettivo *implicazione* e quello "*se e solamente se*" è estremamente importante. Infatti la validità di quest'ultimo è subordinata alla contemporanea validità dell'implicazione nei due sensi

$$A \equiv B \iff A \supset B \text{ e } A \subset B$$

cioè (4.3)

$$(A \equiv B) \equiv (A \supset B) \cap (A \subset B)$$

La dimostrazione è immediata e si può ottenere manipolando e confrontando le rispettive tavole di verità.

A	B	$A \supset B$	$A \subset B$	$(A \supset B) \cap (A \subset B)$	$(A \equiv B)$
F	F	V	V	V	V
F	V	V	F	F	F
V	F	F	V	F	F
V	V	V	V	V	V

4.1.3 Insiemi minimi di connettivi

La relazione (4.3) ci mostra che alcuni connettivi binari possono essere espressi sulla base di altri connettivi; in questo caso $\equiv$ viene espresso in funzione di $\supset, \subset$ e $\cap$. Si presenta quindi spontanea la domanda di quali siano i connettivi idonei a esprimere tutti gli altri, e quale sia l'insieme minimo tra essi che permette di esprimere tutte le 16 combinazioni di figura 4.2. Si consideri ad esempio l'insieme dei connettivi che contiene solo $\cup$ e $\cap$, cioè OR e AND. I casi in cui sia A che B sono falsi è chiaramente critico; infatti dalla tabella di figura 4.1 vediamo che sia $A \cup B$ e $A \cap B$ sono ambedue falsi; quindi, sulla base di questi due soli connettivi, non è possibile esprimere connettivi che assumono valore vero quando A e B sono falsi. Per poter esprimere anche questi casi, è necessario arricchire l'insieme introducendo anche il connettivo NOT. Ecco allora che l'insieme dei connettivi AND, OR, NOT è un insieme sufficiente per ricostruire tutti gli altri connettivi, come si può verificare dalla tabella di verità di figura 4.3.

$F\,F\,V\,V$ $F\,V\,F\,V$	A B		$F\,F\,V\,V$ $F\,V\,F\,V$	A B	
$F\,F\,F\,F$	$f_0 \equiv A \cap \overline{A}$		$V\,F\,F\,F$	$f_8 \equiv \overline{A \cup B}$	NOR
$F\,F\,F\,V$	$f_1 \equiv A \cap B$	AND	$V\,F\,F\,V$	$f_9 \equiv (A \cup \overline{B}) \cap (\overline{A} \cup B)$	XNOR
$F\,F\,V\,F$	$f_2 \equiv A \cap \overline{B}$			$\equiv (\overline{A} \cap \overline{B}) \cup (A \cap B)$	
$F\,F\,V\,V$	$f_3 \equiv A$		$V\,F\,V\,F$	$f_{10} \equiv \overline{B}$	
$F\,V\,F\,F$	$f_4 \equiv \overline{A} \cap B$		$V\,F\,V\,V$	$f_{11} \equiv A \cup \overline{B}$	$\subset$
$F\,V\,F\,V$	$f_5 \equiv B$		$V\,V\,F\,F$	$f_{12} \equiv \overline{A}$	
$F\,V\,V\,F$	$f_6 \equiv (A \cap \overline{B}) \cup (\overline{A} \cap B)$	XOR	$V\,V\,F\,V$	$f_{13} \equiv \overline{A} \cup B$	$\supset$
	$\equiv (\overline{A} \cup \overline{B}) \cap (A \cup B)$		$V\,V\,V\,F$	$f_{14} \equiv \overline{A \cap B}$	NAND
$F\,V\,V\,V$	$f_7 \equiv A \cup B$	OR	$V\,V\,V\,V$	$f_{15} \equiv A \cup \overline{A}$	

Figura 4.3: I 16 connettivi binari della tabella di figura 4.2 espressi mediante AND, OR, NOT

Per costruire la tabella basta partire da A e B e costruire in modo sistematico tutte le unioni, intersezioni e negazioni tra i vari elementi. P.es. da $A = FFVV$ ricavo $\overline{A} = VVFF$, da $B = FVFV$ ricavo $\overline{B} = VFVF$, e da questi $A \cap B = FFFV$, $A \cap \overline{B} = FFVF$, $A \cup B = VVVF$ ecc. Così facendo si riempie quasi interamente la tabella, cioè quasi tutti i connettivi $f_0 ... f_{15}$ possono essere costruiti usando solo il connettivo $\cup$ oppure solo il connettivo $\cap$ (con l'eventuale negazione). L'unica eccezione è costituita dai connettivi f_6 e f_9, che necessitano l'impiego contemporaneo di $\cup$ e $\cap$. L'espressione risolutiva per f_9 si può ricavare tenendo conto dell'espressione (4.3) e del fatto che $A \supset B = \overline{A} \cup B$ e $A \subset B = A \cup \overline{B}$, e dunque $f_9 \equiv (A \cup \overline{B}) \cap (\overline{A} \cup B)$, mentre la f_6 è la sua negazione.
E' interessante notare che ci sono due modi per realizzare f_6 e f_9, combinando assieme $\cup$ e $\cap$, ma a ben guardare un po' tutti i connettivi possono essere espressi in modi alternativi; p.es. $f_0 \equiv A \cap \overline{A}$, ma anche $f_0 \equiv \overline{A \cup \overline{A}}$, e così via. Tutto ciò deriva dalla circostanza che l'insieme dei connettivi AND, OR, NOT non è il minimo; il connettivo AND può infatti essere espresso in funzione del connettivo OR (e viceversa); a tal riguardo vale l'importante teorema di De Morgan

Teorema 4.1 (De Morgan)**.**

$$\overline{A \cup B} \equiv \overline{A} \cap \overline{B} \qquad \overline{A \cap B} \equiv \overline{A} \cup \overline{B} \tag{4.4}$$

La dimostrazione procede per induzione perfetta, cioè effettuando la verifica direttamente sulle tavole di verità

A	B	$\overline{A}$	$\overline{B}$	$\overline{A} \cap \overline{B}$	$A \cup B$	$\overline{A \cup B}$
F	F	V	V	V	F	V
F	V	V	F	F	V	F
V	F	F	V	F	V	F
V	V	F	F	F	V	F

A	B	$\overline{A}$	$\overline{B}$	$\overline{A} \cup \overline{B}$	$A \cap B$	$\overline{A \cap B}$
F	F	V	V	V	F	V
F	V	V	F	V	F	V
V	F	F	V	V	F	V
V	V	F	F	F	V	F

(4.5)

Il teorema 4.1 si può generalizzare al caso di più variabili; accade infatti che la negazione di un connettivo di più variabili si ottiene negando ogni variabile e scambiando tra di loro i due operatori $\cup$ e $\cap$. In termini formali si ha

$$\overline{F}(x_1, x_2, \cdots, x_n)[\cup, \cap] = F(\overline{x_1}, \overline{x_2}, \cdots, \overline{x_n})[\cap, \cup] \tag{4.6}$$

I connettivi che legano le variabili non negate si esprimono allora come

$$A \cap B \equiv \overline{\overline{A} \cup \overline{B}} \qquad A \cup B \equiv \overline{\overline{A} \cap \overline{B}} \tag{4.7}$$

Poiché l'AND può essere espresso in funzione di OR e NOT, quest'ultimo insieme è da solo sufficiente per coprire tutti i connettivi di tabella 4.3, che sono riportati nella prima colonna della tabella 4.4. Per farlo bisogna sostituire

tutti i simboli $\cap$ con la prima espressione della (4.7); così facendo si ottiene la seconda colonna della tabella 4.4. Lo stesso ragionamento può essere fatto per l'OR, che può essere espresso in funzione di AND e NOT; per farlo bisogna sostituire tutti i simboli $\cup$ con la seconda espressione della (4.7); così facendo si ottiene la terza colonna della tabella 4.4.

	AND, OR, NOT	OR, NOT	AND, NOT
	$f_0 \equiv A \cap \overline{A}$	$\overline{\overline{A} \cup A}$	$A \cap \overline{A}$
AND	$f_1 \equiv A \cap B$	$\overline{\overline{A} \cup \overline{B}}$	$A \cap B$
	$f_2 \equiv A \cap \overline{B}$	$\overline{\overline{A} \cup B}$	$A \cap \overline{B}$
	$f_3 \equiv A$	A	A
	$f_4 \equiv \overline{A} \cap B$	$\overline{A \cup \overline{B}}$	$\overline{A} \cap B$
	$f_5 \equiv B$	B	B
XOR	$f_6 \equiv (A \cap \overline{B}) \cup (\overline{A} \cap B)$	$\overline{(\overline{A} \cup B)} \cup \overline{(A \cup \overline{B})}$	$\overline{\overline{(A \cap \overline{B})} \cap \overline{(\overline{A} \cap B)}}$
OR	$f_7 \equiv A \cup B$	$A \cup B$	$\overline{\overline{A} \cap \overline{B}}$
NOR	$f_8 \equiv \overline{A \cup B}$	$\overline{A \cup B}$	$\overline{A} \cap \overline{B}$
XNOR	$f_9 \equiv (A \cup \overline{B}) \cap (\overline{A} \cup B)$	$\overline{\overline{(A \cup \overline{B})} \cup \overline{(\overline{A} \cup B)}}$	$\overline{(\overline{A} \cap B)} \cap \overline{(A \cap \overline{B})}$
	$f_{10} \equiv \overline{B}$	$\overline{B}$	$\overline{B}$
$\subset$	$f_{11} \equiv A \cup \overline{B}$	$A \cup \overline{B}$	$\overline{\overline{A} \cap B}$
	$f_{12} \equiv \overline{A}$	$\overline{A}$	$\overline{A}$
$\supset$	$f_{13} \equiv \overline{A} \cup B$	$\overline{A} \cup B$	$\overline{A \cap \overline{B}}$
NAND	$f_{14} \equiv \overline{A \cap B}$	$\overline{A} \cup \overline{B}$	$\overline{A \cap B}$
	$f_{15} \equiv A \cup \overline{A}$	$A \cup \overline{A}$	$\overline{\overline{A} \cap A}$

Figura 4.4: I 16 connettivi binari espressi rispettivamente mediante AND, OR, NOT, mediante OR, NOT e mediante AND, NOT

Osserviamo ora che l'insieme minimo che consente di rappresentare tutti i connettivi può essere ulteriormente ridotto al solo connettivo NOR, oppure al solo connettivo NAND; per farlo è sufficiente esprimere AND, OR e NOT in funzione del solo NOR oppure del solo NAND. Per questo motivo i connettivi NOR e NAND sono chiamati *universali*. Ecco come procedere:

$$
\begin{aligned}
\overline{A} &\equiv \overline{A \cup A} \equiv A \downarrow A & \overline{A} &\equiv \overline{A \cap A} \equiv A|A \\
A \cup B &\equiv \overline{\overline{A \cup B}} \equiv \overline{A \downarrow B} & A \cup B &\equiv \overline{\overline{A \cup B}} \equiv \overline{\overline{A} \cap \overline{B}} \equiv \overline{A}|\overline{B} \\
&\equiv (A \downarrow B) \downarrow (A \downarrow B) & &\equiv (A|A)|(B|B) \\
A \cap B &\equiv \overline{\overline{A \cap B}} \equiv \overline{\overline{A} \cup \overline{B}} \equiv \overline{A} \downarrow \overline{B} & A \cap B &\equiv \overline{\overline{A \cap B}} \equiv \overline{A|B} \\
&\equiv (A \downarrow A) \downarrow (B \downarrow B) & &\equiv (A|B)|(A|B)
\end{aligned}
\tag{4.8}
$$

Si osservi che invece *non* è possibile ottenere il NOT usando solo AND o solo OR.
Come vedremo nel seguito, il fatto che AND-NOT e OR-NOT siano insiemi minimi grazie ai quali possiamo rappresentare tutte le 16 possibili combinazioni di tabella 4.3 e che NOR e NAND siano connettivi universali, avrà un'importanza fondamentale in ambito tecnico-applicativo.

4.2 Algebra Booleana

4.2.1 Impostazione assiomatica

La formalizzazione e la soluzione di problemi del mondo reale, quale quello appena analizzato riguardante le cinture di sicurezza, richiede di poter usare le tavole di verità associate alle variabili binarie T e F, ma soprattutto di poter manipolare in modo rigoroso tali variabili e i rispettivi connettivi binari. Inoltre, potrebbe anche accadere che la formula 4.1 abbia delle espressioni equivalenti, che sono magari più semplici o più utili ai fini applicativi; bisogna allora essere in grado di esprimere in modo rigoroso queste formulazioni equivalenti. E' dunque necessario dotarsi di una tecnica affidabile di manipolazione delle espressioni del calcolo funzionale di verità.

In matematica, la disciplina che studia i simboli matematici e le regole per la loro manipolazione formale si chiama *Algebra*; l'Algebra che studia i valori logici *vero* e *falso* si chiama *Algebra Booleana*, e come ricordato precedentemente essa venne introdotta nel 1854 da George Boole, con l'opera *An Investigation of the Laws of Thought*. Vediamone una definizione assiomatica.

Definizione 4.1. *Un'Algebra Booleana è un insieme $\mathcal{B}$ caratterizzato da:*

A0 - Leggi di composizione *Ci sono due leggi di composizione interna di-arie, denominate rispettivamente AND e OR, e una legge di composizione interna unaria, denominata NOT; tali leggi (od "operatori Booleani") sono definite come segue:*

Operatore	Nome	Simboli usati	che soddisfa
AND(x, y)	congiunzione	$x \wedge y \quad x \cdot y$	$x \cdot y = 1$ se $x = 1, y = 1$
			$x \cdot y = 0$ altrimenti
OR(x, y)	disgiunzione	$x \vee y \quad x + y$	$x + y = 0$ se $x = 0, y = 0$
			$x + y = 1$ altrimenti
NOT(x)	negazione	$\neg x \quad \overline{x}$	$\overline{x} = 0$ se $x = 1$ e $\overline{x} = 1$ se $x = 0$

Inoltre, per gli elementi dell'insieme $\mathcal{B}$ valgono i seguenti assiomi:

A1 - Esistenza Esistono due elementi $x, y \in \mathcal{B}$, tali che $x \neq y$

A2 - Chiusura Se $x, y \in \mathcal{B}$, allora $x \cdot y \in \mathcal{B}$, $x + y \in \mathcal{B}$, $\overline{x} \in \mathcal{B}$, $\overline{y} \in \mathcal{B}$

A3 - Elemento neutro Esiste un elemento neutro per la disgiunzione, indicato con 0, tale che $x + 0 = x$

Esiste un elemento neutro per la congiunzione, indicato con 1, tale che $x \cdot 1 = x$

A4 - Commutatività $\forall x, y \in \mathcal{B}$ valgono le seguenti relazioni $x + y = y + x$

$$x \cdot y = y \cdot x$$

A5 - Associatività $\forall x, y, z \in \mathcal{B}$ valgono le seguenti relazioni $x + (y + z) = (x + y) + z$

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z$$

A6 - Distributività $\forall x, y, z \in \mathcal{B}$ valgono le seguenti relazioni $x + (y \cdot z) = (x + y) \cdot (x + z)$;

$$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$$

A7 -Complementarità $\forall x \in \mathcal{B}$ vale $x \cdot \overline{x} = 0$ e $x + \overline{x} = 1$

Il più semplice insieme $\mathcal{B}$ che soddisfa i postulati sopra è rappresentato dalla cosiddetta *Algebra Booleana a due elementi*, o *Algebra Binaria*, basata sugli elementi 0 e 1 e caratterizzata dalle seguenti tre leggi di composizione

$$\begin{array}{c|cc} \cdot & 0 & 1 \\ \hline 0 & 0 & 0 \\ 1 & 0 & 1 \end{array} \qquad \begin{array}{c|cc} + & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 1 \end{array} \qquad \begin{array}{c|cc} A & 0 & 1 \\ \hline \overline{A} & 1 & 0 \end{array} \tag{4.9}$$

L'Algebra Booleana a due elementi è quella che useremo per la progettazione delle reti logiche; i suoi due elementi 1 e 0 corrispondono alle costanti logiche *vero* e *falso* (V e F) prima introdotte, e le tre leggi di composizione interna sono rispettivamente AND, OR e NOT e corrispondono ai connettivi precedentemente introdotti nelle tavole di verità. In questo contesto la simbologia che si usa è però diversa; in particolare si osservi che i simboli $+$ e $\cdot$ nulla hanno a che vedere con gli stessi simboli usati nel contesto dell'aritmetica.
Un altro importante esempio di Algebra Booleana è quello derivante dalla teoria degli insiemi; si consideri un insieme S e il corrispondente insieme delle parti $\mathcal{P}(S)$, costituito dall'insieme di tutti i sottinsiemi costruiti con elementi di S. Allora $\mathcal{P}(S)$ è un Algebra Booleana, nella quale sono definite tre leggi di composizione interna date dall'intersezione $\cap$, dall'unione $\cup$ e dalla complementazione tra insiemi. L'elemento neutro per la disgiunzione, o 0 dell'insieme, è costituito dall'insieme vuoto, mentre l'elemento neutro per la congiunzione, o 1 dell'insieme, è l'insieme S.
Disponendo degli operatori Booleani e agendo sugli elementi di $\mathcal{B}$, potremmo ricavare infinite relazioni notevoli, dette *leggi Booleane* o *teoremi*, alcune delle quali sono già state viste precedentemente con le tavole di verità; un esempio potrebbe essere la relazione (4.4) che esprime i teoremi di De Morgan. Una legge Booleana è dunque un'identità tra due termini Booleani, costruiti su un insieme di variabili e sulle costanti 0 e 1 a partire dagli operatori $\wedge, \vee, \neg$. Per verificare una legge è sufficiente procedere con la cosiddetta *induzione perfetta*, che consiste nella sostituzione di tutti i possibili valori per le variabili, verificando che l'uguaglianza sia sempre soddisfatta.

Le leggi Booleane sono ovviamente infinite, ma poiché l'Algebra Booleana è un insieme *finitamente assiomatizzabile*, è possibile descrivere compiutamente l'Algebra Booleana usando solamente un insieme *finito* di leggi base, che sono gli *assiomi* precedentemente descritti. Ciò significa che partendo dagli assiomi si possono poi ricavare tutte le leggi (o teoremi) dell'Algebra Booleana, combinando gli assiomi e i teoremi che ne derivano in tutti i modi possibili.
Gli assiomi non sono soggetti a verifica, nel senso che sono verità primitive che costituiscono il punto di partenza di una teoria matematica. Essi devono inoltre essere non contradditori tra loro e possibilmente indipendenti. Quest'ultima caratteristica è in qualche modo legato al seguente problema: assegnato un insieme di assiomi, esso è il minimo possibile o ne esiste uno più piccolo? Per esempio, nel nostro caso si può osservare che l'operatore disgiunzione $\cdot$, usato come operatore base nel quadro assiomatico appena analizzato, può essere definito in funzione degli altri due operatori $+$ e $\neg$, come già verificato nel teorema di De Morgan 4.1; ecco allora che potremmo rimuoverlo, pervenendo a un quadro assiomatico più sintetico. Il problema fu affrontato da Edward V. Huntington, che nel 1933 riuscì a formulare un insieme minimo di assiomi tra loro indipendenti, che partono dai soli operatori $+$ e $\neg$ e che prevedono, oltre alla commutatività e all'associatività, anche la cosiddetta equazione di Huntington

$$\overline{(\overline{x} + y)} + \overline{(\overline{x} + \overline{y})} = x \tag{4.10}$$

4.2.2 Teoremi principali dell'Algebra Booleana

A partire dagli assiomi sopra esposti, si potrebbero ricavare tutti i possibili teoremi dell'Algebra Booleana; tra questi ce ne sono alcuni molto semplici e utili nella manipolazione delle formule; li elenchiamo di seguito

	base (b)	duale (d)
T1 - Idempotenza	$x + x = x$	$x \cdot x = x$
T2 - Nullifico	$x + 1 = 1$	$x \cdot 0 = 0$
T3 - Doppia negazione	$\overline{\overline{x}} = x$	
T4 - Assorbimento 1	$x + x \cdot y = x$	$x \cdot (x + y) = x$
T5 - Assorbimento 2	$x + \overline{x} \cdot y = x + y$	$x \cdot (\overline{x} + y) = x \cdot y$
T6 - Assorbimento 3	$x \cdot y + \overline{x} \cdot z + y \cdot z = x \cdot y + \overline{x} \cdot z$	$(x+y) \cdot (\overline{x}+z) \cdot (y+z) = (x+y) \cdot (\overline{x}+z)$
T7 - De Morgan	$\overline{x + y} = \overline{x} \cdot \overline{y}$	$\overline{x \cdot y} = \overline{x} + \overline{y}$
T8 -	$x \cdot (x + y + z) = x$	$x + (x \cdot y \cdot z) = x$
T9 -	$(x + y) \cdot (\overline{x} + y) = y$	$(x \cdot y) + (\overline{x} \cdot y) = y$
T10 -	$(x + y) \cdot (\overline{x} + z) = x \cdot z + \overline{x} \cdot y$	$(x \cdot y) + (\overline{x} \cdot z) = (x + z) \cdot (\overline{x} + y)$

La dimostrazione di uno qualunque di questi teoremi può essere fatta per induzione perfetta, che è la via più lunga e tediosa, oppure sfruttando gli assiomi e/o i teoremi già dimostrati. Vediamo alcune dimostrazioni tralasciando il segno "$\cdot$", per non appesantire la notazione. T1-T3 sono banali, poichè basta fare una verifica diretta tramite induzione perfetta. Vediamo la dimostrazione di T4, T5 e T6, mettendo tra parentesi, subito dopo il segno di uguaglianza, l'assioma o il teorema usato.

T4 - Assorbimento 1

base $x + xy \overset{(A3)}{=} x \cdot 1 + xy \overset{(A6)}{=} x(1 + y) \overset{(T2)}{=} x$

duale $x(x + y) \overset{(A6)}{=} xx + xy \overset{(T1)}{=} x + xy \overset{(T4.b)}{=} x$

T5 - Assorbimento 2

base $x + \overline{x}y \overset{(T2)}{=} x(1 + y) + \overline{x}y \overset{(A6)}{=} x + xy + \overline{x}y \overset{(A6)}{=} x + (x + \overline{x})y \overset{(A7)}{=} x + y$

duale $x(\overline{x} + y) \overset{(A6)}{=} x\overline{x} + xy \overset{(A7)}{=} xy$

T6 - Assorbimento 3

base $xy + \overline{x}z + yz \overset{(A7)}{=} xy + \overline{x}z + yz(x + \overline{x}) \overset{(A6)}{=} xy + \overline{x}z + yzx + yz\overline{x} \overset{(A6)}{=}$

$xy(z + 1) + \overline{x}z(y + 1) \overset{(T2)}{=} xy + \overline{x}z$

duale $(x + y)(\overline{x} + z)(y + z) \overset{(A7)}{=} (x + y)(\overline{x} + z)(y + z + x\overline{x}) \overset{(A6)}{=}$

$(x + y)(\overline{x} + z)(y + z + x)(y + z + \overline{x}) \overset{(A4)}{=}$

$(x + y)(x + y + z)(\overline{x} + z)(\overline{x} + z + y) \overset{(T4)}{=} (x + y)(\overline{x} + z)$

La verifica del teorema T7 di De Morgan, basata l'induzione perfetta, è già stata fatta nella dimostrazione 4.5 del teorema 4.1; la ripetiamo qua per comodità del lettore, ricordando la fondamentale importanza di questo teorema per le applicazioni e per la semplificazione delle espressioni complesse

x	y	$x + y$	$\overline{x + y}$	$\overline{x}$	$\overline{y}$	$\overline{x} \cdot \overline{y}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

x	y	$x \cdot y$	$\overline{x \cdot y}$	$\overline{x}$	$\overline{y}$	$\overline{x} + \overline{y}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

Lasciamo infine i teoremi T8-T9 e T10 per esercizio al lettore.

4.2.3 Principio di dualità

Se osserviamo l'elenco dei teoremi, possiamo notare che le proposizioni della colonna di destra possono essere ottenute dalla colonna centrale semplicemente scambiando "0" con "1" e "+" con "·" e viceversa. Questa è una manifestazione del cosiddetto *Principio di dualità*. Per comprenderlo riprendiamo la tabella di figura 4.1, limitatamente ai valori di AND e OR, e facciamo lo scambio "0" con "1" e "+" con "·"; si ottiene

x	y	$x+y$	$x\cdot y$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

x	y	$x\cdot y$	$x+y$
1	1	1	1
1	0	0	1
0	1	0	1
0	0	0	0

Figura 4.5: Principio di dualità: scambiando "0" con "1" e "+" con "·" la tavola di verità continua a valere

Si può osservare che la tavola di verità che ne deriva conserva la propria validità, sia pur con una permutazione delle righe, del tutto ininfluente. Il Principio di dualità consente allora di trasformare in modo duale le espressioni Booleane, realizzando nuove espressioni che continuano a valere. Si osservi tuttavia che i valori delle espressioni così ottenute sono in generale diversi da quelle di partenza; per charire meglio il concetto riprendiamo p.es. il teorema T10; l'espressione Booleana $(x+y)\cdot(\overline{x}+z)$ ha come duale la $x\cdot y+\overline{x}\cdot z$.

1	2	3	4	5
x	y	z	$(x+y)\cdot(\overline{x}+z)$	$x\cdot y+\overline{x}\cdot z$
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	0
1	1	0	0	1
1	1	1	1	1

(a)

6	7	8	9
x	y	z	$x\cdot y+\overline{x}\cdot z$
1	1	1	1
1	1	0	1
1	0	1	0
1	0	0	0
0	1	1	1
0	1	0	0
0	0	1	1
0	0	0	0

(b)

Figura 4.6: Principio di dualità applicato all'espressione Booleana $(x+y)\cdot(\overline{x}+z)$

Nella tabella di figura 4.6a sono riportati, in colonna 4 e 5, i valori di queste due espressioni duali, e come si vede sono diversi. Per verificare la validità del Principio di dualità dobbiamo ora sostituire "0" con "1" e "+" con "·" nelle colonne $1, 2, 3$ e nella colonna 4, che riporta i valori dell'espressione; così facendo otteniamo rispettivamente le colonne $6, 7, 8$ e 9. Si può verificare che a ciascuna terna delle colonne duali $6, 7, 8$ corrisponde un valore duale di colonna 9 che è esattamente il valore attribuito dall'espressione di colonna 5 alla stessa terna letta nelle colonne primitive $2, 3, 4$. Per esempio, a 011 delle colonne $6, 7, 8$ corrisponde 1 in colonna 9; questo è lo stesso valore attribuito dalla colonna 5 alla terna 011 delle colonne primitive $1, 2, 3$.

4.3 Variabili, funzioni Booleane e porte logiche

Se osserviamo le colonne 4 o 5 della tabella di figura 4.6, vediamo che a ogni terna di valori per le variabili Booleane x, y, z viene associato un valore per l'espressione corrispondente. In Matematica un'associazione di questo genere si chiama *funzione*; in generale una funzione è una relazione tra due insiemi, chiamati rispettivamente *dominio* e *codominio* della funzione, che associa a ogni elemento del dominio uno e un solo elemento del codominio. Se chiamiamo X e Y i due insiemi, una generica funzione si indica con il simbolo $f : X \to Y$. Nel nostro caso l'insieme X corrisponde all'insieme di tutte e sole le possibili 2^3 terne binarie, che possiamo rappresentare col simbolo $X = \mathbf{2^3}$;[1] Y è invece l'insieme $\{0, 1\}$, che rappresentiamo analogamente col simbolo $Y = \mathbf{2}$. Ecco allora che la funzione rappresentata dalla colonna 4 della tabella di figura 4.6 è una funzione del tipo $f : \mathbf{2^3} \to \mathbf{2}$. Se dunque $x_1, x_2, \cdots, x_n$ sono n variabili Booleane (o *binarie*) che possono assumere l'uno o l'altro dei due valori 0 e 1, si indica con

$$f(x_1, x_2, \cdots, x_n)$$

una generica *funzione Booleana* del tipo

$$f : \mathbf{2^n} \to \mathbf{2}$$

che a ogni valore dell'n-pla $x_1, x_2, \cdots, x_n$ associa un valore dell'insieme $\{0, 1\}$. La tabella di verità di una simile funzione è rappresentata in figura 4.7.

x_1	x_2	$\cdots$	x_{n-1}	x_n	$f(x_1, x_2, \cdots, x_n)$
0	0	$\cdots$	0	0	$f(0, 0, \cdots, 0, 0)$
0	0	$\cdots$	0	1	$f(0, 0, \cdots, 0, 1)$
		$\vdots$			$\vdots$
1	1	$\cdots$	1	0	$f(1, 1, \cdots, 1, 0)$
1	1	$\cdots$	1	1	$f(1, 1, \cdots, 1, 1)$

Figura 4.7: Tabella di verità di una generica funzione Booleana n-aria

Si osservi che il numero di n-ple binarie dell'insieme $\mathbf{2^n}$ cresce in modo esponenziale; diventa dunque inagevole rappresentare le funzioni Booleane mediante tavole di verità simili a quella di figura 4.6 o 4.7, se non che per valori molto limitati di n.
Poiché l'insieme $\mathbf{2^n}$ contiene 2^n n-ple binarie, e a ciascuna di esse si associa un valore della funzione, il numero totale di funzioni Booleane n-arie che si possono realizzare risulta essere

$$2^{2^n}$$

Risulta fondamentale, per tutto quanto faremo nel seguito, studiare in particolare le funzioni Booleane per $n = 1$ (a una variabile, o *unarie*) e $n = 2$ (a due variabili, o *di-arie*).

4.3.1 Funzioni a una variabile

Con $n = 1$ ci sono in tutto $2^{2^1} = 4$ funzioni unarie del tipo $y = f(x)$. Passiamole in rassegna:

[1] si osservi che l'uso del grassetto sta proprio a indicare che $\mathbf{2^3}$ non è il numero $2^3 = 8$, ma un simbolo, cioè il simbolo dell'insieme che contiene tutti e sole le 2^3 terne binarie

	f_0	f_1	f_2	f_3
x	**0**	x	$\bar{x}$	**1**
0	0	0	1	1
1	0	1	0	1

	0
x	z
0	0
1	0

	1
x	z
0	1
1	1

	Buffer
x	z
0	0
1	1

	NOT
x	z
0	1
1	0

Figura 4.8: Tutte le possibili $2^{2^1} = 4$ funzioni Booleane a 1 variable

La prima funzione fornisce sempre 0 in uscita, qualunque sia l'ingresso; è dunque la *funzione nulla*, che denotiamo con **0**. Un discorso analogo vale per la seconda funzione, che fornisce sempre 1 in uscita (*funzione unitaria*), e che denotiamo con **1**. La terza funzione replica il valore di x in uscita, cioè $y = x$, e si tratta dunque della *funzione identità*. Poiché la logica Booleana è connessa, come già anticipato, all'ambito circuitale, per la rappresentazione della funzione identità si usa il termine *Buffer*, e le si attribuisce lo schema circuitale dato da un triangolo; esso corrisponde a un dispositivo che fornisce sull'uscita a destra esattamente cò che si presenta in ingresso a sinistra. La quarta funzione è la più importante dal punto di vista della logica Booleana, poiché è uno dei connettivi base già incontrati, cioè NOT. Esso fornisce in uscita la negazione di quanto sta all'ingresso; si può dunque scrivere $y = \bar{x}$. Per la sua rappresentazione circuitale si usa il triangolo di prima concatenato con un piccolo circoletto, che in tutta la circuiteria logica ha sempre il significato di una *negazione* o *complementazione*.

4.3.2 Funzioni a due variabili

Con $n = 2$ ci sono in tutto $2^{2^2} = 16$ funzioni Booleane di-arie del tipo $z = f(x, y)$. Le abbiamo già implicitamente incontrate in figura 4.2 parlando di connettivi binari; le riportiamo di seguito usando il nuovo linguaggio dell'Algebra Booleana:

		f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
			AND					XOR	OR	NOR	XNOR					NAND	
x	y	**0**	$\cdot$					$\oplus$	$+$	$\downarrow$	$\odot$					$\mid$	**1**
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Figura 4.9: Tutte le possibili $2^{2^2} = 16$ funzioni Booleane a due variabili

Tra tutte queste funzioni sappiamo che alcune sono particolarmente significative, precisamente le sei già evidenziate in figura 4.1, che sono rispettivamente AND, OR, XOR, NAND, NOR e XNOR; nella tabella seguente esse sono espresse, assieme alle rimanenti, mediante i simboli $\cdot$, $+$, $\oplus$, $\mid$, $\downarrow$ e $\odot$, che si usano tradizionalmente nell'ambito dell'Algebra Booleana; si noti che $+$, $\cdot$ e $\odot$ stanno rispettivamente per $\cup$, $\cap$ e $\equiv$, visti nella precedente tabella 4.1 parlando di connettivi.

$f_0 = \mathbf{0}$		$f_8 = x \downarrow y$	NOR
$f_1 = x \cdot y$	AND	$f_9 = x \odot y$	XNOR
$f_2 = x \cdot \overline{y}$		$f_{10} = \overline{y}$	
$f_3 = x$		$f_{11} = x + \overline{y}$	
$f_4 = \overline{x} \cdot y$		$f_{12} = \overline{x}$	
$f_5 = y$		$f_{13} = \overline{x} + y$	
$f_6 = x \oplus y$	XOR	$f_{14} = x \mid y$	NAND
$f_7 = x + y$	OR	$f_{15} = \mathbf{1}$	

Figura 4.10: Le 16 funzioni Booleane della figura 4.9 espresse mediante $+$, $\cdot$, $\oplus$, $\odot$, $\downarrow$, $\mid$ e complementazione

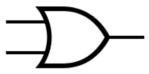
Data l'estrema importanza, in ambito circuitale, delle funzioni AND, OR, XOR, NAND, NOR e XNOR, dette anche *operatori Booleani*, le analizziamo singolarmente facendo riferimento alle tabelle di figura 4.15. Nella stessa figura sono evidenziati anche i simboli schematici, detti *Porte Logiche*, che vengono associati alle suddette funzioni nella descrizione dei circuiti logici.

AND E' detta anche *prodotto logico*. La porta AND restituisce 0 in uscita se anche uno solo dei due valori d'ingresso è pari a 0; restituisce 1 solo quando entrambi gli ingressi sono a 1. Il simbolo circuitale è

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

OR E' detta anche *somma logica*. La porta OR restituisce 1 in uscita se almeno uno dei due valori d'ingresso è pari a 1; restituisce 0 solo quando entrambi gli ingressi sono a 0. Il simbolo circuitale è

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1



XOR E' l'OR esclusivo. La porta XOR restituisce 1 in uscita se o uno o l'altro dei due valori d'ingresso è pari a 1, ma non entrambi; restituisce 0 quando entrambi gli ingressi sono a 0 o a 1. Il simbolo circuitale è

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

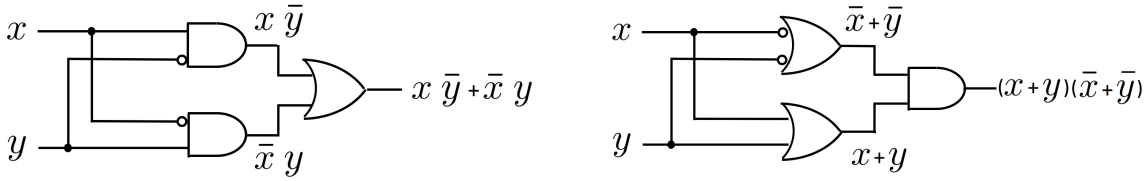
Si osservi che una porta XOR è in grado di realizzare la somma binaria modulo 2 e che costituisce un *rilevatore di differenza* tra i due valori d'ingresso. Dalla figura 4.3 possiamo osservare che la funzione f_6 corrisponde a due espressioni; la prima è

$$x \oplus y = x\overline{y} + \overline{x}y \tag{4.11}$$

che corrisponde a un OR di due AND. Mediante manipolazione algebrica si ottiene anche la seconda espressione

$$x \oplus y = x\bar{y} + \bar{x}y = x\bar{y} + \bar{x}y + x\bar{x} + y\bar{y} = (x + y) \cdot (\bar{x} + \bar{y}) \tag{4.12}$$

che corrisponde a un AND di due OR. Si possono pertanto realizzare due circuiti del tutto equivalenti, visibili in figura 4.11



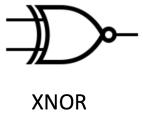
(a) Realizzazione basata sull'equazione 4.11

(b) Realizzazione basata sull'equazione 4.12

Figura 4.11: La funzione XOR realizzata mediante porte AND e OR

XNOR E' la negazione della porta XOR. La porta XNOR restituisce 1 in uscita solo quando entrambe le variabili d'ingresso hanno lo stesso valore, o entrambe a 0 oppure entrambe a 1. Il simbolo circuitale si ottiene concatenando la porta XOR con la porta NOT, rappresentata dal circoletto

x	y	$x \odot y$
0	0	1
0	1	0
1	0	0
1	1	1

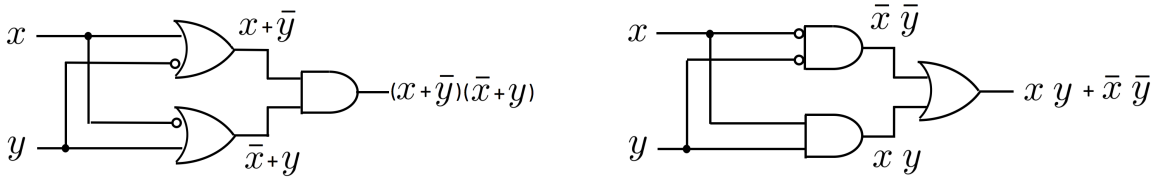


Si osservi che una porta XNOR costituisce un *rilevatore di identità* tra i due valori d'ingresso. Per trovare una rappresentazione circuitale mediante porte AND e OR bisogna partire dal fatto che si tratta di uno XOR negato, traendone le conseguenze sempre usando il teorema T7 di De Morgan

$$x \odot y = \overline{x \oplus y} = \overline{\bar{x}y + x\bar{y}} = \overline{(\bar{x}y)} \cdot \overline{(x\bar{y})} = (x + \bar{y}) \cdot (\bar{x} + y) = \tag{4.13}$$

$$= x\bar{x} + xy + \bar{y}\bar{x} + \bar{y}y = xy + \bar{x}\bar{y} \tag{4.14}$$

Dalla prima riga otteniamo $(x + \bar{y}) \cdot (\bar{x} + y)$, e quindi l'AND di due OR, che corrisponde alla realizzazione circuitale di figura 4.12a; dalla seconda otteniamo invece $xy + \bar{x}\bar{y}$, cioè l'OR di due AND, che porta alla realizzazione circuitale di figura 4.12b



(a) Realizzazione basata sull'equazione 4.13

(b) Realizzazione basata sull'equazione 4.14

Figura 4.12: La funzione XNOR realizzata mediante porte AND e OR

NAND E' la negazione dell'AND. La porta NAND restituisce 1 in uscita se almeno uno dei due valori d'ingresso è pari a 0; restituisce 0 solo quando entrambi gli ingressi sono a 1. Il simbolo circuitale si ottiene concatenando

la porta AND con la porta NOT, rappresentata dal circoletto

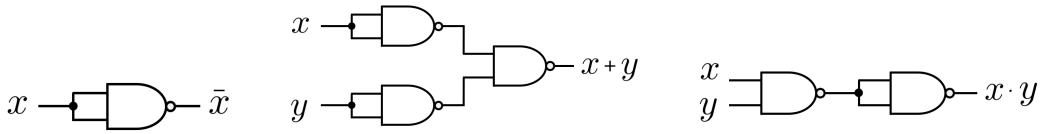
x	y	$x \mid y$
0	0	1
0	1	1
1	0	1
1	1	0

Nella tabella 4.4 avevamo anticipato che AND e NOT, oppure OR e NOT, sono sufficienti per esprimere una qualunque tra le 16 possibili funzioni; inoltre dalle equazioni (4.8) abbiamo visto che tramite una porta NAND si possono realizzare l'AND, l'OR e il NOT; ciò significa che la porta NAND è un operatore *universale*, nel senso che da sola consente di costruire una qualunque tra le 16 possibili funzioni. Per declinare le relazioni (4.8) nei termini del linguaggio dell'Algebra Booleana è sufficiente usare gli assiomi e i teoremi che abbiamo visto poc'anzi

$$\overline{x} = \overline{x \cdot x} = x|x \qquad x + y = \overline{\overline{x} \cdot \overline{y}} = \overline{x}|\overline{y} \qquad x \cdot y = \overline{\overline{x \cdot y}} = \overline{x|y}$$

NOT OR AND

Le figure 4.13a, 4.13b e 4.13c mostrano la realizzazione circuitale corrispondente.



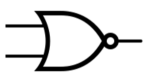
(a) Funzione NOT (b) Funzione OR (c) Funzione AND

Figura 4.13: Funzioni NOT, OR e AND realizzate mediante una porta universale NAND

La giustificazione del NOT è evidente: poichè nel circuito sono ammesse le sole due configurazioni d'ingresso in cui $x = y = 1$ oppure $x = y = 0$, per il primo caso si ottiene l'uscita a 1, che complementata va a 0, realizzando la negazione dell'ingresso; nel secondo caso si ha l'uscita a 0, che complementata va a 1, realizzando nuovamente la negazione dell'ingresso. L'OR si giustifica mediante il teorema T7 di De Morgan; ecco allora che prima di entrare nel NAND bisogna complementare le variabili con un NOT realizzato tramite due porte NAND. Per quanto riguarda la costruzione dell'AND è evidente che esso si realizza aggiungendo un NOT alla porta NAND.

NOR E' la negazione dell'OR. La porta NOR restituisce 0 in uscita se anche uno solo dei due valori d'ingresso è pari a 1; restituisce 1 solo quando entrambi gli ingressi sono a 0. Il simbolo circuitale si ottiene concatenando la porta OR con la porta NOT, rappresentata dal circoletto

x	y	$x \downarrow y$
0	0	1
0	1	0
1	0	0
1	1	0

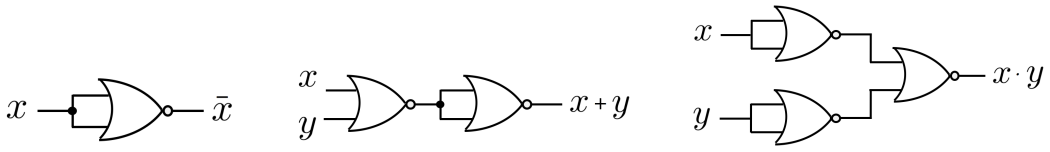


Ricordiamo che anche l'operatore NOR è un operatore *universale*, che consente di costruire una qualunque tra le 16 possibili funzioni; ciò deriva dalla circostanza che, similmente al caso della porta NAND, col NOR si possono costruire le porte NOT, OR e AND mediante l'uso degli assiomi e dei teoremi prima visti

$$\overline{x} = \overline{x + x} = x \downarrow x \qquad x + y = \overline{\overline{x + y}} = \overline{x \downarrow y} \qquad x \cdot y = \overline{\overline{x} + \overline{y}} = \overline{x} \downarrow \overline{y}$$

NOT OR AND

Le figure 4.14a, 4.14b e 4.14c mostrano la realizzazione circuitale corrispondente.



(a) Funzione NOT (b) Funzione OR (c) Funzione AND

Figura 4.14: Funzioni NOT, OR e AND realizzate mediante una porta universale NOR

La giustificazione del NOT è identica a prima: poichè nel circuito sono ammesse le sole due configurazioni d'ingresso in cui $x = y = 1$ oppure $x = y = 0$, nel primo caso l'uscita è a 1 e complementata va a 0, realizzando la negazione dell'ingresso; nel secondo caso l'uscita è a 0 e complementata va a 1, realizzando nuovamente la negazione dell'ingresso. Per quanto riguarda la costruzione dell'OR è evidente che esso si realizza aggiungendo un NOT alla porta NOR. L'AND si giustifica invece mediante il teorema T7 di De Morgan; ecco allora che prima di entrare nel NOR bisogna complementare le variabili con un NOT realizzato tramite due porte NOR. Si osservi che questa relazione è la duale di quella usata per il NAND.

AND

x	y	AND
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	OR
0	0	0
0	1	1
1	0	1
1	1	1

XOR

x	y	XOR
0	0	0
0	1	1
1	0	1
1	1	0

NAND

x	y	NAND
0	0	1
0	1	1
1	0	1
1	1	0

NOR

x	y	NOR
0	0	1
0	1	0
1	0	0
1	1	0

XNOR

x	y	XNOR
0	0	1
0	1	0
1	0	0
1	1	1

Figura 4.15: Le 6 funzioni di-arie più importanti con i rispettivi simboli circuitali

Finora abbiamo trattato solo porte associate a funzioni a due variabili; la generalizzazione a n variabili ha pieno senso solo per le porte AND e OR; trattiamo per esteso il caso con $n = 3$, poiché gli altri sono la banale generalizzazione di questo. Le tavole di verità delle funzioni AND e OR a 3 variabili sono riportate nella successiva figura 4.16; la funzione AND vale 1 solo quando tutte le variabili in ingresso hanno valore 1; la funzione OR vale 0 solo quando tutte le variabili in ingresso hanno valore 0.

x	y	z	AND
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

(a) La funzione AND a 3 variabili

x	y	z	OR
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

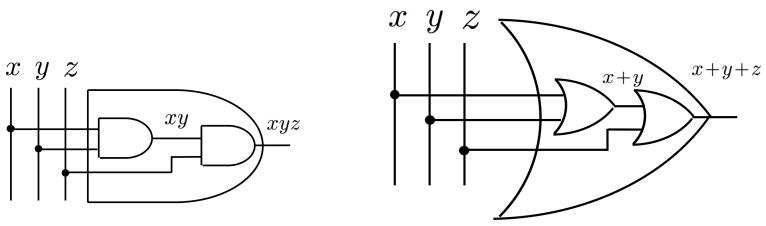
(b) La funzione OR a 3 variabili

Figura 4.16: Le funzioni AND e OR a 3 variabili

Si osservi che per l'associatività A5 di prodotto e somma logiche si ha

$$x + y + z = (x + y) + z \qquad x \cdot y \cdot z = (x \cdot y) \cdot z$$

e dunque una porta AND o una porta OR a tre variabili si realizza usando due porte a due variabili, collegate come in figura 4.17. L'associatività A5 di prodotto e somma logiche costituisce anche una giustificazione teorica alla circostanza di poter gestire con connettivi binari quali AND e OR a due variabili funzioni aventi un numero qualunque di variabili.



(a) Porta AND a 3 variabili

(b) Porta OR a 3 variabili

Figura 4.17: Porte AND e OR a 3 variabili

4.3.3 Realizzazione circuitale delle porte logiche

Le funzioni logiche di base, a 1 e 2 variabili, possono essere facilmente realizzate impiegando i dispositivi che abbiamo introdotto nel capitolo 3, cioè i tubi termoionici, i diodi, i transistor e i C-MOS. Lasciando perdere i tubi a vuoto, la cui tecnologia è ampiamente superata, la realizzazione delle porte logiche si è sviluppata nell'arco temporale perseguendo obiettivi di minimazione della potenza dissipata, dei tempi di risposta e dei costi, ottenendo congiuntamente un aumento progressivo della densità di integrazione sui relativi circuiti integrati. Il risultato è stato la costituzione delle cosiddette *famiglie logiche*, che a seconda del periodo in cui sono state introdotte e dello sviluppo tecnologico corrente hanno privilegiato l'uso di uno o dell'altro dei vari dispositivi a semiconduttore per la realizzazione delle funzioni di base. Anche se in pratica sono molte di più, le principali famiglie logiche sono le seguenti (in ordine cronologico):

Resistor-Transistor Logic (RTL) è una classe di porte logiche costruite usando resistenze nella rete d'ingresso e transistori bipolari a giunzione (BJT) come dispositivi di commutazione. La RTL è stata la prima classe di

circuito logico digitale basata sull'impiego dei transistor. Le prime porte RTL furono costruite con elementi discreti, ma nel 1961 divenne la prima famiglia logica a esser realizzata su un circuito integrato monolitico. Circuiti integrati RTL sono stati utilizzati nel computer *Apollo Guidance*, il cui progetto risale al 1961 con un primo volo fatto nel 1966.

Diode-Transistor Logic (DTL) è la classe di di porte logiche che precede la grande famiglia TTL. Si chiama così perché la funzione logica viene eseguita da una rete di diodi, mentre la funzione di inversione-amplificazione viene eseguita da un transistore.

Transistor-Transistor Logic (TTL) è senza dubbio la classe più nota e famosa, poiché ha avuto un larghissimo impiego. Fa uso di transistori bipolari a giunzione (BJT) e resistenze. Si chiama logica transistor-transistor perché il transistor svolge sia la funzione logica che la funzione di inversione-amplificazione. I circuiti integrati basati sulla famiglia TTL sono stati ampiamente usati in applicazioni quali computer, controlli industriali, apparecchiature di prova, strumentazione elettronica, elettronica di consumo, sintetizzatori e molto altro. Dopo la loro introduzione come circuito integrato a opera di *Sylvania* nel 1963, i circuiti integrati TTL sono stati prodotti da diverse aziende di semiconduttori. La serie 7400 (chiamato anche 74xx) della *Texas Instruments* è diventata particolarmente popolare. I produttori di porte basati sulla tecnologia TTL hanno offerto una vasta gamma di porte logiche, flip-flop, contatori, multiplexer e altri circuiti.

Complementary Metal-Oxide-Semiconductor (CMOS) è la tecnologia più recente per la costruzione di porte logiche inserite in circuiti integrati. La tecnologia CMOS è usata in microprocessori, microcontrollori, nella RAM statica e in molti altri circuiti logici digitali. E' la tecnologia più raffinata, che consente un abbattimento dei consumi e dei tempi di risposta, congiuntamente alla possibilità di realizzare un'altissima densità di integrazione.

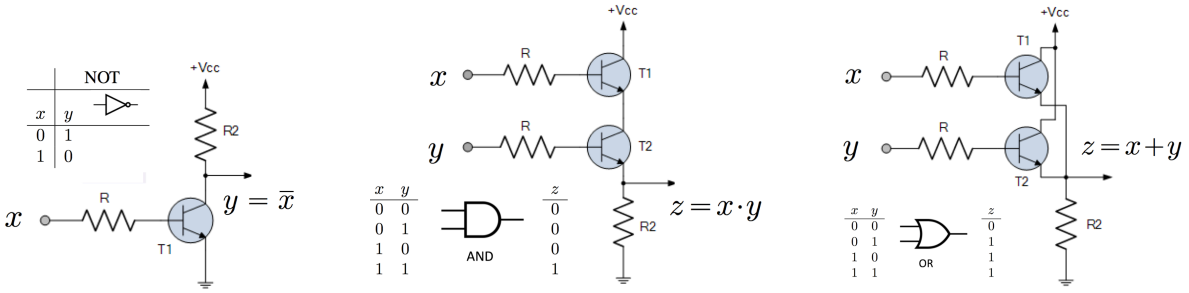
Non è questa la sede per descrivere, nel dettaglio, la realizzazione di tutte le porte logiche nelle varie tecnologie; ci limiteremo a una mera illustrazione didattica, che ha lo scopo essenziale di colmare la connessione che ancora manca tra porta logica astratta, che esprime una certa funzione Booleana, e la circuiteria che la realizza concretamente. Dati gli scopi limitati della nostra trattazione, faremo riferimento a circuiti assimilabili alla famiglia RTL, con qualche anticipazione sulle porte TTL e CMOS. Nei corsi di *Reti Logiche* ed *Elettronica* sarà poi possibile studiare meglio la struttura e il funzionamento delle porte più complesse.

NOT Nella porta NOT bisogna realizzare un'inversione del valore logico. Questa funzione si realizza mediante un singolo transistor, nel quale la variabile d'ingresso va ad alimentare la base, mentre quella di uscita si ricava sul collettore del transistor. Se associamo la costante 1 a un livello alto di tensione, p.es. la V_{CC} di figura 4.18a, e la costante 0 a un livello basso, cioè la tensione di massa pari a $0\,V$, il funzionamento dell'invertitore è il seguente: quando la variabile x assume valore logico 1, e cioè viene portata a tensione V_{CC}, il transistor si polarizza direttamente ed entra in piena conduzione. Tale condizione corrisponde alla saturazione vista in figura 3.25[2]; la tensione V_{CE} crolla idealmente a $0\,V$, portando la variabile y in uscita nello stato logico 0[3]. Quando viceversa la variabile x assume valore logico 0, e cioè viene portata a tensione $0\,V$ (connessione a massa), il transistor blocca la conduzione e la I_C diventa (praticamente) nulla. Tale condizione corrisponde all'interdizione vista in figura 3.25; la tensione V_{CE} diventa pari alla V_{CC}, portando a 1 logico il valore dell'uscita.

AND Nella porta AND bisogna fare in modo che se anche uno solo degli ingressi è a 0, l'uscita resti a 0. Il circuito di figura 4.18b realizza questa condizione. Poiché l'uscita a 1 significa livello alto di tensione, per realizzarla bisogna che entrambi i due transistor associati ai due ingressi, che sono posti in serie, siano in piena conduzione. Per realizzare ciò è necessario portare le variabili x e y d'ingresso a V_{CC}. Se anche uno solo dei due ingressi rimane a livello basso, uno dei due transistor va in interdizione e la tensione di uscita resta a livello basso, il che corrisponde a 0 logico.

[2] Per ottenere la saturazione bisogna dimensionare adeguatamente le resistenze R e R_2

[3] In realtà sussiste la tensione V_{CEsat}, di circa $0,3\,V$ per il germanio e $0,6\,V$ per il silicio

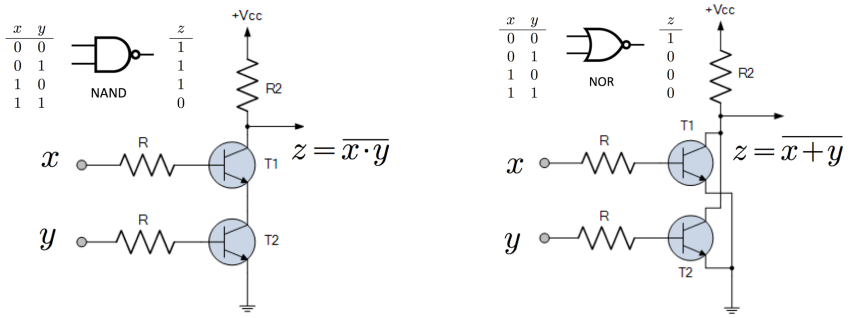


(a) Porta NOT (b) Porta AND (c) Porta OR

Figura 4.18: Porte NOT, AND e OR realizzate in tecnologia RTL

OR Nella porta OR bisogna fare in modo che se anche uno solo degli ingressi è a 1, l'uscita resti a 1. Il circuito di figura 4.18c realizza questa condizione. Poiché l'uscita a 1 significa livello alto di tensione, per realizzarla basta che anche uno solo dei due transistor associati ai due ingressi, che sono posti in parallelo, sia in piena conduzione. Per realizzare ciò è sufficiente portare o l'ingresso x o l'ingresso y al valore V_{CC}. Solo se entrambi gli ingressi rimangono a livello basso i transistor sono in interdizione e la tensione di uscita resta a livello basso, il che corrisponde a 0 logico.

NAND La porta NAND viene realizzata a partire dal circuito della porta AND, spostando semplicemente la resistenza $R2$ di carico dall'emettitore di $T2$ al collettore di $T1$ (si veda la figura 4.19a). In questo modo, quando entrambi i transistor sono in piena conduzione grazie al fatto che entrambi gli ingressi sono a 1, l'uscita va a 0; e questa rimane l'unica condizione per la quale si ha uscita bassa. Infatti se anche uno solo degli ingressi va a 0, uno dei due transistor va in interdizione e l'uscita resta a 1.



(a) Porta NAND realizzata con tecnologia RTL (b) Porta NOR realizzata con tecnologia RTL

Figura 4.19: Porte NAND e NOR in tecnologia RTL

NOR Anche in questo caso si parte della porta negata, che è la OR, spostando la resistenza $R2$ di carico dall'emettitore di $T2 - T1$ al collettore di $T2 - T1$ (si veda la figura 4.19b). In questo modo, quando entrambi i transistor sono in interdizione grazie al fatto che entrambi gli ingressi sono a 0, l'uscita va a 1; e questa rimane l'unica condizione per la quale si ha uscita alta. Infatti se anche uno solo degli ingressi va a 1, uno dei due transistor va in saturazione e l'uscita collassa a 0.

4.3.4 Forme canoniche

Sia assegnata una qualunque funzione Booleana espressa mediante tavola di verità; tanto per fissare le idee possiamo prendere la funzione descritta dalla figura 4.6.

x	y	z	$f = (x+y)\cdot(\overline{x}+z)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

(a) La funzione Booleana $f = (x+y)\cdot(\overline{x}+z)$

x	y	z	$\overline{x}\,y\,\overline{z}$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

(b) Termine minimo $\overline{x}\,y\,\overline{z}$ associato a 010

Figura 4.20: Esempio di funzione Booleana e uno dei termini minimi a essa associati

Supponiamo ora di non sapere quale sia la sua espressione Booleana e di volerla, in qualche modo, ricavare. Osserviamo che la funzione vale 1 in corrispondenza delle terne 010, 011, 101 e 111, cioè quando in ingresso si presenta la terna 010, *oppure* la terna 011, *oppure* ... ecc. Questo ci consente di costruire la nostra funzione a partire dalla somma logica di funzioni *elementari*[4], cioè quelle funzioni che valgono 1 esattamente in corrispondenza di una e una sola configurazione d'ingresso, precisamente quella che entra nella somma logica; una tale funzione si chiama *termine minimo* (o *minterm*). Come esempio prendiamo la funzione che vale 1 solo in corrispondenza della terna 010; essa può essere espressa come prodotto logico di variabili dirette e negate, le prime in corrispondenza delle variabili che in ingresso valgono 1, le seconde in corrispondenza delle variabili che in ingresso valgono 0. Nel nostro caso si tratta della funzione $\overline{x}\,y\,\overline{z}$, rappresentata in figura 4.20b[5]. Un ragionamento analogo si può fare per gli tutti altri termini minimi che compaiono nella funzione f di figura 4.20a; così facendo si ottengono le funzioni $\overline{x}\,y\,z$ per 011, $x\,\overline{y}\,z$ per 101 e $x\,y\,z$ per 111. La funzione di figura 4.20a si può allora esprimere come *somma di prodotti* delle variabili d'ingresso, nella forma

$$f = \overline{x}\,y\,\overline{z} + \overline{x}\,y\,z + x\,\overline{y}\,z + x\,y\,z \tag{4.15}$$

La tecnica che abbiamo appena usato può essere generalizzata a qualunque funzione $y = f(x_1, x_2, \cdots, x_n)$ di n variabili, tenuto conto che i termini minimi m_i sono in tutto tanti quante sono le possibili configurazioni d'ingresso, cioè 2^n. Se codifichiamo le n-ple d'ingresso associate a ciascun termine minimo con il corrispondente intero rappresentato in notazione posizionale in base 2, possiamo indicare i termini minimi che compongono la sommatoria usando gli interi compresi tra 0 e $2^n - 1$. Tenendo conto di tutto ciò la generica funzione si rappresenta come

$$f(x_1, x_2, \cdots, x_n) = \sum_{i=0}^{2^n-1} \mu_i m_i = \sum_{i:\mu_i=1} m_i \tag{4.16}$$

dove μ_i è il valore assunto dalla funzione in corrispondenza del termine minimo m_i e $0 \leqslant i \leqslant 2^n - 1$; nel nostro esempio si ha $m_2 = \overline{x}y\overline{z}$, $m_3 = \overline{x}yz$, $m_5 = x\overline{y}z$, $m_7 = xyz$, $\mu_2 = \mu_3 = \mu_5 = \mu_7 = 1$, $\mu_0 = \mu_1 = \mu_4 = \mu_6 = 0$. La formula (4.16) rappresenta la funzione nei termini della cosiddetta *prima forma canonica* (o *somma di prodotti*). Se nella tavola di verità della nostra funzione mettiamo in evidenza la codifica dei termini minimi otteniamo

[4] A rigore si tratterebbe di un OR esclusivo, visto che le configurazioni d'ingresso si escludono l'un l'altra, nel senso che se compare una non può comparire contemporaneamente anche l'altra. Tuttavia, visto che non è possibile avere in ingresso due configurazioni in contemporanea, i valori di XOR e OR coincidono, e quindi si fa riferimento a quest'ultima.

[5] tralasciamo d'ora in poi l'uso del $\cdot$ per semplificare la notazione

		x	y	z	f	
m_0	$\overline{x}\overline{y}\overline{z}$	0	0	0	0	μ_0
m_1	$\overline{x}\overline{y}z$	0	0	1	0	μ_1
m_2	$\overline{x}y\overline{z}$	0	1	0	1	μ_2
m_3	$\overline{x}yz$	0	1	1	1	μ_3
m_4	$x\overline{y}\overline{z}$	1	0	0	0	μ_4
m_5	$x\overline{y}z$	1	0	1	1	μ_5
m_6	$xy\overline{z}$	1	1	0	0	μ_6
m_7	xyz	1	1	1	1	μ_7

Figura 4.21: Codifica dei termini minimi

$$f(x,y,z) = \sum_{i\in\{2,3,5,7\}} m_i = m_2 + m_3 + m_5 + m_7 = \overline{x}y\overline{z} + \overline{x}yz + x\overline{y}z + xyz \tag{4.17}$$

poiché $2, 3, 5$ e 7 sono le codifiche in base due di 010, 011, 101 e 111.

Un discorso del tutto analogo si può fare procedendo per dualità, e realizzando un *prodotto di somme*. Ciò richiede di analizzare i termini per i quali la funzione va a 0; dalla figura 4.20a osserviamo che ciò accade in corrispondenza delle terne 000, 001, 100 e 110, cioè $0, 1, 4$ e 6 con la codifica usata prima. L'idea è quella di esprimere il valore della funzione come prodotto di somme che sono sempre a 1, tranne che per una singola configurazione che le manda a 0; una funzione che vale sempre 1, tranne che per una singola configurazione per la quale vale 0 si chiama *termine massimo* (o *maxterm*), e viene indicata con M_i. In figura 4.22 viene rappresentato M_1; il modo più semplice per esprimere un termine massimo è quello di ricorrere alla somma logica di variabili dirette e negate, le prime in corrispondenza delle variabili che in ingresso valgono 0, le seconde in corrispondenza delle variabili che in ingresso valgono 1. Nel caso di figura 4.22 si tratta della funzione $x + y + \overline{z}$

x	y	z	$x + y + \overline{z}$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Figura 4.22: Termine massimo $x + y + \overline{z}$ associato a 001

Tenendo conto di tutto ciò si perviene alla *seconda forma canonica* (o *prodotti di somme*), nella quale la generica funzione si rappresenta come

$$f(x_1, x_2, \cdots, x_n) = \prod_{i=0}^{2^n-1} (\mu_i + M_i) = \prod_{i:\mu_i=0} M_i \tag{4.18}$$

dove $0 \leqslant i \leqslant 2^n - 1$, μ_i è il valore assunto dalla funzione in corrispondenza del termine massimo M_i, il quale si codifica secondo la procedura prima descritta; nel nostro esempio si ha $M_0 = x + y + z$, $M_1 = x + y + \overline{z}$, $M_4 = \overline{x} + y + z$, $M_6 = \overline{x} + \overline{y} + z$ e così via.
Se mettiamo in evidenza nella tavola di verità della nostra funzione la codifica dei termini massimi otteniamo

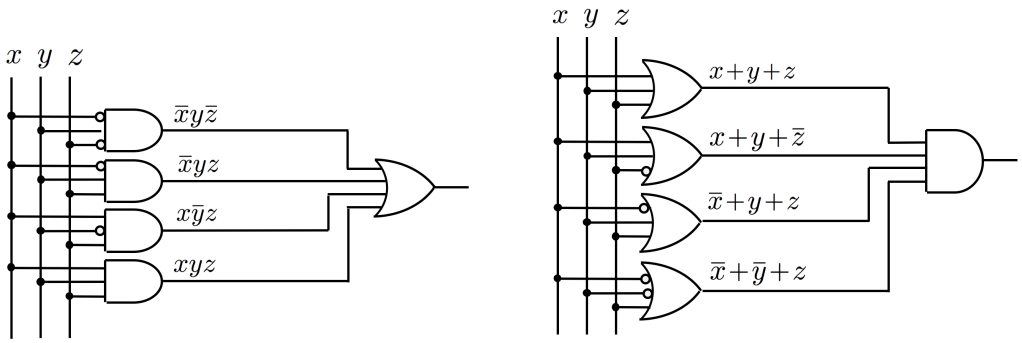
		x	y	z	f	
M_0	$x+y+z$	0	0	0	0	μ_0
M_1	$x+y+\bar{z}$	0	0	1	0	μ_1
M_2	$x+\bar{y}+z$	0	1	0	1	μ_2
M_3	$x+\bar{y}+\bar{z}$	0	1	1	1	μ_3
M_4	$\bar{x}+y+z$	1	0	0	0	μ_4
M_5	$\bar{x}+y+\bar{z}$	1	0	1	1	μ_5
M_6	$\bar{x}+\bar{y}+z$	1	1	0	0	μ_6
M_7	$\bar{x}+\bar{y}+\bar{z}$	1	1	1	1	μ_7

Figura 4.23: Codifica dei termini massimi

Esprimendo la funzione di figura 4.20a in questo modo si ottiene

$$f(x,y,z) = \prod_{i\in\{0,1,4,6\}} M_i = M_0 \cdot M_1 \cdot M_4 \cdot M_6 = (x+y+z)\cdot(x+y+\bar{z})\cdot(\bar{x}+y+z)\cdot(\bar{x}+\bar{y}+z) \quad (4.19)$$

poiché $0, 1, 4$ e 6 sono le codifiche in base due di 000, 001, 100 e 110.
La realizzazione operativa delle funzioni (4.17) e (4.19) mediante porte AND, OR, NOT è basata sui circuiti di figura 4.24. Poiché ogni porta AND, OR a n ingressi richiede $n-1$ porte a 2 ingressi, per il circuito di figura 4.24a



(a) Circuito derivante dai *minterm* relativi alla funzione di figura 4.21

(b) Circuito derivante dai *maxterm* relativi alla funzione di figura 4.23

Figura 4.24: Porte NOT, AND e OR necessarie a realizzare la funzione di figura 4.20a secondo *minterm* e *maxterm*

servono $2 \cdot 4 = 8$ porte AND e 3 porte OR a 2 ingressi; per l'altro circuito di figura 4.24b servono $2 \cdot 4 = 8$ porte OR e 3 porte AND a 2 ingressi; in entrambi i casi servono dunque 11 porte a 2 ingressi.
Naturalmente le espressioni (4.17) (somma di prodotti) e (4.19) (prodotto di somme) rappresentano la stessa funzione, come si può verificare facilmente. Prendiamo la somma dei prodotti e cerchiamo di semplificarla usando gli assiomi e i teoremi introdotti in precedenza nelle sezioni 4.2.1 e 4.2.2

$$\bar{x}y\bar{z} + \bar{x}yz + x\bar{y}z + xyz \overset{(A6)}{=} \bar{x}y(\bar{z}+z) + xz(\bar{y}+y) \overset{(A7)}{=} \bar{x}y\cdot 1 + xz\cdot 1 \overset{(A3)}{=} \bar{x}y + xz \quad (4.20)$$

Se invece prendiamo il prodotto di somme si ottiene

$$(x+y+z)\cdot(x+y+\bar{z})\cdot(\bar{x}+y+z)\cdot(\bar{x}+\bar{y}+z) \overset{(T9)}{=} (x+y)\cdot(\bar{x}+z) \overset{(T10)}{=} \bar{x}y + xz \quad (4.21)$$

A seguito della semplificazione la complessità del circuito associato alla funzione si riduce drasticamente, come si può vedere dalla figura 4.25; servono in tutto 3 porte a due ingressi al posto di 11. Le formule (4.17) e (4.19), che portano entrambe alla forma semplificata $\bar{x}y + xz$ per la funzione appena analizzata, aprono il problema

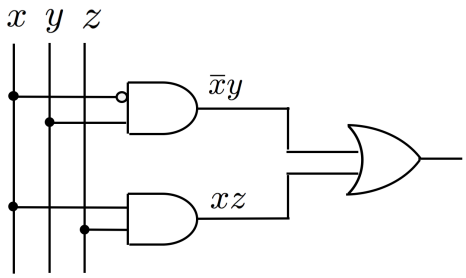


Figura 4.25: Circuito semplificato equivalente ai circuiti di figura 4.24

della ricerca della forma algebrica minima, che consenta cioè di realizzare la funzione usando il minimo numero possibile di porte logiche. Tanto per inquadrare la questione, si tenga conto che la (4.17) richiederebbe quattro porte AND con tre ingressi e una porta OR a quattro ingressi, più tutti i NOT che servono alla complementazione delle varie variabili; nel caso della (4.19) servono invece quattro porte OR con tre ingressi e una porta AND a quattro ingressi; se invece usiamo la funzione semplificata bastano due AND e un OR a due ingressi. Affronteremo nel seguito il problema della minimazione delle espressioni.
Si osservi inoltre che è sempre possibile passare dal prodotto di somme (4.21) alla somma di prodotti (4.20) e viceversa, sfruttando il T7 di De Morgan a partire dalla doppia negazione T3

$$\overline{\overline{(x+y+z)\cdot(x+y+\overline{z})\cdot(\overline{x}+y+z)\cdot(\overline{x}+\overline{y}+z)}} \overset{(T7)}{=} \overline{\overline{(x+y+z)}+\overline{(x+y+\overline{z})}+\overline{(\overline{x}+y+z)}+\overline{(\overline{x}+\overline{y}+z)}}$$
$$\overset{(T7)}{=} \overline{\overline{x}\,\overline{y}\,\overline{z}+\overline{x}\,\overline{y}\,z+x\,\overline{y}\,\overline{z}\;+\;x\,y\,\overline{z}}$$

La semplificazione si può a questo punto concludere nel modo seguente

$$\overline{\overline{x}\,\overline{y}\,\overline{z}+\overline{x}\,\overline{y}\,z+x\,\overline{y}\,\overline{z}\;+\;x\,y\,\overline{z}} \overset{(A6)}{=} \overline{\overline{x}\,\overline{y}(z+\overline{z})+x\,\overline{z}(y+\overline{y})} \overset{(A7)}{=} \overline{\overline{x}\,\overline{y}+x\,\overline{z}} \overset{(T7)}{=} \overline{\overline{x}\,\overline{y}}\cdot\overline{x\,\overline{z}}$$
$$\overset{(T7)}{=} (x+y)(\overline{x}+z) \overset{(T10)}{=} \overline{x}y+xz \tag{4.22}$$

In questo caso il passaggio ha richiesto molti calcoli in più, e non è stato conveniente; tuttavia la tecnica può soccorrere in qualche situazione particolare.
Quanto detto finora conferma nuovamente che ogni funzione Booleana $f(x_1, x_2, ..., x_n)$ può essere espressa indifferentemente o come somma di prodotti (OR di AND) o come prodotto di somme (AND di OR). Il teorema T7 di De Morgan costituisce lo strumento di connessione tra le due forme canoniche, consentendo di passare dall'una all'altra. La giustificazione formale è data dal fatto che possiamo esprimere la funzione f come la negazione di una certa funzione φ, espressa come somma dei termini minimi che corrispondono agli zeri della funzione f; in tal caso si ha $\varphi_i = \overline{\mu_i}$ e si ottiene

$$\overline{\varphi} = \overline{\sum_{i=0}^{2^n-1} \varphi_i m_i} = \overline{\sum_{i:\mu_i=0} \varphi_i m_i} = \prod_{i=0}^{2^n-1} (\overline{\varphi_i} + \overline{m_i}) = \prod_{i=0}^{2^n-1} (\mu_i + M_i) = f$$

nella quale la terza uguaglianza deriva dall'applicazione del teorema di De Morgan.
Per chiarire il procedimento possiamo prendere nuovamente come esempio la funzione di figura 4.21 studiata precedentemente

		x	y	z	f		φ	
m_0	$\overline{x}\overline{y}\overline{z}$	0	0	0	0	μ_0	1	φ_0
m_1	$\overline{x}\overline{y}z$	0	0	1	0	μ_1	1	φ_1
m_2	$\overline{x}y\overline{z}$	0	1	0	1	μ_2	0	φ_2
m_3	$\overline{x}yz$	0	1	1	1	μ_3	0	φ_3
m_4	$x\overline{y}\overline{z}$	1	0	0	0	μ_4	1	φ_4
m_5	$x\overline{y}z$	1	0	1	1	μ_5	0	φ_5
m_6	$xy\overline{z}$	1	1	0	0	μ_6	1	φ_6
m_7	xyz	1	1	1	1	μ_7	0	φ_7

Figura 4.26: La funzione di figura 4.21 con i valori della φ_i

Si ricava allora

$$f = \sum_{i:\mu_i=1} \mu_i m_i = \mu_2 m_2 + \mu_3 m_3 + \mu_5 m_5 + \mu_7 m_7 = \overline{x}y\overline{z} + \overline{x}yz + x\overline{y}z + xyz \tag{4.23}$$

$$\overline{\varphi} = \overline{\sum_{i:\mu_i=0} \varphi_i m_i} = \overline{\varphi_0 m_0 + \varphi_1 m_1 + \varphi_4 m_4 + \varphi_6 m_6} = \overline{\overline{x}\,\overline{y}\,\overline{z} + \overline{x}\,\overline{y}\,z + x\,\overline{y}\,\overline{z} \;+\; x\,y\,\overline{z}} \tag{4.24}$$

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

dove la 4.23 è espressa mediante *minterm*, mentre la 4.24 è espressa mediante *maxterm*. Siamo dunque passati dalla somma dei prodotti al prodotto delle somme.

4.3.5 Interpretazione circuitale

Come già accennato in precedenza, la prima applicazione circuitale dell'Algebra Booleana si deve a Shannon, che la usò nella progettazione di circuiti complessi per la commutazione a contatti. Questo tipo di applicazione è ancora largamente usata, anche se i contatti dei relè sono oggi sostituiti da dispositivi a stato solido (SCR, TRIAC). Tuttavia l'importanza dell'Algebra Booleana è legata soprattutto all'impiego nell'ambito dei circuiti logici, il cui peso è oggidì preponderante.
Nell'interpretazione di Shannon le costanti logiche 0 e 1 indicano rispettivamente un circuito aperto o uno chiuso, mentre le variabili indicano il contatto di un interruttore o di un relè. Con i simboli x e $\overline{x}$ si indicano due contatti azionati contemporaneamente, ma sempre tali che quando uno è aperto, l'altro è chiuso e viceversa. Si consideri ora un assieme di contatti $x_1, x_2, \cdots, x_n$ in parallelo tra loro, come illustrato in figura 4.27

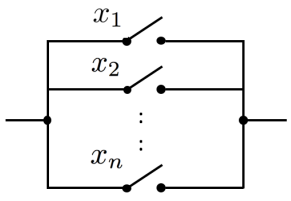


Figura 4.27: n contatti in parallelo realizzano la funzione Booleana $x_1 + x_2 + \cdots + x_n$

Il circuito presenterà continuità elettrica tra i punti a e b quando almeno uno dei contatti è chiuso. Ne consegue che la somma logica

$$x_1 + x_2 + ... + x_n$$

descrive, secondo l'analogia di Shannon, il comportamento elettrico di n contatti in parallelo.
Se invece i contatti $x_1, x_2, \cdots, x_n$ sono in serie tra loro, come illustrato in figura 4.28

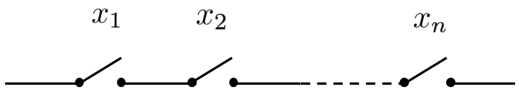


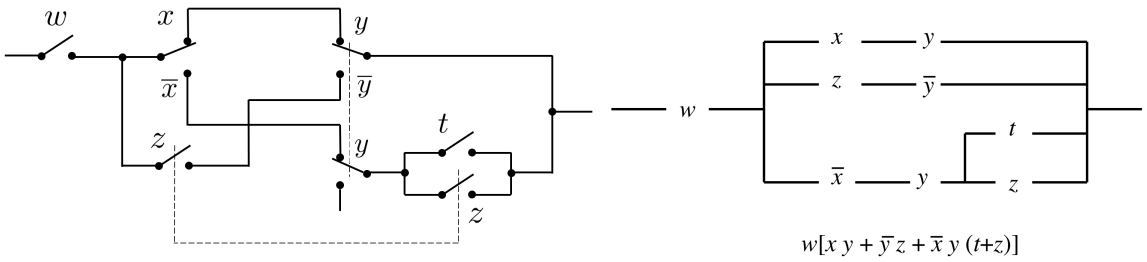
Figura 4.28: n contatti in serie realizzano la funzione Booleana $x_1 \cdot x_2 \cdot ... \cdot x_n$

il circuito presenterà continuità elettrica tra i punti a e b solo quando tutti i contatti sono chiusi. Ne consegue che il prodotto logico

$$x_1 \cdot x_2 \cdot ... \cdot x_n$$

descrive, secondo l'analogia di Shannon, il comportamento elettrico di n contatti in serie. L'associazione tra stato di apertura di un contatto e variabile Booleana apre alla possibilità di progettare reti complesse ricorrendo all'Algebra Booleana.

Esempio 4.1. Come primo esempio possiamo analizzare il circuito di figura 4.29a; esso è costituito da alcuni interruttori e due *deviatori*; un deviatore è un interruttore nel quale si aggiunge un contatto anche per la posizione di riposo, in modo che possa realizzare continuità elettrica lungo due percorsi alternativi. Per prima cosa dobbiamo assegnare le variabili Booleane, tenendo conto che le linee tratteggiate che collegano gli interruttori indicano che essi sono azionati contemporaneamente dallo stesso pulsante. Fatta questa operazione si perviene al circuito di



(a) Circuito complesso di commutazione

(b) Variabili Booleane associate al circuito di figura 4.29a

Figura 4.29: Circuito di commutazione e sua rappresentazione Booleana

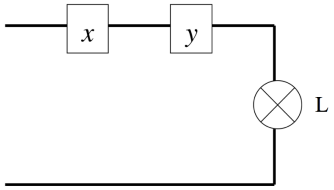
figura 4.29b. Esso è costituito dalla serie tra la variabile w e un blocco circuitale, costituito da tre rami in parallelo; ecco allora che avremo come espressione Booleana associata $w \cdot (p_1 + p_2 + p_3)$, dove p_1, p_2 e p_3 sono i tre rami in parallelo; p_1 è la serie di x e y, e dunque $p_1 = xy$, mentre $p_2 = z\overline{y}$. p_3 è invece la serie di tre elementi, l'ultimo dei quali è il parallelo tra t e z; dunque $p_3 = \overline{x}y(t + z)$. Mettendo tutto assieme si ottiene

$$w \cdot [xy + z\overline{y} + \overline{x}y(t + z)]$$

○

Questo appena fatto è un esempio di analisi di un circuito. E' evidente che a partire dall'espressione Booleana potremmo ricavare la tavola di verità e sapere qual è il comportamento del circuito. Ma l'Algebra Booleana eprime tutte le sue enormi potenzialità soprattutto quando si vuole progettare un circuito che esegua determinate funzioni, e cioè quando bisogna realizzare la *sintesi* di un circuito.

Esempio 4.2. Come esempio concreto, immaginiamo si voglia risolvere il seguente problema, di interesse molto pratico. In un appartamento c'è un corridoio; quando si entra si vuole accendere la luce dall'interruttore x prossimo all'ingresso, che supponiamo sia a destra nel disegno di figura 4.31); uscendo si vuole spegnere la luce dall'interruttore y prossimo all'uscita sulla sinistra. Per risolvere il problema, dobbiamo esprimere le condizioni



(a) Gli interruttori x e y comandano la luce di un corridoio

x	y	L
0	0	0
0	1	1
1	0	1
1	1	0

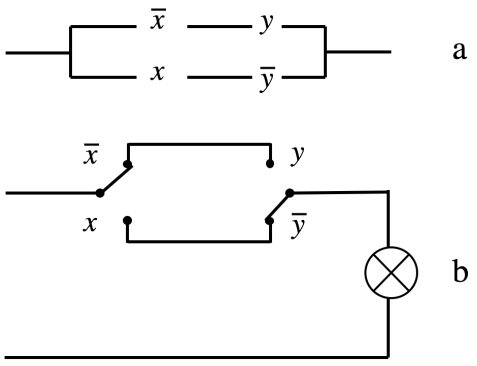
(b) Tavola di verità che esprime il funzionamento del circuito di figura 4.31

Figura 4.30: Circuito che comando una luce da due punti e relativa tavola di verità

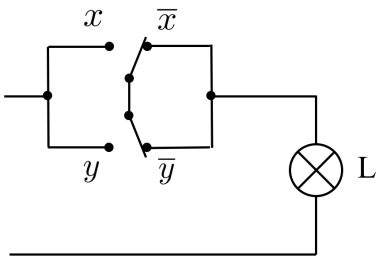
di funzionamento della lampada e degli interruttori ricorrendo agli strumenti dell'Algebra Booleana. Ciascuno dei due interruttori può stare in una sola tra due posizioni, che associamo alle due variabili 0 e 1. Anche la luce L può essere spenta (0) o accesa (1). Ecco allora che dobbiamo costruire una tavola di verità con due variabili d'ingresso (x e y) e una variabile L che assumerà dei valori che sono *funzione* di x e y; abbiamo insomma una funzione Booleana $f : \mathbf{2^2} \to \mathbf{2}$.
Supponiamo che, entrando in corridoio con la luce ancora spenta, entrambi gli interruttori siano (p.es) nello stato 0. A questa configurazione d'ingresso corrisponde $L = 0$. Se entriamo in corridoio da sinistra, all'atto di accendere la luce cambiamo la posizione dell'interruttore x, che passa da 0 a 1; in corrispondenza la luce deve accendersi, e dunque alla coppia 01 corrisponde $L = 1$. Uscendo dal corridoio a destra si muove l'interruttore y, che comanda lo spegnimento della luce; e dunque a 11 corrisponde $L = 0$. La configurazione che manca, 01, è quella che si crea quando si entra in corridoio da destra con la luce spenta e si muove l'interruttore; dunque a 01 corrisponde $L = 1$. Se mettiamo tutto nella tavola di verità otteniamo la funzione di figura 4.30b. Usando la prima forma canonica ricaviamo immediatamente l'espressione risolutiva

$$x\overline{y} + \overline{x}y \tag{4.25}$$

la quale, nella nuova interpretazione circuitale, corrisponde al parallelo di due interruttori posti in serie, così come rappresentato in figura 4.31a,a; la figura 4.31a,b mostra invece l'attuazione pratica, che usa due deviatori; essi realizzano la condizione che se x è chiuso, necessariamente $\overline{x}$ deve essere aperto e viceversa. Naturalmente il



(a) Gli interruttori x e y che risolvono il problema dell'accensione della luce nel corridoio



(b) Altra configurazione circuitale, basata sulla II forma canonica

Figura 4.31: Circuiti che risolvono il problema del comando di una luce da due punti diversi

problema può essere risolto anche mediante la seconda forma canonica, usando la quale si ottiene

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

Questa impostazione porta a un circuito un po' strano (si veda la figura 4.31b), che funziona altrettanto correttamente, ma che non si usa nella pratica dei collegamenti elettrici poiché richiede una cavo in più per la connessione tra i due deviatori.
L'equazione 4.25, risolutiva del problema, corrisponde anche al circuito di uno XOR, già visto nella figura 4.11b. In che rapporto sta allora quest'ultimo con il circuito 4.31a? La realizzazione mediante interruttori consente di incarnare la funzione Booleana direttamente al livello base dei circuiti di commutazione, in modo tale che ogni variabile Booleana sia immersa nel circuito; quella basata sulla porta XOR costituisce invece una sorta di interfaccia tra variabili d'ingresso, che determinano il comportamento del circuito, e la variabile d'uscita, che deve comandare la lampadina. Questo secondo approccio è concettualmente più elegante, ma soprattutto consente di svincolare il carico dalla rete logica. Se infatti il carico assorbe molta corrente, tutti gli interruttori della rete dovranno essere dimensionati di conseguenza; ciò potrebbe essere poco funzionale, visto che gli interruttori in grado di gestire correnti maggiori sono più grossi, costosi e più difficili da azionare. Usando invece un circuito simile a quello di figura 4.32 solo l'interruttore del relè deve essere dimensionato in modo tale da poter gestire la corrente del carico; gli altri due deviatori, relativi ai due comandi x e y, possono essere anche a bassa corrente. Si noti che nel circuito l'uscita della porta OR controlla direttamente la base di un transistor; quando l'uscita OR va a 1, il transistor entra in saturazione, la V_{CE} crolla al valore di saturazione e tutta la tensione V_{CC} di alimentazione va sul relè, che commutando chiude il circuito d'uscita e accende la lampadina. ◯

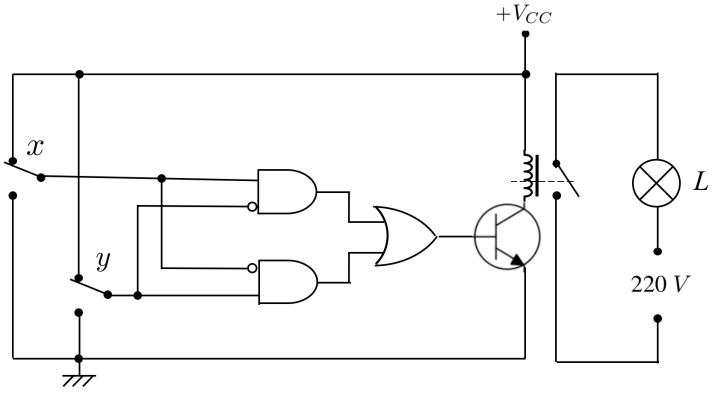


Figura 4.32: Controllo di una lampadina di potenza mediante una rete logica asservita a un transistor che controlla un relè

Esempio 4.3. Proviamo ora a complicare un po' il progetto, portando a tre il numero degli interruttori. Si vuol progettare un circuito capace di accendere o di spegnere una lampada mediante uno qualsiasi di tre interruttori indipendenti. Indicando al solito con $L = 1$ la condizione di lampada accesa, fissiamo come specifica che quando i tre interruttori sono aperti L valga 0. Chiudendo uno qualsiasi degli interruttori L dovrà assumere il valore 1, mentre tornerà al valore 0 azionando un qualsiasi altro interruttore. Infine essa assumerà nuovamente il valore 1 quando tutti i tre interruttori saranno chiusi. Anche questa situazione è molto frequente in pratica, per esempio in un ampio salone nel quale si vogliano avere più punti luce. Chiamando x, y, z le variabili logiche associate a ciascun interruttore si ricava la tavola di verità di figura 4.33

x	y	z	L
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Figura 4.33: Tavola di verità per il circuito a tre interruttori

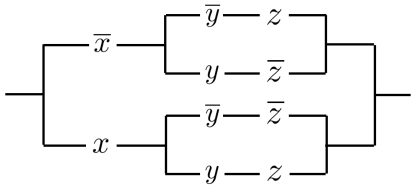
La sintesi del circuito mediante termini minimi fornisce

$$\overline{x}\,\overline{y}\,z + \overline{x}\,y\,\overline{z} + x\,\overline{y}\,\overline{z} + x\,y\,z$$

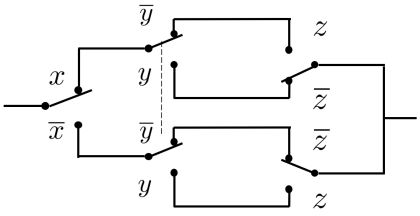
Prima di procedere alla realizzazione circuitale, è però opportuno semplificare il più possibile l'espressione, ottenendo

$$\overline{x}\,(\overline{y}\,z + y\,\overline{z}) + x(\overline{y}\,\overline{z} + y\,z)$$

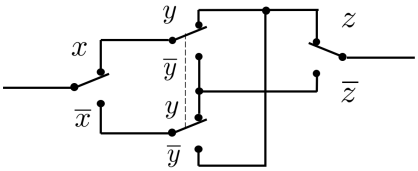
Il circuito che si ricava è rappresentato in figura 4.34.



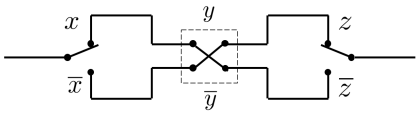
(a) Schema astratto per un controllo luce mediante tre interruttori



(b) Configurazione circuitale corrispondente



(c) Semplificazione circuitale del circuito 4.34b



(d) Attuazione pratica, mediante invertitore, del circuito 4.34c

Figura 4.34: Circuito con tre interruttori ottenuto da una sintesi basata sui termini minimi: realizzazione teorica e circuito commerciale, facente uso di un invertitore

Nella figura 4.34a viene rappresentato lo schema astratto, mentre nella 4.34b si illustra il circuito corrispondente. Si osservi tuttavia che i due interruttori associati alla variabile $\overline{z}$, quando sono chiusi (come in figura) portano allo stesso potenziale le variabili y e $\overline{y}$ della parte centrale del circuito; dunque i due rami più interni si possono fondere. Poiché lo stesso discorso vale anche per gli interruttori della variabile z, il risultato finale è che si semplifica il circuito e si risparmia un deviatore. Il circuito semplificato è rappresentato in figura 4.34c, mentre in figura 4.34d si mostra la realizzazione commerciale, che fa uso del cosiddetto *invertitore*[6], rappresentato nello schema con le connessioni ingresso-uscita incrociate; l'altra posizione è quella con le connessioni dirette. L'invertitore consente

[6]Tale invertitore nulla ha a che fare con l'altro invertitore introdotto nell'algebra Booleana

fra l'altro di generalizzare il problema a $n \geqslant 4$ interruttori; in tutti questi casi il circuito risolutivo è dato da due deviatori in testa alla catena e da $n-2$ invertitori in cascata nelle posizioni intermedie.
Se ora passiamo a una sintesi mediante termini massimi otteniamo

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

che con la semplificazione porta al circuito di figura 4.35. Anche in questo caso la soluzione basata sui termini massimi è meno vantaggiosa, poiché richiede cinque deviatori invece di quattro.

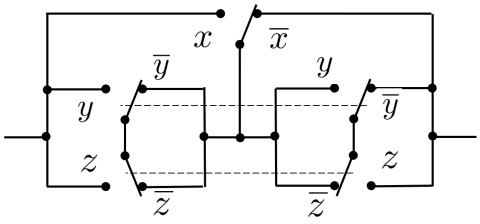


Figura 4.35: Circuito con tre interruttori ottenuto da una sintesi basata sui termini massimi

○

Al di là della straordinaria importanza applicativa del metodo formale basato sull'Algebra Booleana, è importante riflettere, soprattutto alla luce di quest'ultimo esempio, che molto spesso in ambito ingegneristico la teoria offre diverse soluzioni tra loro equivalenti; sta poi al progettista scegliere la migliore, tenuto conto della razionalità e semplicità delle soluzioni prospettate, dei costi, del numero di componenti impiegati ecc.

4.3.6 Semplificazione delle espressioni Booleane

Si è già visto, dagli esempi che precedono, che le forme canoniche non esauriscono le espressioni analitiche di una funzione; anzi, come per qualsiasi relazione algebrica, anche quelle logiche possono essere trasformate in un certo numero di espressioni formalmente diverse, ma equivalenti dal punto di vista matematico. Ad esempio

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

Si diranno *equivalenti* due funzioni che abbiano la stessa tavola di verità, forma semplificata di una funzione ogni sua espressione non canonica, forma minima quella in cui ogni variabile, diretta o negata che sia, compare il minor numero di volte.
Espressioni semplificate si possono ottenere applicando le relazioni fondamentali dell'algebra Booleana alle espressioni canoniche, ma questa strada richiede una notevole pratica, si applica facilmente solo a funzioni di un limitato numero di variabili e non dà alcuna garanzia di pervenire effettivamente alla forma minima, dato il carattere artigianale del procedimento. Si semplifichi per esempio la funzione:

$$f = x\,y\,z\,w + x\,y\,z\,\overline{w} + x\,y\,\overline{z}\,w + \overline{x}\,y\,z\,w + x\,y\,\overline{z}\,\overline{w}$$

Per la proprietà distributiva il primo termine si può semplificare con il secondo, mentre il terzo termine si può semplificare con il quinto, ottenendo:

$$\begin{aligned} f &= x\,y\,z\,(w+\overline{w}) + x\,y\,\overline{z}\,(w+\overline{w}) + \overline{x}\,y\,z\,w = x\,y\,z\ + x\,y\,\overline{z}\ + \overline{x}\,y\,z\,w \\ &= x\,y + \overline{x}\,y\,z\,w = y\,(x + \overline{x}\,z\,w) = y\,(x+z\,w) \end{aligned}$$

nella quale l'ultima uguaglianza deriva dal secondo teorema dell'assorbimento T5.
Il termine

$$x\ y\ z = x\ y\ z\ w + x\ y\ z\ \overline{w}$$

viene detto *implicante*, in quanto implica i termini $x\ y\ z\ w$ e $x\ y\ z\ \overline{w}$, nel senso che vale 1 quando valgono 1 o l'uno o l'altro dei termini minimi implicati.

La semplificazione prodotta, nonostante sia efficace, deriva da una procedura arbitraria e non sistematica. Nel seguito studieremo due metodi procedurali per ottenere una semplificazione efficace.

Le mappe di Karnaugh

Il metodo proposto da *Karnaugh* è un metodo grafico di semplificazione che permette di ottenere molto semplicemente la forma minima di una funzione espressa come somma di prodotti *minterm*, facendo ricorso a particolari mappe di rappresentazione. Quale limitazione si ha che, sebbene il metodo sia concettualmente applicabile a funzioni di qualsiasi numero di variabili, esso diviene difficoltoso già per $5-6$ variabili.
Le mappe di Karnaugh, che possono essere considerate un ulteriore metodo di rappresentazione di una funzione logica, consistono in matrici m righe e k colonne, in cui $m = 2^i, k = 2^j$ per qualche i, j, col vincolo che $2^i \cdot 2^j = 2^n$ è pari al numero di elementi della matrice, e n è il numero delle variabili. Di conseguenza esse hanno 4 elementi per le funzioni di due variabili, 8 per quelle di tre variabili, 16 per quelle di quattro e così via. Ogni elemento della matrice rappresenta un termine minimo, che entra in un'espressione *minterm* di somma di prodotti. Nella figura 4.36 sono riprodotte le strutture delle mappe per $2, 3$ e 4 variabili.

$y \backslash x$	0	1
0	$\overline{x}\overline{y}$	$x\overline{y}$
1	$\overline{x}y$	xy

$z \backslash xy$	00	01	11	10
0	$\overline{x}\overline{y}\overline{z}$	$\overline{x}y\overline{z}$	$xy\overline{z}$	$x\overline{y}\overline{z}$
1	$\overline{x}\overline{y}z$	$\overline{x}yz$	xyz	$x\overline{y}z$

$zw \backslash xy$	00	01	11	10
00	$\overline{x}\overline{y}\overline{z}\overline{w}$	$\overline{x}y\overline{z}\overline{w}$	$xy\overline{z}\overline{w}$	$x\overline{y}\overline{z}\overline{w}$
01	$\overline{x}\overline{y}\overline{z}w$	$\overline{x}y\overline{z}w$	$xy\overline{z}w$	$x\overline{y}\overline{z}w$
11	$\overline{x}\overline{y}zw$	$\overline{x}yzw$	$xyzw$	$x\overline{y}zw$
10	$\overline{x}\overline{y}z\overline{w}$	$\overline{x}yz\overline{w}$	$xyz\overline{w}$	$x\overline{y}z\overline{w}$

Figura 4.36: Mappe di Karnaugh per 2, 3 e 4 variabili

Prendiamo ora come esempio il caso della funzione a 3 variabili descritta dalla tavola di verità di figura 4.37.

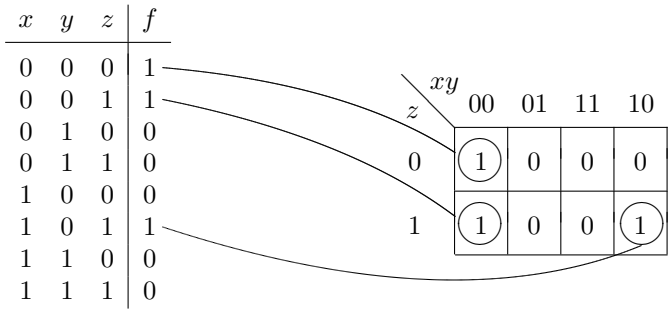


Figura 4.37: Costruzione di una mappa di Karnaugh per una funzione a 3 variabili

Usando la prima forma canonica *minterm* si ottiene

$$\overline{x}\overline{y}\overline{z} + \overline{x}\overline{y}z + x\overline{y}z$$

Nella mappa mettiamo un 1 in corrispondenza delle terne per le quali la funzione vale 1 e uno 0 in corrispondenza delle terne per le quali la funzione vale 0. L'assegnazione delle coordinate delle caselle della tabella dev'essere tale che passando da ciascuna casella a una adiacente, sia in senso orizzontale che verticale, vari il valore di una sola variabile. Si noti che devono essere considerate adiacenti anche le caselle terminali, come se la mappa fosse chiusa circolarmente su sè stessa, sia in senso orizzontale che verticale.
Ricapitolando si può dire che ciascun elemento della matrice corrisponde a un termine minimo di n variabili, e che la rappresentazione di una qualsiasi funzione di n variabili si ottiene contrassegnando con 1 o con 0 le posizioni corrispondenti ai termini minimi da cui la funzione è composta.
Una funzione riportata sulla mappa di Karnaugh può essere semplificata osservando che due caselle adiacenti, sia in senso orizzontale che verticale, differiscono per il valore di una sola variabile, che in una delle due caselle apparirà come variabile affermata, nell'altra come negata; si ricava dunque

$$fx + f\overline{x} = f(x + \overline{x}) = f$$

Per spiegare la procedura facciamo riferimento alla figura 4.38

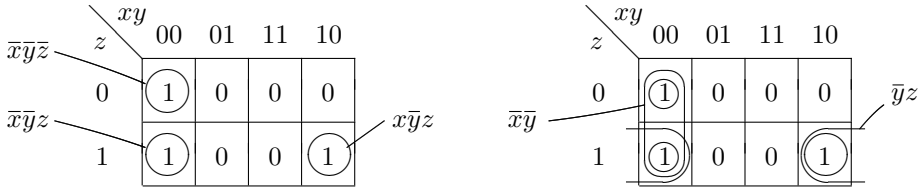


Figura 4.38: Semplificazione di una funzione mediante mappa di Karnaugh per una funzione a 3 variabili

Le caselle relative ai termini $\overline{x}\overline{y}\overline{z}$ e $\overline{x}\overline{y}z$ sono adiacenti, e la semplificazione è

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

Sono però adiacenti anche le caselle $\overline{x}\overline{y}z$ e $x\overline{y}z$, che si semplificano come

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

La funzione semplificata si ottiene allora come somma degli implicanti necessari e sufficienti a coprire tutti i *minterm* della funzione; nel caso di cui sopra si ha

$$f = \overline{x}\overline{y} + \overline{y}z$$

Facciamo ora un altro esempio, riconsiderando la funzione a tre variabili di figura 4.21, descritta dall'equazione (4.17) che riportiamo qua sotto per comodità del lettore

$$f(x,y,z) = \bar{x}y\bar{z} + \bar{x}yz + x\bar{y}z + xyz \tag{4.26}$$

x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

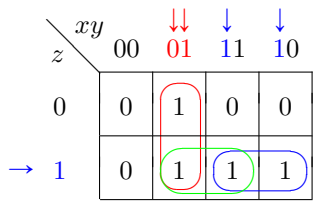


Figura 4.39: Mappa di Karnaugh per la funzione a 3 variabili descritta dalla funzione di figura 4.21

Se chiamiamo *sottocubo* il ricoprimento associato a un implicante, la regola pratica per individuare l'implicante a esso associato è che esso è costituito dal prodotto dalle variabili che rimangono costanti su 0 o su 1 all'interno del sottocubo stesso; nel primo dei due casi di figura 4.39 esse sono $x = 0$ e $y = 1$ (frecce rosse), e dunque $\bar{x}y$; nel secondo caso sono $x = 1$ e $z = 1$ (frecce blu), e dunque xz. Poiché i sottocubi devono avere 2^i elementi, per qualche valore di i, si possono costruire solo i due sottocubi con $i = 1$ che prendono dentro gli implicanti $\bar{x}y$ e xz; si osservi che il sottocubo associato all'implicante yz (in verde in figura), pur legittimo, sarebbe superfluo, poiché entrambi i suoi termini minimi $\bar{x}yz$ e xyz sono già compresi negli altri due sottocubi; d'altra parte già sappiamo, dal III teorema dell'assorbimento T6, che

$$\bar{x}y + xz + yz = \bar{x}y + xz \tag{4.27}$$

Facciamo ora un altro esempio, modificando la funzione nel modo seguente

$$f(x,y,z) = \bar{x}y\bar{z} + \bar{x}yz + xy\bar{z} + xyz \tag{4.28}$$

x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

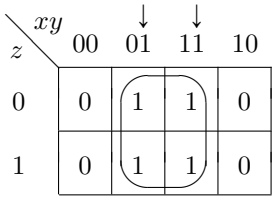


Figura 4.40: Altro esempio di mappa di Karnaugh per una funzione a 3 variabili

In questo caso si riesce a formare un sottocubo di 2^2 termini minimi, che hanno come implicante

$$\bar{x}y\bar{z} + \bar{x}yz + xy\bar{z} + xyz = y(\bar{x}\bar{z} + \bar{x}z + x\bar{z} + xz) = y(\bar{x}(\bar{z}+z) + x(\bar{z}+z)) = y(\bar{x}+x) = y$$

Si osservi dalla figura 4.43 che $y = 1$ è in effetti l'unica variabile a rimanere costante.
Se ne deduce che un implicante di $n-1$ variabili implica i termini minimi associati a 2 caselle adiacenti, un implicante di $n-2$ variabili implica i termini minimi associati a 2^2 caselle adiacenti (appartenenti alla stessa riga o alla stessa colonna o raccolti attorno allo stesso vertice) e in generale un implicante di $n-i$ variabili implica i termini minimi associati a un sottocubo di 2^i caselle adiacenti. Un implicante si dice *primo* se corrisponde a un sottocubo non completamente coperto da altri sottocubi, *massimale* se non è possibile aumentare ulteriormente la sua dimensione; si definisce infine *copertura* della funzione un insieme di sottocubi tali da coprire tutti gli 1 della funzione. La minimizzazione della funzione si ricava a partire da una copertura minima, cioè una copertura formata dal più piccolo insieme di sottocubi primi massimali.

Facciamo ora un esempio con 4 variabili; la funzione

$$y = \overline{x}\,\overline{y}\,\overline{z}\,\overline{w} + \overline{x}\,y\,z\,w + \overline{x}\,\overline{y}\,z\,w + \overline{x}\,y\,z\,\overline{w} \tag{4.29}$$

è rappresentata dalla mappa di Karnaugh di figura 4.41.

zw \ xy	00	01	11	10
00	(1)	0	0	0
01	0	0	0	0
11	1	1	0	0
10	0	1	0	0

Figura 4.41: Mappa di Karnaugh per la funzione (4.29) di 4 variabili

I tre implicanti primi sono $\overline{x}\overline{y}\overline{z}\overline{w}$, $\overline{x}zw$ e $\overline{x}yz$; la loro somma rappresenta la funzione in forma semplificata.

Consideriamo ora la seguente funzione:

$$f = \overline{x}\,\overline{y}\,\overline{z}\,\overline{w} + \overline{x}\,\overline{y}\,\overline{z}\,w + \overline{x}\,\overline{y}\,z\,w + \overline{x}\,\overline{y}\,z\,\overline{w} + \overline{x}\,y\,\overline{z}\,w + \overline{x}\,y\,z\,w + x\,\overline{y}\,\overline{z}\,\overline{w} + x\,\overline{y}\,\overline{z}\,w + x\,\overline{y}\,z\,w + x\,\overline{y}\,z\,\overline{w} \tag{4.30}$$

In figura 4.42 è riportata la mappa di Karnaugh relativa ed è anche indicata la copertura minima di implicanti primi massimali.

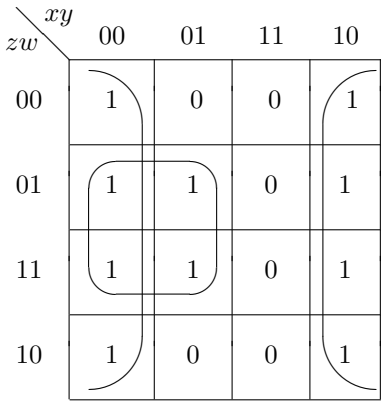


Figura 4.42: Mappa di Karnaugh per la funzione (4.30) di 4 variabili

La funzione semplificata risulta

$$y = \overline{x}\, w + \overline{y} \tag{4.31}$$

Se anziché considerare ciascuna casella della mappa come rappresentativa di un termine minimo la si considera rappresentativa di un termine massimo, la mappa può essere usata anche per costruire la funzione semplificata secondo la II forma canonica basata sul prodotto di somme. In tal caso i sottocubi e la relativa copertura minima vanno costruiti sugli 0. Ciascun sottocubo sarà l'implicante dei relativi termini massimi e verrà rappresentato in forma simbolica dalla somma logica delle variabili che rimangono costanti sul sottoinsieme, dirette se le relative coordinate valgono 0, negate se valgono 1. Tutto ciò deriva dalla circostanza che

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

Come esempio riprendiamo la funzione 4.26 e la relativa mappa

x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

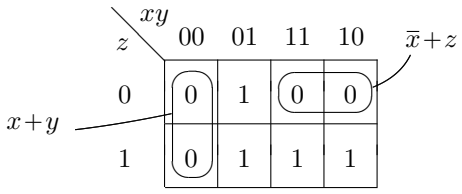


Figura 4.43: Mappa di Karnaugh usata per la semplificazione dei *maxterm*

La funzione semplificata che ne deriva è

$$(x + y)(\overline{x} + z) = xz + \overline{x}y$$

che corrisponde alla funzione 4.27 già trovata in precedenza basandoci sulla mappa di Karnaugh associata ai *minterm*.

Il metodo delle mappe di Karnaugh può essere applicato, come già accennato, anche a funzioni di 5 o 6 variabili. In tal caso non si potrà più realizzare una mappa piana, in cui ogni casella sia adiacente a caselle le cui coordinate differiscano per un'unica variabile. La mappa per cinque variabili viene pertanto realizzata mediante due mappe da quattro variabili, una associata al valore 0 della quinta variabile, l'altra al valore 1. Le adiacenze vanno quindi ricercate anche tra caselle occupanti posizioni omologhe sulle due mappe.
Si voglia ad esempio semplificare la seguente funzione

$$\begin{aligned} f = {} & x\,\overline{y}\,z\,w\,r + x\,\overline{y}\,z\,w\,\overline{r} + x\,\overline{y}\,z\,\overline{w}\,r + x\,\overline{y}\,z\,\overline{w}\,\overline{r} + \\ & + x\,\overline{y}\,\overline{z}\,w\,r + x\,\overline{y}\,\overline{z}\,w\,\overline{r} + x\,\overline{y}\,\overline{z}\,\overline{w}\,r + x\,\overline{y}\,\overline{z}\,\overline{w}\,\overline{r} + \\ & + \overline{x}\,y\,z\,w\,r + x\,y\,z\,w\,r + \overline{x}\,\overline{y}\,z\,w\,r + \overline{x}\,y\,\overline{z}\,\overline{w}\,r \end{aligned} \tag{4.32}$$

Le due mappe da 4 variabili, che si ottengono, una per $r = 0$ e una per $r = 1$ sono le seguenti

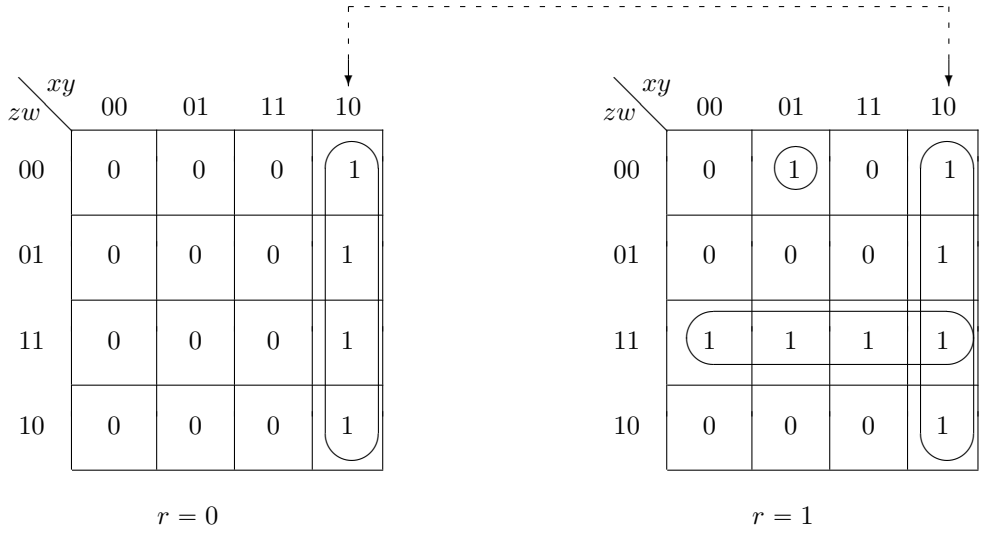


Figura 4.44: Uso della mappa di Karnaugh per la funzione (4.32) di 5 variabili

La funzione semplificata risulta

$$f = x\,\overline{y} + z\,w\,r + \overline{x}\,y\overline{z}\,\overline{w}\,r \tag{4.33}$$

Se le variabili fossero 6, p.es. x, y, z, w, r, t, sarebbe necessario usare quattro mappe nelle quattro variabili x, y, z, w, una per ogni coppia possibile di valori associati alle variabili r, t.

Le condizioni non specificate

Nella sintesi di una funzione logica di n variabili si può presentare il caso in cui per k configurazioni delle variabili di ingresso la funzione vale 1, per m configurazioni vale 0, ma $k + m < 2^n$. Le restanti $2^n - (m + k)$ configurazioni vengono dette *condizioni non specificate* (o anche *d.c.c.*, acronimo di *don't care condition*).
In pratica questa situazione si verifica ogni qual volta in un circuito certe configurazioni di ingresso sono fisicamente impossibili, o rendono privo di significato il valore dell'uscita. Da un punto di vista strettamente analitico ciò accade quando una funzione F è funzione delle variabili $\phi_1, \phi_2, \ldots, \phi_m$, ognuna delle quali è a sua volta funzione

delle variabili $x_1, x_2, ..., x_n$, cioè

$$F = f(\phi_1, \phi_2, \ldots, \phi_m) \qquad \phi_i = f_i(x_1, x_2, ..., x_n) \qquad i = 1, 2, ..., m$$

Le condizioni non specificate si ricavano dalle tavole di verità delle ϕ_i , e sono tutte e sole le configurazioni delle ϕ_i che non compaiono in tali tabelle. Sia ad esempio:

$$F = f(\phi_1, \phi_2, \phi_3) \qquad \begin{array}{l} \phi_1 = x\,(y\,z + \overline{y}\,\overline{z}) \\ \phi_2 = w\,(y\,\overline{z} + \overline{y}\,z) \\ \phi_3 = x\,y\,z + \overline{x}\,\overline{y}\,\overline{z} \end{array} \tag{4.34}$$

La tavola di verità delle ϕ_i è riportata nella figura 4.45. Nella seconda tavola della stessa figura sono riportati anche

x	y	z	w	ϕ_1	ϕ_2	ϕ_3
0	0	0	0	0	0	1
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	0	1	1	0	1	0
0	1	0	0	0	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	0	0
1	0	1	1	0	1	0
1	1	0	0	0	0	0
1	1	0	1	0	1	0
1	1	1	0	1	0	1
1	1	1	1	1	0	1

ϕ_1	ϕ_2	ϕ_3	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	–
1	0	0	1
1	0	1	0
1	1	0	–
1	1	1	–

Figura 4.45: Le tavole di verità delle funzioni ϕ_1, ϕ_2, ϕ_3, descritte dalle equazioni 4.34, e quella della funzione F

i valori assunti dalla funzione F; si vede che le terne di possibili valori ϕ_1, ϕ_2, ϕ_3 sono $000, 001, 010, 100, 101$, mentre le configurazioni rimanenti $011, 110, 111$ non compaiono. Ne consegue che la tavola di verità della F conterrà condizioni non specificate in corrispondenza di queste configurazioni d'ingresso.
Sulla tavola di verità le condizioni non specificate vengono indicate con un trattino, nella forma canonica raccogliendo in parentesi i termini minimi corrispondenti, sulle mappe di Karnaugh contrassegnando con il simbolo $\times$ la casella corrispondente a ciascuna condizione non specificata.
Le condizioni non specificate possono venir sfruttate nelle semplificazioni, in modo da pervenire ad espressioni minime più semplici. Se si opera con le mappe di Karnaugh, le semplificazioni vanno ancora fatte in modo da coprire tutte le caselle contrassegnate con un 1, ma se serve si possono aggregare anche le caselle associate a condizioni non specificate, che possiamo contrassegnare con $\times$; in pratica si tratta di utilizzare le condizioni non specificate per allargare al massimo i sottocubi di copertura della funzione, assegnando a ciascuna di loro il valore 1 o 0 a seconda che torni o meno utile per ottenere sottocubi più ampi. Nell'esempio di sopra si ottiene

ϕ_3 \ $\phi_1\phi_2$	00	01	11	10
0	1	1	×	1
1	0	×	×	0

Vediamo un altro esempio; si prenda la seguente funzione:

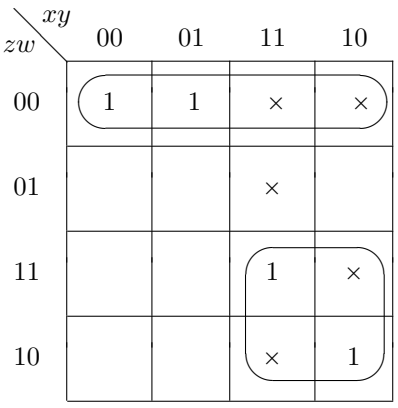


Figura 4.46: Semplificazione di una funzione usando le condizioni non specificate

Senza considerare le condizioni d.c.c. si otterrebbe:

$$F = \overline{x}\,\overline{z}\,\overline{w} + x\,y\,z\,w + x\,\overline{y}\,z\,\overline{w}$$

mentre tenendo conto anche di queste ultime si ha:

$$F = \overline{z}\,\overline{w} + x\,z$$

Il metodo tabellare di Quine - Mc Cluskey

Il metodo di *Quine - Mc Cluskey* è un procedimento tabellare che consente di ottenere la forma minima come somma di prodotti per qualsiasi funzione logica. Esso si basa sulla relazione:

$$f\,x + f\,\overline{x} = f$$

già usata in precedenza per le mappe di Karnaugh; solo che ora essa viene applicata in modo sistematico a tutti i termini minimi della funzione. Per capirne la logica riprendiamo la solita funzione di figura 4.20a:

		x	y	z	f	
0	$\overline{x}\overline{y}\overline{z}$	0	0	0	0	
1	$\overline{x}\overline{y}z$	0	0	1	0	
2	$\overline{x}y\overline{z}$	0	1	0	1	*
3	$\overline{x}yz$	0	1	1	1	*
4	$x\overline{y}\overline{z}$	1	0	0	0	
5	$x\overline{y}z$	1	0	1	1	*
6	$xy\overline{z}$	1	1	0	0	
7	xyz	1	1	1	1	*

Livello			x	y	z	f	
0	0	$\overline{x}\overline{y}\overline{z}$	0	0	0	0	
1	1	$\overline{x}\overline{y}z$	0	0	1	0	
	2	$\overline{x}y\overline{z}$	0	1	0	1	*
	4	$x\overline{y}\overline{z}$	1	0	0	0	
2	3	$\overline{x}yz$	0	1	1	1	*
	5	$x\overline{y}z$	1	0	1	1	*
	6	$xy\overline{z}$	1	1	0	0	
3	7	xyz	1	1	1	1	*

Livello			x	y	z	f	
1	2	$\overline{x}y\overline{z}$	0	1	0	1	*
	3	$\overline{x}yz$	0	1	1	1	*
2	5	$x\overline{y}z$	1	0	1	1	*
3	7	xyz	1	1	1	1	*

Figura 4.47: Trasformazione della tavola di verità della funzione di figura 4.20a per ottenere la tabella di Quine-Mc Cuskey

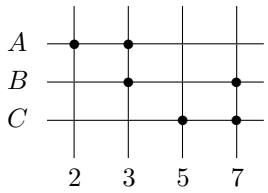
La semplificazione per via algebrica porta a

$$\overline{x}y\overline{z} + \overline{x}yz + x\overline{y}z + xyz = \overline{x}y(\overline{z} + z) + xz(\overline{y} + y) = \overline{x}y + xz \tag{4.35}$$

E' ben evidente che la struttura della semplificazione $\bar{x}y\bar{z} + \bar{x}yz = \bar{x}y(\bar{z} + z)$, che riguarda i termini minimi contrassegnati con $*$ in tabella, è del tipo $f\,\bar{z} + f\,z = f$ e riguarda due termini minimi che differiscono per la sola variabile z, che si trova diretta in uno dei due termini e negata nell'altro. La stessa cosa si può dire per i due termini minimi contrassegnati con $*$. Ordiniamo allora le righe della tabella in modo da mettere su livelli adiacenti le n-ple che differiscono per una sola variabile. Una possibilità è quella di procedere per peso crescente delle n-ple binarie associate ai termini minimi - dove per peso di un n-pla s'intende il numero di 1 in essa presenti - in modo che a ciascun peso corrisponda un livello; così facendo si ottiene la seconda tabella di figura 4.47. Se ora eliminiamo le righe che corrispondono a termini minimi per i quali la funzione vale 0 - che non rientrano nella somma di prodotti - otteniamo la terza tabella di figura 4.47, che corrisponde alla tabella di Quine-Mc Cuskey. A questo punto si può procedere con le semplificazioni, che generano una seconda tabella. Poichè $\bar{x}y\bar{z}$ si semplifica con $\bar{x}yz$ per ottenere $\bar{x}y$, su questa tabella segniamo con una lineetta la variabile z che è stata oggetto di semplificazione; segniamo inoltre a lato i termini minimi che sono coinvolti nella semplificazione, cioè $2-3$; viceversa la coppia $2-5$ non porta ad alcuna semplificazione. Esaurito il confronto tra i livelli 1 e 2, possiamo ora controllare le semplificazioni tra i livelli 2 e 3; in questo caso troviamo semplificazioni per entrambe le coppie, cioè $3-7$ e $5-7$. La tabella che si ottiene è la seguente:

		x	y	z	
	$2-3$	0	1	–	A
	$3-7$	–	1	1	B
	$5-7$	1	–	1	C

Poichè non è possibile fare altre semplificazioni ci si ferma, contrassegnando con delle lettere progressive gli implicanti che non si possono più semplificare. L'espressione finale deriva dalla somma degli implicanti coinvolti, cioè $f = A + B + C = \bar{x}y + yz + xz$. Si osservi tuttavia che l'espressione che otteniamo non è in forma minima, poiché come noto il termine yz si può eliminare per il terzo teorema dell'assorbimento. Per individuare il minimo numero di implicanti che implicano tutti i termini minimi dobbiamo costruire il seguente reticolo, disponendo sulle righe gli implicanti, sulle colonne i termini minimi, e ponendo un segno (p.es. un pallino) in corrispondenza dei termini minimi che formano ciascun implicante.



Procedendo per colonne osserviamo che il termine minimo 2 è coperto solo da A e dunque questo implicante è indispensabile; il termine minimo 5 è coperto solo da C e dunque anche questo implicante è indispensabile. L'implicante B non è invece necessario alla composizione della funzione, poiché i termini minimi a lui associati, 3 e 7, sono già coperti rispettivamente da A e da C. Dunque la funzione semplificata è $f = A + C = \bar{x}y + xz$.

La procedura completa per la semplificazione mediante il metodo tabellare di Quine-Mc Cluskey è dunque la seguente:

1. si esprimono i termini minimi che sono a 1 sostituendo ogni variabile diretta con 1 e ogni variabile negata con 0. Ad esempio il termine minimo $\bar{x}yz$ viene rappresentato con 011;
2. si ordinano tutti i termini minimi in gruppi aventi lo stesso peso, cioè lo stesso numero di 1; tali raggruppamenti vengono chiamati livelli;
3. si costruisce una tabella disponendo i livelli in ordine crescente e associando a ciascun termine minimo il corrispondente numero decimale. Ad esempio i termini minimi

$$\bar{x}\,\bar{y}\,\bar{z} \qquad \bar{x}\,\bar{y}\,z \qquad \bar{x}\,y\,\bar{z} \qquad \bar{x}\,y\,z \qquad x\,y\,z$$

generano la seguente tabella

	Livello	Numero	Termine minimo
$\surd$	0	0	000
$\surd$	1	1	001
$\surd$		2	010
$\surd$	2	3	011
$\surd$	3	7	111

4. a partire dal primo termine minimo del primo livello disponibile, si confrontano tutti i termini minimi del livello k con tutti quelli del livello $k+1$, semplificando tra loro i termini che differiscono per un solo bit. Si costruisce in tal modo una seconda tabella, nella quale le semplificazioni avvenute si indicano con una lineetta, mentre gli implicanti vengono contraddistinti con i numeri dei termini minimi che li hanno generati. Nella tabella si contrassegnano tutti i termini minimi che hanno dato luogo ad almeno una semplificazione, mentre i termini minimi che non hanno portato a semplificazioni vengono contrassegnati con lettere progressive dell'alfabeto. Riferendosi all'esempio riportato sopra, si ottiene la seguente tabella

$\surd$	$0,1$	$0\ 0-$
$\surd$	$0,2$	$0-0$
$\surd$	$1,3$	$0-1$
$\surd$	$2,3$	$0\ 1-$
A	$3,7$	$-1\ 1$

5. con lo stesso procedimento di prima, nella seconda tabella si confrontano tutti i termini del livello k con tutti quelli del livello $k+1$. Sono semplificabili tra loro i termini che differiscono per un solo bit e che siano già stati semplificati rispetto alla stessa variabile. Si costruisce in tal modo una terza tabella con le stesse modalità esposte per la costruzione della seconda tabella. Nell'esempio che si sta trattando si ha

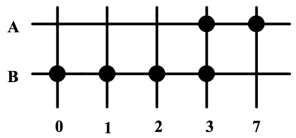
$$B \mid 0,1,2,3 \mid \quad 0\ -\ -$$

mentre nella seconda tabella il termine $3,7$ non dà luogo a semplificazioni e viene quindi contrassegnato con A.

6. si prosegue in modo analogo, con la costruzione di tabelle successive, finché non è più possibile eseguire semplificazioni. Tutti i termini che nelle successive tabelle non sono stati contrassegnati vengono chiamati *implicanti primi* e la loro somma logica realizza la funzione desiderata. Può però accadere che l'espressione minima di tale funzione si realizza però con un numero di implicanti inferiore a quello degli implicanti primi. Essa infatti si ricava come somma del minimo numero di implicanti primi con cui vengono implicati tutti i termini minimi della funzione. Nell'esempio che si sta esaminando gli implicanti primi sono

$$A = -\,1\,1 \qquad B = 0\ -\ -$$

7. la scelta più opportuna degli implicanti primi necessari, che può divenire complessa già con un numero di variabili relativamente ridotto, avviene mediante la costruzione di un semplice *reticolo*, avente i termini minimi sulle colonne e gli implicanti primi sulle righe. Su ogni riga, cioè in corrispondenza di ciascun implicante, si contrassegnano opportunamente i termini minimi implicati. Nell'esempio trattato si ottiene il seguente reticolo



Dall'esame del reticolo si individuano poi i termini minimi che sono implicati da un unico implicante primo. Ciascuno di essi diviene evidentemente essenziale nella realizzazione della funzione. Ogni implicante essenziale così individuato implica d'altra parte altri termini minimi che risultano automaticamente coperti e pertanto non vanno più considerati. E' semplice infine trovare, sia pure per tentativi, la copertura minima dei termini rimasti. Nell'esempio che si sta trattando ambedue gli implicanti primi A e B sono essenziali; l'espressione semplificata è allora

$$f = A + B = y\,z + \overline{x}$$

Esempio 4.4. Più significativo, soprattutto per quanto riguarda la realizzazione della copertura minima, è l'esempio seguente. Si voglia semplificare la funzione

$$f = \sum(1, 3, 4, 6, 7, 9, 10, 11, 12, 13, 14, 15)_m \tag{4.36}$$

nella quale la notazione binaria associata a ciascun numero intero rappresenta la codifica dei termini minimi. La divisione dei termini minimi in livelli e il riordinamento dei livelli dà luogo alla tabella di sinistra, mentre al centro e sulla destra ci sono le successive tabelle di semplificazione

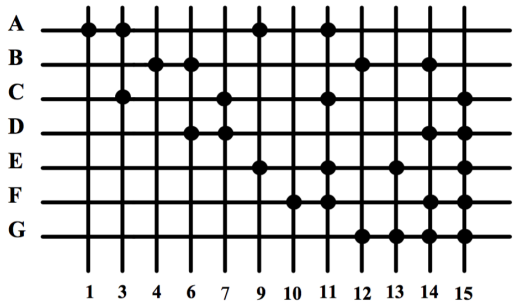
	Livello	Numero	Termine minimo
√	1	1	0001
√	1	4	0100
√	2	3	0011
√		6	0110
√	2	9	1001
√	2	10	1010
√	2	12	1100
√	3	7	0111
√	3	11	1011
√	3	13	1101
√	3	14	1110
√	4	15	1111

√	1, 3	0 0 − 1
√	1, 9	− 0 0 1
√	4, 6	0 1 − 0
√	4, 12	− 1 0 0
√	3, 7	0 − 1 1
√	3, 11	− 0 1 1
√	6, 7	0 1 1 −
√	6, 14	− 1 1 0
√	9, 11	1 0 − 1
√	9, 13	1 − 0 1
√	10, 11	1 0 1 −
√	10, 14	1 − 1 0
√	12, 13	1 1 0 −
√	12, 14	1 1 − 0
√	7, 15	− 1 1 1
√	11, 15	1 − 1 1
√	13, 15	1 1 − 1
√	14, 15	1 1 1 −

A	1, 3, 9, 11	− 0 − 1
B	4, 6, 12, 14	− 1 − 0
C	3, 7, 11, 15	− − 1 1
D	6, 7, 14, 15	− 1 1 −
E	9, 11, 13, 15	1 − − 1
F	10, 11, 14, 15	1 − 1 −
G	12, 13, 14, 15	1 1 − −

Il reticolo di semplificazione è invece rappresentato in figura 4.4.

Dall'esame del reticolo risulta che il termine minimo 1 è implicato solamente da A, quello 4 solo da B e quello 10 solo da F. Di conseguenza A, B, F sono implicanti primi essenziali. La loro scelta copre i termini minimi $1, 3, 4, 6, 9, 10, 11, 12, 14, 15$. Per coprire i rimanenti termini minimi 7 e 13 si può scegliere per il primo l'implicante C o quello D, per il secondo quello E o quello G.



Ci sono pertanto quattro realizzazioni equivalenti della funzione assegnata

$$
\begin{aligned}
f &= A + B + F + D + E = \overline{y}\, w + y\, \overline{w} + x\, z + y\, z + x\, w \\
f &= A + B + F + D + G = \overline{y}\, w + y\, \overline{w} + x\, z + y\, z + x\, y \\
f &= A + B + F + C + E = \overline{y}\, w + y\, \overline{w} + x\, z + z\, w + x\, w \\
f &= A + B + F + C + G = \overline{y}\, w + y\, \overline{w} + x\, z + z\, w + x\, y
\end{aligned}
\tag{4.37}
$$

Tali realizzazioni si comprendono non appena si valuti attentamente la mappa di Karnaugh 4.48 associata alla funzione 4.36

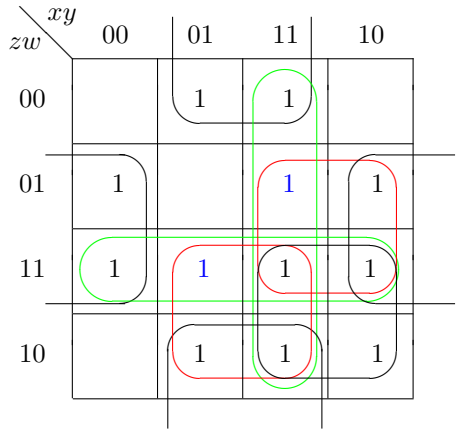


Figura 4.48: Mappa di Karnaugh

Si può vedere che i sottocubi neri sono necessari per coprire i corrispondenti termini minimi, e sono i primi tre termini della 4.37; per coprire però i termini minimi associati ai due 1 color blu ci sono i due sottocubi rossi oppure i due sottocubi verdi; poiché sono tra loro indipendenti, possiamo completare la copertura con entrambi i sottocubi rossi, entrambi i sottocubi verdi, oppure uno rosso e uno verde nei due modi possibili; queste combinazioni sono quelle relative agli ultimi due termini della 4.37.