L'algebra di Boole e la minimizzazione di funzioni booleane

Introduzione

Come già descritto, tutte le informazioni trattate da un calcolatore sono espresse da stringhe di bit, secondo convenzioni e codifiche adottate caso per caso. Le elaborazioni compiute su tali informazioni consistono pertanto nel costruire, a partire da determinate configurazioni di bit, altre configurazioni che rappresentino, nel codice e con le convenzioni prescelte, i risultati richiesti.
A livello logico, i circuiti elettronici preposti alla memorizzazione dei bit ed alla loro elaborazione sono detti di *commutazione, numerici o logici* e sono caratterizzati dal fatto che i relativi segnali di ingresso e uscita, dovendo rappresentare ciascuno un bit, sono di tipo *binario*, cioè possono assumere due soli valori diversi: presenza o assenza di un impulso, uno fra due valori di tensione, corrente o frequenza e così via.
Lo studio di tali circuiti, eseguito al livello delle unità hardware e dei loro componenti, può essere realizzato esaminandone il comportamento funzionale o "logico" e prescindendo dalla fisica realizzazione dei componenti di livello inferiore che li costituiscono; pertanto un circuito di commutazione si presenta come un n-polo ai cui morsetti sono applicati segnali binari e le cui funzioni sono definite dalle leggi di corrispondenza fra i segnali di ingresso e quelli di uscita; una rete di n-poli di tal genere costituisce una *rete di commutazione* o *rete logica*.
Uno strumento di incomparabile utilità per l'analisi, il progetto e la sintesi delle reti di commutazione è costituito dall'*algebra di Boole*, che consente di descrivere in forma algebrica le funzioni dei circuiti componenti e delle reti, fornendo altresì i metodi per la realizzazione del progetto logico.
Con l'impiego sistematico dell'algebra di Boole, il progetto di un'apparecchiatura numerica viene condotto secondo una metodologia puramente matematica che fornisce indicazioni non solo sul comportamento della rete, ma anche sulla sua struttura e, soprattutto, sui collegamenti di realizzare fra gli n-poli elementari per ottenere dalla rete una data elaborazione dei segnali. Ciò è possibile in quanto l'algebra di Boole consente di stabilire una corrispondenza biunivoca fra espressioni algebriche e reti di commutazione: le prime descrivono cioè le elaborazioni effettuate dalle seconde nonché la loro realizzazione in termini di schemi a blocchi sufficientemente dettagliati e, viceversa, data una rete, si trae da questa l'espressione algebrica che la descrive.
George Boole (1815-1864) nel suo lavoro *An investigation into the laws of thought on which are founded the mathematical theories of logic and probabilities* studiò un mezzo matematico per descrivere in forma algebrica la logica delle proposizioni non analizzate (logica aristotelica) e le relazioni fra di esse; l'algebra di Boole nacque pertanto nel quadro degli studi sulla logica matematica, iniziati da Leibnitz nel XVII secolo e particolarmente fiorenti nel secolo scorso. Trascurata per lungo tempo, essa ha avuto negli ultimi decenni un rinnovato sviluppo sia dal punto di vista puramente matematico, per il ritrovato interesse nel campo della logica matematica, sia dal punto di vista applicativo, per la sua utilità nel campo dell'analisi e della sintesi dei circuiti numerici.
Oggi l'algebra di Boole è inquadrata nella più vasta teoria delle strutture algebriche: le idee ed i concetti originali del logico inglese sono stati ampliati e la struttura da lui proposta viene vista come caso particolare di strutture più generali, che vanno genericamente sotto il nome di *algebre di Boole*. In tali strutture si inquadrano, come altrettanti "modelli", *l'algebra della logica* che, sulla base dei concetti originali del Boole, è applicata agli studi di logica formale, *l'algebra degli insiemi*, applicata alla teoria degli insiemi, l'*algebra binaria*, di nostro interesse, per lo studio ed il progetto dei circuiti numerici e l'*algebra delle reti* per lo studio dei sistemi di interconnessione.

Gli elementi base dell'algebra booleana e le sue proprietà

Come tutte le algebre, l'algebra di Boole si sviluppa partendo dalla definizione di un *insieme di supporto X*, detto anche *sostegno*, su cui sono definiti degli operatori binari che associano a coppie di elementi di *X* ancora un elemento dello stesso insieme.
Gli operatori definiti nell'algebra di Boole sono due:

- l'operatore di *somma* che si indicherà con $\oplus$,
- l'operatore di *prodotto* che si indicherà con $\otimes$.

Entrambi gli operatori sono funzioni definite nel dominio $X \times X$ e hanno per codominio *X*.
Ogni tripla del tipo $<X, \oplus, \otimes>$ è detta *algebra*. Su una data algebra possono poi valere le seguenti proprietà definite assiomaticamente per ogni *x, y,* z appartenenti a *X*:

I. *proprietà commutativa*:
   1. della somma: $x \oplus y = y \oplus x$
   2. del prodotto: $x \otimes y = y \otimes x$

II. *proprietà distributiva*:
   1. della somma rispetto al prodotto: $x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$
   2. del prodotto rispetto alla somma: $x \oplus (y \otimes z) = (x \oplus y) \otimes (x \oplus z)$

III. *dell'esistenza dell'elemento neutro*:
   1. per la somma: $x \oplus O = x$
   2. I per il prodotto: $x \otimes I = x$

   con *O* ed *I* appartenenti all'insieme *X*;

IV. *del complemento*:
   per cui per ogni elemento *x* del sostegno deve esistere e deve essere unico l'elemento *x'*, detto *complemento* di *x*, tale che:
   1. $x \oplus x' = I$
   2. $x \otimes x' = O$

Data la univocità del complemento, si può introdurre nell'algebra un terzo operatore unario denominato *complemento* ed indicato con $\neg$. Con la notazione $x' = \neg x$ si fa riferimento al complemento di *x*. Si noti che valgono le seguenti proprietà:

- $O = \neg I$
- $I = \neg O$
- $x = \neg(\neg x)$

Le quattro proprietà rappresentano le *regole fondamentali* del modello di algebra. Con esse si possono poi dimostrare le ulteriori proprietà:

V. *associativa*:
   $\alpha)$ della somma: $x \oplus (y \oplus z) = (x \oplus y) \oplus z$
   $\beta)$ del prodotto: $x \otimes (y \otimes z) = (x \otimes y) \otimes z$

VI. *idempotenza*:
   a) per la somma: $x \oplus x = x$
   b) per il prodotto: $x \otimes x = x$

VII. *assorbimento*:
   $\alpha)$ della somma: $x \oplus (x \otimes y) = x$
   $\beta)$ del prodotto: $x \otimes (x \oplus y) = x$

VIII. *del minimo e del massimo*:
   a) per il minimo: $x \otimes O = O$
   $\beta)$ per il massimo: $x \oplus I = I$

Una sestupla di elementi $<X, \oplus, \otimes, \neg, O, I>$ per cui valgono le proprietà I, V, VI, VII discusse è detta *reticolo*. Una delle proprietà di un reticolo è che risulta possibile introdurre in esso una relazione d'ordine $x<=y$ ($x \oplus y=y$ o $x \otimes y=x$) che gode delle proprietà *riflessiva*, *antisimmetrica* e *transitiva*.
Un reticolo che gode delle rimanenti proprietà II, II, IV e VIII è detto *Algebra di Boole*.

Si osservi che tutti gli assiomi di un'algebra di Boole sono caratterizzati dal *principio di dualità* che recita che d*a una qualsiasi identità booleana se ne ricava un'altra sostituendo l'operatore di somma con quello di prodotto e l'elemento O con I.*
Il principio di dualità non dice che le due identità sono uguali o equivalenti, ma afferma soltanto che se una identità è valida nell'algebra lo è anche la sua duale.

Si dimostra che in un'algebra di Boole valgono le due *relazioni di De Morgan*, l'una duale dell'altra:

1. $\neg (x \oplus y) = (\neg x \otimes \neg y)$
2. $\neg (x \otimes y) = (\neg x \oplus \neg y)$

Per la prima il complemento di una somma è uguale al prodotto dei complementi dei termini; per la seconda il complemento di un prodotto è uguale alla somma dei complementi dei fattori.

Dimostrazione:
Per il principio di dualità basta dimostrare la validità di una delle due relazioni. Si applichino le proprietà del complemento alla 1):

a) $(x \oplus y) \oplus (\neg x \otimes \neg y) = I$
b) $(x \oplus y) \otimes (\neg x \otimes \neg y) = O$

La a) si dimostra con i seguenti passaggi:

$(x \oplus y) \oplus (\neg x \otimes \neg y) =$ (per la proprietà distributiva)
$(x \oplus y \oplus \neg x) \otimes (x \oplus y \oplus \neg y) =$ (per la proprietà del complemento)
$(y \oplus I) \otimes (x \oplus I) =$ (per la proprietà del massimo)
$I \otimes I = I$

La b) si dimostra con passaggi duali:

$(x \oplus y) \otimes (\neg x \otimes \neg y) =$ (per la proprietà distributiva)
$(\neg x \otimes \neg y \otimes x) \oplus (\neg x \otimes y \otimes \neg y) =$ (per la proprietà del complemento)
$(\neg y \otimes O) \oplus (\neg x \otimes O) =$ (per la proprietà del minimo)
$O \oplus O = O$

Si noti che applicando ad entrambi i membri delle due relazioni di De Morgan la complementazione si ottiene:

1. $\neg(\neg(x \oplus y)) = \neg(\neg x \otimes \neg y)$
2. $\neg(\neg(x \otimes y)) = \neg(\neg x \oplus \neg y)$

ma la doppia negazione si può eliminare ottenendo:

1. $x \oplus y = \neg(\neg x \otimes \neg y)$
2. $x \otimes y = \neg(\neg x \oplus \neg y)$

dalla quale emerge che:

- *l'operatore somma si ottiene da una espressione in cui compaiono gli operatori di complemento e prodotto;*
- *l'operatore prodotto si ottiene da una espressione in cui compaiono gli operatori di complemento e somma.*

Se si definiscono:

- le *costanti booleane* come gli elementi appartenenti all'insieme *X* di sostegno;
- *le variabili booleane*, variabili a cui possono essere associati insiemi di costanti booleane;

allora si possono costruire *espressioni* con gli operatori di complementazione, somma e prodotto.

$$E = a \otimes b \oplus (\neg b \oplus (c \otimes a))$$

In assenza di parentesi si stabilisce che:

- si eseguano per prime le complementazioni;
- successivamente i prodotti;
- ed infine le somme.

In generale, quindi, due espressioni si dicono *equivalenti* se:

- assumono lo stesso valore comunque si assegni valore alle variabili in esse contenute;
- o è possibile trasformare una delle due nell'altra applicando i postulati dell'algebra.

All'algebra di Boole si associano diversi modelli nei quali si ritrovano interpretazioni diverse sia degli elementi dell'insieme di sostegno *X* che degli operatori.
I modelli più importanti sono:

- *l'algebra booleana binaria*;
- *l'algebra degli insiemi;*
- *l'algebra delle proposizioni*;
- *l'algebra delle reti*.

Poiché in ognuno di essi valgono le proprietà dell'algebra, vengono definiti come algebre di Boole.

L'algebra booleana binaria

Nell'algebra binaria, utilizzata per l'analisi *e progettazione di circuiti logici*, sono definiti:

- un insieme costituito da due soli valori $\{0,1\}$ detti *bit*;
- gli operatori di *somma logica*, *prodotto logico* e di *negazione* che come vedremo corrispondono ad altrettanti tipi di porte logiche.

Boole		**Binaria**	
Insieme di sostegno	*X*	*Due soli valori detti bit*	*{0,1}*
somma	$\oplus$	*Somma logica*	*OR*
prodotto	$\otimes$	*Prodotto logico*	*AND*
complemento	$\neg$	*Negazione*	*NOT*
minimo	*O*	*zero*	*0*
massimo	*I*	*uno*	*1*

Per gli operatori logici si è soliti far ricorso ad una notazione più semplice:

- per la somma si usa il simbolo +:
$$a \; OR \quad b \quad \sim a + b$$
- per il prodotto si usa il puntino · che però può anche essere omesso:
$$a \; AND \; b \quad \sim \quad a \cdot b \quad \text{oppure solo } ab$$
- per la negazione si "soprasegna" la variabile
$$NOT \; a \quad \sim \bar{a}$$

Nell'algebra binaria si definisce *funzione booleana y* su *n* variabili booelane:
$$y = f(x_1, x_2, \ldots, x_n)$$

la corrispondenza che associa ad ogni combinazione di valori delle *n* variabili indipendenti $x_1, x_2, \ldots, x_n$ un valore booleano della variabile dipendente *y*. Poiché ognuna delle variabili può assumere due soli valori, la funzione *f* viene ad essere definita su un insieme discreto di valori uguale proprio alle 2^n combinazioni di valori delle n variabili. La tabella che riporta tutte le 2^n combinazioni di valori delle *n* variabili e i valori corrispondenti della funzione viene detta *tavola di verità*.

Dalle tavole di verità si nota che l'*OR* tra due variabili logiche *a* e *b* assume valore 0 quando entrambe le variabili sono zero, ed il valore 1 negli altri tre casi. L'*AND* assume invece il valore uno quando entrambe le variabili valgono 1 e zero in tutti gli altri casi. Il *NOT*, che si applica ad una sola variabile, è uguale ad 1 se la variabile è 0, e 0 in caso contrario.

a	*b*	*ab*
0	0	0
0	1	0
1	0	0
1	1	1

a	*b*	*a* + *b*
0	0	0
0	1	1
1	0	1
1	1	1

a	$\bar{a}$
0	1
1	0

Una qualsiasi funzione *f* si presenta quindi come un'espressione composta dagli operatori *OR*, *AND* e *NOT*: la presenza di parentesi fissa l'ordine di esecuzione; in assenza delle parentesi si esegue prima il *NOT*, poi l'*AND* ed infine l'*OR*.

a	b	c	$\bar{a}b+\overline{(b\bar{c})}$	$\bar{a}+(b\bar{c})$
0	0	0	1	1
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	0
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0

Due funzioni che presentano la stessa tavola di verità sono equivalenti. A titolo di esempio si dimostra la relazione di De Morgan:

$$\overline{ab} = \bar{a} + \bar{b}$$

costruendo le tavole di verità di primo e secondo membro e verificando che sono uguali.

a	*b*
0	0
0	1
1	0
1	1

$\overline{ab}$
1
1
1
0

$\bar{a}+\bar{b}$
1
1
1
0

Poiché una funzione booleana definita su due variabili può assumere solo quattro valori, 2^4 sono le possibili funzioni binarie diverse che si possono applicare nell'algebra:

a	*b*	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}
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

con:

- f_0: detta *contraddizione*;
- f_1: ab; ossia l'*AND*;
- f_2: $a\bar{b}$;
- f_3: a; ossia uguale ad a;
- f_4: $\bar{a}b$;
- f_5: b; ossia uguale a b;
- f_6: $\bar{a}b + a\bar{b}$; detta *or esclusiva* o *XOR* che risulta uguale ad 1 quando i valori di a e b sono diversi;
- f_7: $a + b$; ossia l'*OR*;
- f_8: $\overline{a+b}$; ossia il complemento dell'*OR* detta anche *NOR* (indicata con il simbolo ↓);
- f_9: $\bar{a}\bar{b}$ ab; detta *equivalenza EQ* che risulta uguale ad 1 quando i valori di a e b sono uguali;
- f_{10}: $\bar{b}$; il complemento di b;
- f_{11}: $a+\bar{b}$; detta *implicazione* $b \rightarrow a$, con b antecedente;
- f_{12}: $\bar{a}$; il complemento di a;
- f_{13}: $\bar{a} + b$; detta *implicazione* $a \rightarrow b$, con a antecedente;
- f_{14}: $\overline{ab}$; ossia il complemento dell'*AND* detta anche *NAND* (indicata con il simbolo |);
- f_{15}: *tautologia*.

La tavola di verità consente anche di determinare la forma algebrica di una funzione f che essa rappresenta. Infatti fissata una tavola di verità, è possibile ricavare da essa due rappresentazioni della funzione dette *forme normali* o *canoniche*, non solo tra loro equivalenti ma equivalenti anche a qualsiasi altra espressione indicante la stessa funzione.
Se si indica con *letterale* l'occorrenza di una variabile a in una funzione di n variabili sia nella sua forma semplice a che in quella complementata $\bar{a}$, si definisce:

- *clausola di ordine n* (n>=1) ogni prodotto di n letterali,
- *fattore di ordine n* (n>=1) ogni somma di n letterali.

Su n variabili si determinano 2^n clausole o fattori diversi di ordine n, tanti quante sono le combinazioni dei letterali nelle due forme ammesse (semplice e complementata). A partire da clausole e fattori e possibile poi definire per una funzione booleana di n variabili:

- i *mintermini*, (P_i) ovvero clausole di ordine n che assumono valore 1 per una sola combinazione (i) dei valori delle variabili indipendenti; nello specifico il valore della variabile deve essere 1 se il suo letterale è nella forma semplice, 0 se è complimentata;
- i *maxtermini* (S_i) ovvero fattori di ordine n che assumono valore 0 per una sola combinazione (i) dei valori delle variabili indipendenti; nello specifico il valore della variabile deve essere 0 se il suo letterale è nella forma semplice, 1 se è complementata.

Ad ogni mintermine è naturale associare un pedice che rappresenta la codifica decimale della stringa binaria relativa alla configurazione delle n variabili di ingresso cui esso vale 1. Dualmente, ad un maxtermine è usuale associare un pedice che rappresenta la codifica decimale della stringa binaria relativa alla configurazione delle n variabili di ingresso complementate per cui esso vale 0.
In una tavola di verità ad ogni riga corrisponde pertanto un unico mintermine che assume valore 1 oppure un unico maxtermine che assume valore 0.

a	b	c	Mintermine = 1	Maxtermine = 0
0	0	0	$\bar{a}\bar{b}\bar{c}$ (P_0)	$a+b+c$ (S_7)
0	0	1	$\bar{a}\bar{b}c$ (P_1)	$a+b+\bar{c}$ (S_6)
0	1	0	$\bar{a}b\bar{c}$ (P_2)	$a+\bar{b}+c$ (S_5)
0	1	1	$\bar{a}bc$ (P_3)	$a+\bar{b}+\bar{c}$ (S_4)
1	0	0	$a\bar{b}\bar{c}$ (P_4)	$\bar{a}+b+c$ (S_3)
1	0	1	$a\bar{b}c$ (P_5)	$\bar{a}+b+\bar{c}$ (S_2)
1	1	0	$ab\bar{c}$ (P_6)	$\bar{a}+\bar{b}+c$ (S_1)
1	1	1	abc (P_7)	$\bar{a}+\bar{b}+\bar{c}$ (S_0)

Introdotti i mintermini e maxtermini, il passaggio dalla tavola di verità alla forma algebrica della funzione può essere ottenuto:

- come *SOP* (*sum of product*, *prima forma canonica*) sommando tutti e solo i mintermini che nella tavola di verità corrispondono alle righe nelle quali la funzione assume valore 1;
- come *POS* (*product of sum*, *seconda forma canonica*) moltiplicando tutti e soli i maxtermini che nella tavola di verità corrispondono alle righe nelle quali la funzione assume valore 0.

a	b	c	f_1	f_2	f_3
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	0	0
0	1	1	0	0	1
1	0	0	0	0	1
1	0	1	1	0	1
1	1	0	0	0	1
1	1	1	0	1	1

Ad ogni funzione corrisponde quindi una sola forma *SOP* ed una sola forma *POS*, tra loro equivalenti.

f	*SOP*	*POS*
f_1	$\bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + ab\bar{c}$	$(a+b+\bar{c})(a+\bar{b}+\bar{c})(\bar{a}+b+c)(\bar{a}+\bar{b}+c)(\bar{a}+\bar{b}+\bar{c})$
f_2	abc	$(\bar{a}+\bar{b}+\bar{c})(a+b+\bar{c})(a+\bar{b}+\bar{c})(a+\bar{b}+c)(\bar{a}+\bar{b}+c)(\bar{a}+b+\bar{c})$ $(\bar{a}+\bar{b}+\bar{c})$
f_3	$\bar{a}bc + a\bar{b}\bar{c} + a\bar{b}c + abc + abc$	$(a+b+c)(a+b+\bar{c})\,(a+\bar{b}+c)$

Si sceglie pertanto tra le due quella che presenta il minor numero di mintermini o di maxtermini: la *SOP* conviene quando la funzione assume il valore 1 un numero di volte inferiore al valore 0 (primi due casi dell'esempio); conviene la *POS* in caso contrario (terzo caso in tabella).
A titolo di esempio si determinino le due funzioni somma (S) e riporto (R) utili per il calcolo della somma algebrica di due bit. Per la tavola di verità di S si ricorda che essa assume valore 0 quando i due termini sono entrambi uguali, e il valore 1 nel caso hanno valore diverso. La funzione R assume il valore 1 solo quando a e b hanno entrambi valore 1: infatti la somma algebrica di 1+1 in binario è uguale a 0 con riporto 1.

a	b	S	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Le due funzione *S* ed *R* sono ottenibili come:

$$S = \bar{a}b + a\bar{b} \qquad e \qquad R = ab$$

Si noti che e $R = a\ AND\ b$.
L'esistenza delle forme canoniche fa capire come l'insieme di operatori <*AND, OR , NOT*> sia un insieme *funzionalmente completo*, nel senso che una qualsiasi funzione può essere rappresentata in una forma algebrica che impiega unicamente questi tre operatori.
In realtà esistono vari insiemi di operatori funzionalmente completi. Come visto dall'applicazione delle relazioni di De Morgan, una *OR* si può ottenere da una espressione in cui compaiono gli operatori di *AND* e *NOT*, e dualmente, una *AND* è ottenibile da espressioni che utilizzano operatori *OR* e *NOT*. Ovvero i due insiemi di operatori <*AND, NOT*> e <*OR, NOT*> sono funzionalmente completi.
Attraverso semplici manipolazioni algebriche è infine possibile dimostrare che gli operatori *NAND* e *NOR* costituiscono da soli scelte funzionalmente complete. In particolare, una funzione in prima forma canonica può essere trasformata in una forma equivalente in *logica NAND*, sostituendo tutti gli operatori con *NAND*, e dualmente, una funzione in seconda forma canonica può essere trasformata in una forma equivalente in *logica NOR*, sostituendo tutti gli operatori con *NOR*.

Ad esempio la funzione in forma *SOP*:

$f = \bar{a}b\bar{c} + a\bar{b}c$

può essere trasformata nella forma *NAND*:

f= ((a|a) | b | (c|c)) | (a | (b|b) | c)

mentre la funzione in forma POS:

$f=(a+b+c)(a+b+\bar{c})$

può essere sempre trasformata nella forma *NOR*

f= (a ↓ b ↓ c) ↓ (a ↓ b ↓ (c↓c))

Nella definizione di una funzione $y = f(x_1, x2,...,x_n)$ può accadere che in corrispondenza di alcune n-ple di valori delle x_i non sia assegnato il valore di *y* o, in altri termini, per la funzione è *indifferente* il valore (0 o l) assunto in tali punti. Ciò può avvenire per due motivi:

1. le variabili x_i non sono in realtà indipendenti fra loro, ma assumono solo alcune delle 2^n possibili configurazioni; risulta allora indifferente il valore che *y* assume in corrispondenza delle configurazioni che in realtà non si verificheranno mai;
2. pur assumendo le x_i i valori corrispondenti ai punti di non specificazione, è realmente indifferente ai fini pratici il valore di *y* in tali punti.

Si dice allora che la funzione ha *punti di indifferenza* (*don't care*) o valori di non specificazione: essi saranno indicati con il simbolo ϕ (che ricorda appunto sia il simbolo "0" che il simbolo "l") o con il simbolo - .
Ovviamente, se *y* ha *m* punti di indifferenza, esistono 2^m funzioni compatibili; assumeremo una qualsiasi di queste per rappresentare *y*.

Ad esempio, per la funzione di seguiti si ricavano le seguenti funzioni compatibili in forma normale *POS*:

a	***b***	***c***	***y***
0	0	0	1
0	0	1	0
0	1	0	Φ
0	1	1	Φ
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

$y_1 = P_0 + P_4$
$y_2 = P_0 + P_4 + P_2$
$y_3 = P_0 + P_4 + P_3$
$y_4 = P_0 + P_4 + P_2 + P_3$

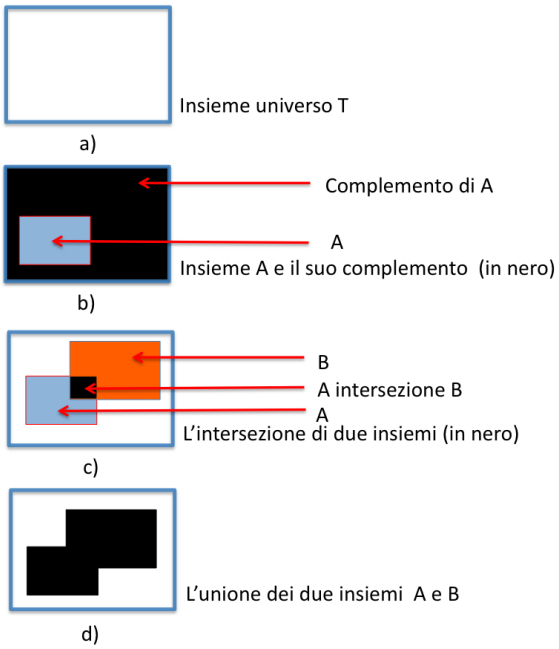
Algebra degli insiemi

Anche l'algebra degli insiemi è un modello isomorfo all'algebra di Boole nel quale si definiscono:

- l'*universo T* come insieme contenente un qualsiasi altro insieme, compreso l'insieme che non ha elementi detto *insieme vuoto*;
- l'unione di *A* e *B* come l'insieme di tutti gli elementi appartenenti ad entrambi;
- l'intersezione di *A* e *B* come l'insieme contenente gli elementi che sono presenti contemporaneamente in *A* e *B*;
- la complementazione dell'insieme *A* come l'insieme di tutti gli elementi dell'universo *T* che ad esso non appartengono.

Boole		**Insiemi**	
Insieme di sostegno	*X*	*L'universo del discorso contenente tutti gli insiemi compreso quello vuoto*	*T*
somma	$\square$	*Unione di due insiemi*	$\cup$
prodotto	$\square$	*Intersezione di due insiemi*	$\cap$
complemento	$\neg$	*Complemento*	$\overline{A}$
minimo	*O*	*Insieme Vuoto*	$\varnothing$
massimo	*I*	*Insieme Universo*	*T*

Nell'algebra degli insiemi le relazioni dell'algebra possono essere illustrate su un piano mediante i *diagrammi di Venn*. In tale rappresentazione gli insiemi vengono rappresentati con figure piane con contorno chiuso e l'insieme T con un rettangolo nel quale sono contenute tutte le altre figure.



A titolo di esempio si nota osservando la figura b) che $A \cup \overline{A} = T$ e $A \cap \overline{A} = \phi$, ma anche che $A \cup A = A$ e $A \cap A = A$.

L'importanza dell'algebra degli insiemi risiede nel teorema di *Stone* del '36 secondo cui ogni *algebra di Boole è rappresentabile su di un'algebra degli insiemi* e quindi dimostrabile mediante i diagrammi di Venn.

Algebra delle proposizioni

La logica matematica applica il formalismo matematico a processi di logica quali la deduzione e l'induzione. In logica, tra le espressioni linguistiche, sono di particolare interesse le proposizioni, ossia quelle frasi che implicano un valore di verità e falsità: ad esempio la frase "*Roma è la capitale d'Italia*" è vera. Sono proposizioni variabili quelle frasi per le quali il valore di verità e falsità dipende dal riscontro con il contesto in cui la frase è usata: ad esempio *"il docente del corso è giovane"* risulta vera o falsa solo dopo aver scoperto l'età del docente. La *tautologia* è una frase sempre vera; una *contraddizione* è sempre falsa. Le proposizioni possono solo assumere o il valore vero o il valore falso (in inglese *TRUE* e *FALSE*).
Nel nostro linguaggio, l'italiano, due o più proposizioni possono essere combinate in proposizioni composte mediante:

- la congiunzione *"e"* ($\wedge$)
- la disgiunzione *"oppure"* ($\vee$).

La congiunzione comporta che la proposizione composta è vera solo quando tutte le proposizioni componenti sono vere, in tutti gli altri casi risulta falsa.

"Gli appunti sono completi" e "validi didatticamente"

La disgiunzione comporta che la proposizione composta è falsa solo quando tutte le proposizioni componenti sono false, in tutti gli altri casi risulta vera.

"x<a" oppure "x>b"

Una proposizione può essere negata con il *non* ($\neg$): la nuova proposizione è falsa se in partenza era vera, è falsa altrimenti.

"oggi non è una giornata fortunata"

La logica delle preposizioni è un'algebra di Boole nella quale valgono le definizioni di tabella.

Boole		**Logica proposizione**	
Insieme di sostegno	*X*	*Due soli valori di verità*	*{falso,vero}*
somma	$\sqcup$	*Disgiunzione*	*oppure*
prodotto	$\sqcap$	*Congiunzione*	*e*
complemento	$\neg$	*Negazione*	*non*
minimo	*O*	*contraddizione*	*falso*
massimo	*I*	*tautologia*	*vero*

Con le preposizioni si dimostrano le relazioni di De Morgan ragionando sull'appartenenza di un punto ad un intervallo: infatti dire che un punto è interno ad un intervallo equivale a dire che non è esterno ad esso. Allora dati due punti a e b, con a<b, un punto x è interno se:

$$x \geq a \text{ e } x \leq b$$

mentre un punto y è esterno se:

$$y < a \text{ oppure } y > b$$

ma essere interno equivale a non essere esterno, quindi:

$$x \geq a \text{ e } x \leq b = \text{non } (x < a \text{ oppure } x > b)$$

Poiché:

$$x \geq a = \text{non } x < a$$
$$x \leq b = \text{non } x > b$$

si ha la relazione di De Morgan:

$$\text{non } (x < a \text{ oppure } x > b) = (\text{non } x < a) \text{ e } (\text{non } x > b)$$

L’algebra di Boole è uno strumento algebrico per trattare le proprietà logiche delle proposizioni: i procedimenti della logica che consentono di trarre conclusioni da alcune premesse sono ricondotti a formulazioni algebriche e dimostrazioni matematiche.
Infatti ogni proposizione diventa un letterale che assume uno dei due valori di verità e le proposizioni composte sono funzioni logiche definite dalle loro tavole di verità. Per semplicità di notazione si è soliti o abbreviare *Falso* e *Vero* con *F* e *V,* o usare al loro posto la codifica binaria associando al primo il valore 0 e al secondo 1. Anche per gli operatori si è soliti adottare la stessa notazione introdotta nell’algebra binaria.

La logica delle proposizioni è importante perché consente di studiare la correttezza della formulazione delle condizioni usate negli algoritmi (e quindi nei programmi) per controllare il flusso di esecuzione. Infatti una condizione è una espressione logica desunta da proposizioni che caratterizzano lo svolgimento di parti dell’algoritmo. Ad esse si arriva analizzando frasi in cui alcune proposizioni implicano il valore di verità di altre proposizioni. Frasi tipiche sono:

- l’azione A deve essere eseguita se condizione è vera;
- l’azione A deve essere ripetuta se condizione è falsa;
- l’azione A deve continuare mentre la condizione è vera.

Ad esempio si considerino i due predicati:

- *x*: *“batteria carica”*
- y: *“serbatoio vuoto”*.

Si studi la condizione *f* :

“il motore si accende” se “batteria carica” oppure “serbatoio vuoto”

x	y	f
F	*F*	*F*
F	*V*	*V*
V	*F*	*V*
V	*V*	*V*

Dalle righe della tavola di verità si evince:

- riga 1: il motore non si accende quando la batteria non è carica e il serbatoio non è vuoto;
- riga 2: il motore si accende quando la batteria non è carica e il serbatoio è vuoto;
- riga 3: il motore si accende quando la batteria è carica e il serbatoio non è vuoto;
- riga 4: il motore si accende quando la batteria è carica e il serbatoio è vuoto.

Si noti che solo la terza riga riporta una condizione corretta della realtà: da essa si ricava la corretta condizione di accensione del motore:

“il motore si accende” se “batteria carica” e non “serbatoio vuoto”

Una nozione importante in logica è quella di implicazione: si dice che *"x implica y”* se dalla verità di *x* (detto antecedente) scaturisce necessariamente quella di *y* (detto conseguente); in termini algebrici, essendo l'implicazione falsa se e solo se *x* è vera ed *y* è falsa, applicando il teorema di De Morgan, si ha:

$$\overline{x \Rightarrow y} = x\,\bar{y}$$

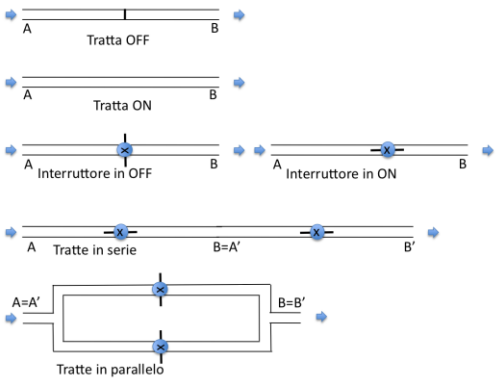
$$x \Rightarrow y = \bar{x} + y$$

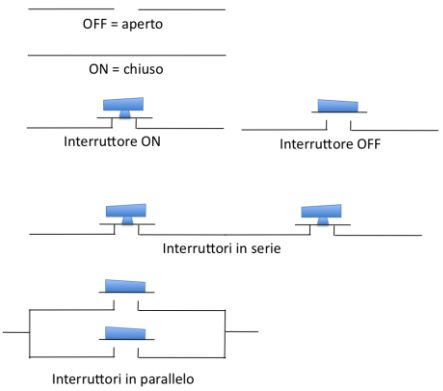
Algebra delle reti

Con la logica delle reti si studia il comportamento di un sistema di interconnessioni esistente tra due punti A e B al fine di determinare se tra di essi esiste un flusso. Le parti della rete sono tratte che collegano tra loro due punti. Le tratte possono essere disposte in modo che il punto terminale della prima coincida con quello iniziale della seconda oppure affiancandole mediante il congiungimento dei punti di ingresso e di uscita: nel primo caso si dice che le due tratte sono state disposte in *sequenza*, nel secondo si dice che sono in *parallelo*. Una tratta nella quale esiste sempre un flusso viene detta *ON*, si dice *OFF* una tratta nella quale il flusso si interrompe. Inoltre il flusso in una tratta può essere regolato da interruttore che pone in *ON* o in *OFF* la tratta stessa. Una sequenza di due tratte con interruttori è *ON* solo se i due interruttori sono entrambi *ON*; un parallelo delle stesse tratte è invece *OFF* solo se gli interruttori sono entrambi *OFF*. Sono alcuni esempi di reti le reti idriche, autostradali e i circuiti elettrici.
Se ad ogni tratta viene associata una variabile v_i che assume valori *OFF* ed *ON*, la funzione $y=f(v_1,v_2,\dots v_n)$ studia se esiste trasmissione tra i punti di ingresso ed uscita della rete.
Una rete siffatta è un modello di algebra di boole nel quale l'operatore di somma è il parallelo di tratte e il prodotto la sequenza di tratte.

Boole		**Logica delle reti**	
Insieme di sostegno	*X*	*Presenza o meno di trasmissione*	*{OFF, ON}*
somma	□	*Tratte in parallelo*	//
prodotto	□	*Tratte in sequenza*	— —
complemento	¬	*Passaggio da ON a OFF e viceversa*	*NOT*
minimo	*O*	*Assenza di flusso*	*OFF*
massimo	*I*	*Presenza di flusso*	*ON*

Nel caso di circuiti elettrici la condizione di *OFF* corrisponde ad un circuito aperto, e quella di *ON* ad uno chiuso: condizioni che determinano il transito di corrente.

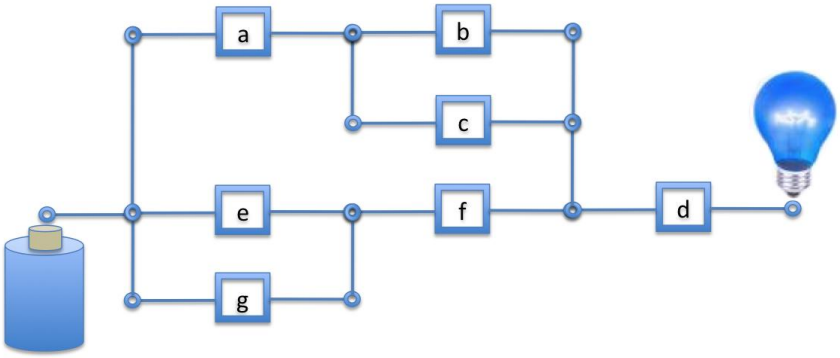




Affinché la lampadina della figura possa accendersi la funzione:

$$((a(b+c))+((e+g)f))d$$

deve essere *ON*.



Le mappe di Karnaugh

Le *mappe di Karnaugh* costituiscono uno strumento grafico molto utile per la rappresentazione e studio di funzioni booleane in forma canonica di poche variabili (n<=5) variabili. Su di esse è inoltre possibile eseguire manipolazioni e semplificazioni delle forme algebriche.
Una mappa di Karanugh per una funzione di n variabili in prima forma canonica si presenta come un insieme 2^n di celle (ad ognuna delle quale è associata un'etichetta corrispondente alla codifica binaria della relativa configurazione di variabili di ingresso) contenenti degli 1 in corrispondenza dei soli mintermini della funzione stessa. Dualmente, per le funzioni in seconda forma canonica di n variabili, una mappa di Karnaugh sarà costituita da un insieme di 2^n celle contententi degli 0 in corrispondenza dei soli maxtermini della funzione stessa.
Di seguito sono riportate le mappe di Karnaugh per funzioni in prima e seconda forma canonica di 2,3 e 4 variabili.

a/b	***0***	***1***
0	P_0	P_1
1	P_2	P_3

a/b	***0***	***1***
0	S_3	S_2
1	S_1	S_0

a/bc	***00***	***01***	***11***	***10***
0	P_0	P_1	P_3	P_2
1	P_4	P_5	P_7	P_6

a/bc	***00***	***01***	***11***	***10***
0	S_7	S_6	S_4	S_5
1	S_3	S_2	S_0	S_1

ab/cd	***00***	***01***	***11***	***10***
00	P_0	P_1	P_3	P_2
01	P_4	P_5	P_7	P_6
11	P_{12}	P_{13}	P_{15}	P_{14}
10	P_8	P_9	P_{11}	P_{10}

ab/cd	***00***	***01***	***11***	***10***
00	S_{15}	S_{14}	S_{12}	S_{13}
01	S_{11}	S_{10}	S_8	S_9
11	S_3	S_2	S_0	S_1
10	S_7	S_6	S_4	S_5

Come si può osservare, nelle mappe di Karnaugh, le clausole (fattori) di ordine n che differiscono per un solo letterale sono rappresentate da celle "adiacenti", ovvero con un lato in comune o con lati posti all'estremità della mappa (l'adiacenza logica si riflette in adiacenza geometrica).

In generale per funzioni di *n* variabili, clausole e fattori di ordine *k* (*k*<=*n*) sono rappresentati sulla mappa (cubo) da gruppi di 2^{n-k} celle (vertici) adiacenti, detti anche *sottocubi di ordine k*.
Nella mappa che segue sono riportati degli esempi di sottocubi di ordine 4 (in rosso, relativo alla clausola $\bar{a}\bar{b}\bar{c}\bar{d}$), di ordine 3 (in verde, relativo alla clausola $a\bar{c}\bar{d}$) e di ordine 2 (giallo, relativo alla clausola $\bar{b}c$) per una funzione *SOP* di 4 variabili.

ab/cd	***00***	***01***	***11***	***10***
00	P_0	P_1	P_3	P_2
01	P_4	P_5	P_7	P_6
11	P_{12}	P_{13}	P_{15}	P_{14}
10	P_8	P_9	P_{11}	P_{10}

Di seguito è riportata la mappa di Karnaugh relativa alla funzione *SOP* $y=\bar{a}\bar{b}\bar{c}+\bar{a}b\bar{c}+a\bar{b}c$, quella relativa alla funzione *POS* $z=(a+b+c)(a+b+\bar{c})(a+\bar{b}+c)$ e quella relativa alla funzione *SOP* $f=a\bar{b}cd+a\bar{b}c\bar{d}+abcd+ab\bar{c}d$

a/bc	***00***	***01***	***11***	***10***
0	1			1
1		1		

a/bc	***00***	***01***	***11***	***10***
0	0			0
1				0

ab/cd	***00***	***01***	***11***	***10***
00				
01				
11		1	1	
10			1	1

Implicanti ed Implicati di una funzione booleana

Le mappe di Karnaugh consentono di introdurre alcuni concetti molto importanti nello studio e manipolazione di funzioni booleane, ovvero quelli di *implicante* ed *implicato* di una funzione.
Sia data una funzione *f* di *n* variabili, vale la seguente definizione.

Una clausola *p* di ordine *k* si dice *implicante* di *f* se la funzione assume il valore 1 in corrispondenza di tutte le configurazioni delle variabili di ingresso relative ai vertici del sottocubo associato a *p*. Si scrive anche che *p*->*f* e che *f* "copre" o "include" *p*.

A partire dalla definizione di implicante è possibile dimostrare che vale la seguente proprietà.

Data una funzione *f* ed un insieme di implicanti $p_1, p_2, \ldots, p_k$ se *f* vale 1 in tutti ed i soli vertici dei sottocubi relativi ai vari p_i, allora essa si può esprimere in forma *SOP* come somma dei suoi implicanti, ovvero:

$f=p_1 + p_2 + \ldots + p_k$.

Un implicante *p*->*f* si dice *implicante primo* di *f*, se non esiste alcun altro implicante *p'* tale che *p'*->*f* e per cui *p'*->*p* (cioè *p'* copra *p*).

Gli implicanti primi di *f* sono quindi clausole associate ai sottocubi contenenti 1 e coperti da *f*, non contenuti in sottocubi con la stessa proprietà.

Di seguito sono mostrati degli esempi di implicanti primi: in giallo la clausola abd , in verde la calusola $a\bar{b}c$, in verde e giallo la clausola acd .

ab/cd	00	01	11	10
00				
01				
11		1	1	
10			1	1

Si dimostra che per ogni funzione *f* esiste almeno un insieme di implicanti primi $R=\{p_1, p_2, \ldots, p_k\}$ tale che *f* può essere espressa come $f = \sum_{p_i \in R} p_i$

Se l'insieme *R* inoltre non contiene implicanti coperti dall'unione di altri implicanti, allora questo è detto *insieme irridondante di implicanti primi* e la $\sum_{p_i \in R} p_i$ è una *forma SOP prima irridondante* di *f.*

Qualsiasi funzione *f* può essere semre espressa in forma *SOP* prima irridondante.

Un implicante primo *p->f* si dice poi *essenziale* se esiste almeno un vertice del sottocubo relativo a *p*, che non appartiene al sottocubo di alcun altro implicante primo di *f* (in altri termini se esiste qualche 1 della funzione *f* coperto solo dal sottocubo relativo a *p*). Chiaramente in ogni insieme irridondante di *f* sono presenti gli implicanti primi essenziali di *f.*

Nell'esempio precedente gli implicanti primi abd ed $a\bar{b}c$ sono essenziali, mentre acd non lo è; inoltre quest'ultimo è coperto dall'unione degli altri due che da soli costituiscono un insieme irridondante di *f*, per cui una forma SOP prima irridondante della funzione sarà:

$$f = abd + a\bar{b}c$$

Per le funzioni in forma POS valgono proprietà duali.

Sia data una funzione *f* di *n* variabili, vale la seguente definizione.

Una fattore *s* di ordine *k* si dice *implicato* di *f* se la funzione assume il valore 0 in corrispondenza di tutte le configurazioni delle variabili di ingresso relative ai vertici del sottocubo associato a *s* e si scrive *s<-f* .

Un implicato *s<-f* si dice *implicato primo* di *f*, se non esiste alcun altro implicato *s'* tale che *s'<-f* e per cui *s'<-s* (tale che gli zeri di *s'* coprano quelli di *s*).

Gli implicati primi di *f* sono quindi fattori associati ai sottocubi contenenti 0 e coperti da *f*, non contenuti in sottocubi con la stessa proprietà.

Un implicato primo *s<-f* si dice poi *essenziale* se esiste almeno un vertice del sottocubo relativo a *s*, che non appartiene al sottocubo di alcun altro implicato primo di *f* (in altri termini se esiste qualche 0 della funzione *f* coperto solo dal sottocubo relativo a *s*).

Si dimostra che per ogni funzione *f* esiste almeno un insieme di implicati primi $R'=\{s_1, s_2, \ldots, s_k\}$ tale che *f* può essere espressa come $f = \prod_{s_i \in R'} s_i$

Se l'insieme *R'* inoltre contiene implicati non coperti dall'unione di altri, allora questo è detto *insieme irridondante di implicati primi* e la $\prod_{s_i \in R'} s_i$ è una *forma POS prima irridondante* di *f.*

Qualsiasi funzione *f* può essere sempre espressa in forma *POS* prima irridondante.

La minimizzazione di funzioni booleane

Le mappe di Karnaugh e le forme irridondanti rappresentano uno strumento per *la minimizzazione del costo* di realizzazione dei circuiti logici che realizzano funzioni booleane.
In generale il costo di una funzione booleana sarà proporzionale al:

- *costo dei letterali* (C_l), ovvero al numero delle variabili indipendenti della funzione, ciascuna moltiplicata per il numero di volte che compare nella sua espressione;
- *costo delle funzioni o porte* (C_p), ovvero al numero di funzioni elementari che la compongono (e.g. *NOT*, *AND*, *OR*) ;
- *costo degli ingressi* (C_i), ovvero al numero di funzioni elementari che la compongono (e.g. *NOT*, *AND*, *OR*) ognuna moltiplicata per il numero di variabili di cui è funzione.

Trovare una forma minima di un funzione booleana significa minimizzarne il costo.

Si dimostra che: *le forme irridondanti di una funzione rappresentano delle forme a costo minimo di una funzione booleana*.

La minimizzazione di una funzione booleana con l'ausilio delle mappe di Karanugh (metodo grafico) procede quindi nei seguenti passi:

1. la funzione in forma *SOP* (*POS*) è rappresentata attraverso una mappa di Karnaugh;
2. si determinano gli implicanti (implicati) primi della funzione;
3. viene selezionato dall'insieme degli implicanti (implicati) primi un insieme irridondante individuando gli implicanti (implicati) primi essenziali;
4. si rappresenta la funzione in forma *SOP* (*POS*) irridondante.

Si vuole ad esempio trovare la forma minima della funzione *SOP* $f = a\bar{b}\bar{c}\bar{d} + ab\bar{c}\bar{d} + \bar{a}b\bar{c}d + ab\bar{c}d + \bar{a}bcd + \bar{a}\bar{b}cd$. Come primo passo andiamo a rappresentare la funzione attraverso una mappa di Karnaugh.

ab/cd	***00***	***01***	***11***	***10***
00			1	
01		1	1	
11	1	1		
10	1			

I possibili implicanti primi di *f* sono le clausole $a\bar{c}\bar{d}$ (in giallo), $\bar{a}cd$ (in verde), $b\bar{c}d$ (in azzurro) , $ab\bar{c}$ (in giallo ed azzurro) $\bar{a}bd$ (in verde ed azzurro). Di questi i primi due sono essenziali in quanto sono gli unici a coprire un 1 della funzione, il primo corrispondente alla configurazione 1000 ed il secondo alla configurazione 0011. Questi due implicanti si inseriscono nell'insieme irridondante: $R=\{ a\bar{c}\bar{d} , \bar{a}cd \}$. A questo punto si ha che se si seleziona l'implicante $b\bar{c}d$ gli implicanti $ab\bar{c}$ e $\bar{a}bd$ risultano coperti dagli altri, e quindi l'insieme finale è $R=\{ a\bar{c}\bar{d} , \bar{a}cd , b\bar{c}d \}$ e la relativa forma irridondante prima a costo minimo è:

$$f = a\bar{c}\bar{d} + \bar{a}cd + b\bar{c}d$$

Si vuole ad esempio trovare la forma minima della funzione *SOP* $f = a\bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}b\bar{c}d + ab\bar{c}d + \bar{a}\bar{b}c\bar{d} + a\bar{b}c\bar{d}$. Come primo passo andiamo a rappresentare la funzione attraverso una mappa di Karnaugh.

ab/cd	00	01	11	10
00	1			1
01		1	1	
11		1		
10	1			1

I possibili implicanti primi di f sono le clausole $\bar{b}\bar{d}$ (in rosso), $b\bar{c}d$ (in azzurro), $\bar{a}bd$ (in giallo ed azzurro). Questi sono tutti essenziali e quindi costituiscono un insieme irridondante: $R=\{\bar{b}\bar{d}\ ,\ b\bar{c}d\ ,\ \bar{a}bd\ \}$. La relativa forma irridondante prima a costo minimo è:

$$f=\bar{b}\bar{d}+\bar{a}bd+b\bar{c}d$$

Esempi di minimizzazione analoghi possono essere condotti sulle funzioni in forma *POS*.

Come visto nella descrizione del comportamento di una funzione booleana non sempre si specifica il valore delle uscite per tutte le configurazioni di ingresso, dando luogo a funzioni incomplete o non completamente specificate.
Per funzioni di tale tipo la minimizzazione con l'ausilio delle mappe di Karnaugh si ottiene nel seguente modo:

1. si attribuisce il valore 1(0) a tutte le condizioni di indifferenza e si determinano gli implicanti (implicati) primi della funzione così ottenuta;
2. si scartano gli implicanti (implicati) che coprono solo 1 (0) corrispondenti alle condizioni di indifferenza;
3. si effettua la scelta ottima degli implicanti (implicati) primi, notando però che solo gli 1 (0) della funzione originaria devono essere coperti.

Ad esempio si vuole minimizzare in forma SOP la funzione definita dalla seguente tabella di verità.

a	b	c	d	f
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	-
1	0	1	1	-
1	1	0	0	-
1	1	0	1	-
1	1	1	0	-
1	1	1	1	-

La mappa di Karnaugh ottenuta attribuendo il valore 1 (in rosso)alla funzione nei punti di indifferenza è la seguente, dove si individuano i seguenti implicanti primi: $\bar{b}\bar{d}$ (giallo), ab (verde), $\bar{c}\bar{d}$ (giallo, verde ed azzurro), ac (arancione, verde e giallo).

ab/cd	00	01	11	10
00	1			1
01	1			
11	1	1	1	1
10	1		1	1

Tutti gli implicanti trovati sono essenziali, ma gli implicanti ab ed ac si scartano in quanto coprono solo 1 relativi a condizioni di indifferenza, quindi un insieme irridondante di implicati primi è { $\bar{b}\bar{d}$, $\bar{c}\bar{d}$ }.

La minimizzazione con metodi sistematici

L'uso delle mappe di Karnaugh per la determinazione delle forme minime delle funzioni booleane diventa impraticabile al crescere del numero delle variabili, perché non è più semplice costruire le mappe. Di seguito si descrive un metodo di minimizzazione dovuro a *Quine-McCluskey* che, pur essendo in certi casi di complessità esponenziale e quindi impraticabile, può essere facilmente essere eseguito in modo automatico (nella pratica si usano delle *euristiche*).
Il metodo di *Quine-McCluskey* applicato a forme *SOP* determina in una prima fase tutti gli implicanti primi di una funzione ed in una seconda le forme minime. In modo duale il metodo si applica anche alle forme *POS*.

Determinazione degli implicanti primi

Sia assegnata la funzione in forma *SOP*:
$y = \sum(P_0, P_2, P_4, P_6, P_7, P_9, P_{11}, P_{15})$

In un primo passo, le configurazioni delle variabili di ingresso per cui la funzione vale 1 (quelle relative ai mintermini) vengono rappresentate in forma tabellare e suddivise in *classi* a seconda del *peso*, ovvero del numero di 1, in ordine di peso crescente. Ciascuna configurazione di una classe viene poi confrontata poi con tutte gli elementi della classe successiva: quando due configurazioni risultano uguali in tutti i bit eccetto 1, esse vengono fuse (operazione di *consenso*) in una configurazione unica formata dai bit coincidenti e da un trattino "-" nel bit diverso. Tutte le configurazioni che hanno originato un consenso vengono marcate con un simbolo "v" e non potranno essere più combinate in futuro.
Nel secondo passo viene esaminata la nuova tabella e si ripete il procedimento precedente combinando le configurazioni che differiscono di 1 bit. Si noti che nel passaggio da una tabella alla successiva una stessa configurazione può essere generata più volte per la fusione di configurazioni diverse, ma viene inserita una sola volta nella tabella etichettandola con tutte le configurazioni da cui può scaturire.
Il procedimento termina quando in un passo non combina più alcuna coppia di configurazioni e non si genera la tabella successiva. Gli implicanti primi corrispondono alle configurazioni non marcate, generate in tutti i passi.

Per la nostra funzione *y* si generano le tre tabelle seguenti.

	a	b	c	d	
0	0	0	0	0	v
2	0	0	1	0	v
4	0	1	0	0	v
6	0	1	1	0	v
9	1	0	0	1	v
7	0	1	1	1	v
11	1	0	1	1	v
15	1	1	1	1	v

	a	*b*	*c*	*d*	
0/2	0	0	-	0	v
0/4	0	-	0	0	v
2/6	0	-	1	0	v
4/6	0	1	-	0	v
6/7	0	1	1	-	$\bar{a}bc$
9/11	1	0	-	1	$a\bar{b}d$
7/15	-	1	1	1	bcd
11/15	1	-	1	1	acd

	a	*b*	*c*	*d*	
0/2 /4/6	0	-	-	0	$\bar{a}\bar{d}$

Al primo passo tutte le configurazioni danno origine ad un consenso, mentre al secondo passo le ultime 4 non originano consensi e quindi costituiscono degli implicanti primi. Al terzo passo la tabella è costituita da un'unica riga che rappresenta l'ultimo implicante primo ed il procedimento termine. L' insieme di implicanti primi è $\{ \bar{a}\bar{d}, \bar{a}bc, a\bar{b}d, bcd, acd \}$.

Selezione degli implicanti primi

Il problema di determinare una forma minima a partire dagli implicanti primi può essere descritto mediante una *tabella di copertura*, ove ogni riga corrisponde ad un implicante primo ed ogni colonna ad una configurazione di variabili per la cui la funzione vale 1. Una "x" in posizione *(i,j)* indica che l'implicante primo *i* copre l'1 della funzione corrispondente alla configurazione della colonna *j*.
La selezione degli implicanti si basa su tecniche di riduzione basate sui concetti di *essenzialità* e *dominanza*. Una riga è *essenziale* se è l'unica a possedere una "x" in qualche colonna. Si dice inoltre che una riga r_i *domina* un'altra riga r_j se r_i possiede tutte le "x" della r_j nelle colonne corrispondenti.
Determinate le righe essenziali si cancellano le colonne da loro coperte e si ottengono nuove tabelle di copertura per gli implicanti primi rimasti a cui è possibile applicare le regole di dominanza per eliminare gli implicanti primi dominati. Eliminate le righe dominate si ottengono altre tabelle di copertura ed il procedimento si itera finché non è più possibile trovare implicanti primi essenziali ed applicare regole di dominanza.

Per il nostro esempio la tabella di copertura iniziale è la seguente.

	0	***2***	***4***	***6***	7	9	11	15
$\bar{a}\bar{d}$	x	x	x	x				
$a\bar{b}d$						x	x	
$\bar{a}bc$				x	x			
acd							x	x
bcd					x			x

Come si può notare l'implicante primo è essenziale in quanto è l'unico a contenere delle x per le colonne 0,2 e 4. Similmente anche $a\bar{b}d$ è essenziale perché è l'unico a coprire la colonna 9. A questo punto i due implicanti sono selezionati (e le relative righe cancellate) e vengono eliminate le colonne 0, 2, 4, 6, 9 e 11 ottenendo la nuova tabella di copertura.

	7	15
$\bar{a}bc$	x	
acd		x
bcd	x	x

A questo punto si nota come la riga relativa a bcd domina le righe acd e $\bar{a}bc$ che quindi possono essere eliminate, ottenendo la nuova tabella di copertura di seguito con l'implicante bcd (l'unico rimasto) che diviene essenziale ed è quindi selezionato.

	7	15
bcd	x	x

Un insieme irridondante di implicanti primi è dato quindi da $\{\bar{a}\bar{d}, a\bar{b}d, bcd\}$.