Capítulo 1

Lógica y Álgebras de Boole

Teniendo en cuenta que en el diseño de sistemas digitales se hace un uso extensivo de la teoría de la lógica, comenzaremos este tema dando una breve introducción a la lógica.

En cualquier disciplina científica, se necesita distinguir entre argumentos válidos y no válidos. Para ello, se utilizan, a menudo sin saberlo, las reglas de la lógica. Algunas de ellas las estudiaremos aquí. En los temas siguientes, utilizaremos estas técnicas en las demostraciones de los razonamientos matemáticos e informáticos que nos vamos a encontrar.

Con un número finito de palabras y de construcciones gramaticales, se pueden construir una infinidad de frases; de la misma forma, con un número finito de proposiciones y conectores lógicos, se pueden construir infinitos razonamientos. La lógica de proposiciones tiene que elaborar métodos lo suficientemente potentes para tratar cualquiera de estos razonamientos.

1.1. Proposiciones y operadores lógicos

Diremos que una *proposición* o enunciado es una oración enunciativa de la que puede decirse si es verdadera o falsa pero no ambas cosas a la vez; es decir, a la que le asignaremos uno y sólo uno de los valores de verdad: verdadero (1) o falso (0), sin ambigüedades.

Ejemplo 1. *Los siguientes enunciados son proposiciones:*

- *Séneca fué un filósofo. (1)*
- *Aristóteles fue presidente de E.E.U.U.(0)*
- *El Deportivo ganó la última liga de fútbol profesional. (0)*
- $2 + 2 = 4$ *(1)*
- $2 + 4 = 7$ *(0),*

Ejemplo 2. *Los siguientes enunciados no son proposiciones:*

- *Ojalá llueva.*

- $x + 3 = 4$
- $x = -x$
- *Paradoja del barbero: En un pueblo, el barbero afeita a todos los hombres que no se afeitan a sí mismos (y únicamente a ellos)*

  `El barbero se afeita a sí mismo.`

  El enunciado no puede ser verdadero ya que si el barbero se afeitase a sí mismo, no formaría parte del grupo de hombres que él afeita. Por otro lado, no puede ser falso ya que, en ese caso, el barbero no se afeitaría a sí mismo y entonces debería ser afeitado por el barbero.

El cálculo proposicional es el estudio de las relaciones lógicas entre las proposiciones. Su objetivo es, por una parte, estudiar la validez de argumentos y, por otro lado, la formalización de argumentos del lenguaje natural.

Sintaxis Las proposiciones anteriores suelen designarse con letras minúsculas p, q, r, etc y se denominan *primitivas (simples)* ya que no hay forma de descomponerlas en otras más simples. Para obtener nuevas proposiciones, llamadas *compuestas*, se utilizan los *conectivos u operadores lógicos*. Los valores de verdad de las proposiciones resultantes dependerán de los valores de verdad de las proposiciones componentes. Nosotros utilizaremos, como símbolos, $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$.

- **Negación** $\neg p$. Se lee "no p"; "no ocurre p"; "no es cierto p".

p	$\neg p$
0	1
1	0

- **Conjunción** $p \wedge q$. Se lee "p y q"; "p sin embargo q"; "p no obstante q".

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

- **Disyunción** $p \vee q$. Se lee "p o q", "al menos p o q"; "como mínimo p o q" (inclusivo).

p	q	$p \vee q$
0	0	0
0	1	1
1	0	1
1	1	1

- **Condicional** $p \to q$. Se lee "Si p, entonces q"; "p es suficiente para q"; "q es necesario para p"; "q siempre que p"; "p sólo si q"; "q si p".

p	q	$p \to q$
0	0	1
0	1	1
1	0	0
1	1	1

- **Bicondicional** $p \leftrightarrow q$. Se lee "p, si, y sólo si, q"; "p es necesario y suficiente para q".

p	q	$p \leftrightarrow q$
0	0	1
0	1	0
1	0	0
1	1	1

Hay que hacer notar que el sentido de los conectores no coincide exactamente con el sentido que tienen en el lenguaje natural (metalenguaje). Por ejemplo, en lógica, el **y** se interpreta de tal manera que las proposiciones siguientes son equivalentes:

- Arturo se pone los calcetines y los zapatos.
- Arturo se pone los zapatos y los calcetines.

En el metalenguaje, no es así. La conjunción "y" puede expresar una idea de sucesión, con lo cual las frases no son equivalentes.

Comprender el significado del condicional es muy importante, y para ello damos el siguiente ejemplo.

Ejemplo 3. *Un político X promete en campaña electoral: "Si resulto elegido presidente, bajará los impuestos". Llamemos:*

- p : *"El político X es elegido presidente"*
- q : *"Se bajan los impuestos"*

La promesa del político se simboliza $p \to q$ y se lee "Si p, entonces q". Se pueden distinguir cuatro casos:

I) *p y q son ambas verdaderas, es decir el candidato es elegido presidente y baja los impuestos. Se cumple la promesa y el valor de verdad de la condicional es* 1.

II) *p es verdadera (El político X es elegido presidente) pero q es falsa (no baja los impuestos). Entonces, la promesa se ha roto y el valor de verdad de $p \to q$ es* 0.

III) *p y q son ambas falsas (El político X no resulta elegido y no se bajan los impuestos). La promesa no se ha roto ya que, al no haber sido elegido presidente, el candidato no tiene potestad para bajar los impuestos, por lo que el valor de verdad de* $p \to q$ *es* 1.

IV) *Finalmente, p es falsa (El candidato no resulta elegido) y q es verdadera (los impuestos bajan). Puede ser que el nuevo presidente está de acuerdo con nuestro candidato en bajar los impuestos y lo haga al llegar al poder. Tampoco, en este caso, se rompe la promesa y el valor de verdad de la condicional sigue siendo* 1.

Ejemplo 4. *Fijémonos en la diferencia entre el argumento anterior y el siguiente: "El padre de Juan le dice a éste: Te compraré un ordenador si, y solamente si, apruebas la Selectividad". En este caso, lo simbolizaríamos como* $p \leftrightarrow q$. *También, aqui hay cuatro posibilidades:*

I) *p y q son ambas verdaderas, es decir Juan aprueba la Selectividad y su padre le compra el ordenador. Se cumple la promesa y el valor de verdad de la bicondicional es* 1.

II) *p es verdadera (Juan aprueba la Selectividad) pero q es falsa (su padre no le compra el ordenador). Entonces, la promesa se ha roto y el valor de verdad de* $p \leftrightarrow q$ *es* 0.

III) *p y q son ambas falsas (Juan suspende la Selectividad y su padre no le compra el ordenador). La promesa no se ha roto ya que Juan no ha cumplido su parte, por lo que el valor de verdad de* $p \leftrightarrow q$ *es* 1.

IV) *Finalmente, p es falsa (Juan suspende) y q es verdadera (su padre le compra el ordenador). Como el padre dejo claro que solamente se compraría el ordenador si Juan aprobaba, el valor de verdad de* $p \leftrightarrow q$ *es* 0.

Al conectar entre sí las proposiciones mediante los operadores lógicos, se obtienen **expresiones bien formadas** (b.f.) Sólo hay que tener en cuenta:

I) Las proposiciones primitivas son siempre expresiones b.f..

II) Si $\mathcal{P}$ es una expresión b.f., $\neg\mathcal{P}$ tambien lo es.

III) Si $\mathcal{P}$ y $\mathcal{Q}$ son expresiones b.f., tambien lo son $\mathcal{P} \wedge \mathcal{Q}$, $\mathcal{P} \vee \mathcal{Q}$, $\mathcal{P} \to \mathcal{Q}$ y $\mathcal{P} \leftrightarrow \mathcal{Q}$.

IV) No hay más reglas.

Para la correcta relación entre las proposiciones y los conectivos en las fórmulas bien formadas, hay que tener en cuenta que *no deben aparecer conectivas adyacentes salvo la negación* [1].

[1] Así, la proposición $p \to \neg q$ sería correcta mientras que $p \to \wedge q$ no lo es.

Además, cuando hay más de una conectiva en una fórmula, entenderemos que cada conectiva afecta a la letra proposicional inmediata o, al conjunto de proposiciones inmediatas encerradas entre paréntesis. Se establece una jerarquía de prioridades entre las conectivas. En el primer nivel se sitúa la negación, en el segundo nivel la conjunción y la disyunción y en último nivel el condicional y el bicondicional. La prioridad dentro del mismo nivel se indica con paréntesis. Si tenemos la proposición compuesta

$$(\neg p \vee q) \wedge (r \to t)$$

y la escribiésemos sin paréntesis, tendríamos:

$$\neg p \vee q \wedge r \to t$$

que, según los niveles de prioridad establecidos, podría equivaler a:

$$[\neg p \vee (q \wedge r)] \to t \text{ o a } [(\neg p \vee q) \wedge r] \to t.$$

Sea $\mathcal{P}$ una proposición compuesta y llamemos $\sum$ al conjunto de sus proposiciones primitivas componentes. Una regla ϕ que hace corresponder a cada elemento de $\sum$ un valor de verdad se denota por

$$\sum \xrightarrow{\phi} \{0, 1\}$$

y se llama una *interpretación* de $\mathcal{P}$. Si $\mathcal{P}$ resulta ser verdadera, ϕ se denomina *modelo* y, si $\mathcal{P}$ es falsa, ϕ se denomina *contraejemplo*. Así, si p y q son dos proposiciones primitivas, entonces $\phi(p) = 0$ y $\phi(q) = 1$ es un contraejemplo para $p \wedge q$ y un modelo para $p \vee q$. Sin embargo, $\phi(p) = \phi(q) = 1$ es un modelo para ambas.

Diremos que un conjunto $\{p_1, p_2, \ldots, p_n\}$ de proposiciones es *consistente* si la conjunción de todas ellas $p_1 \wedge p_2 \wedge \ldots \wedge p_n$ admite algún modelo.

Ejemplo 5.

- *El conjunto* $\{p \to q,\ p \wedge q\}$ *es consistente ya que admite un modelo (p y q verdaderas).*
- *El conjunto* $\{p \to q,\ p \wedge \neg q\}$ *es inconsistente ya que no admite ningún modelo (El único modelo de* $p \wedge \neg q$ *es un contraejemplo de* $p \to q$*).*

1.2. Tablas de Verdad

El método de las tablas de verdad es mecánico, pero tedioso, y permite decidir si una fórmula dada es válida. Una *tabla de verdad* para una proposición compuesta construida a partir de proposiciones p, q, r, etc, es un método que proporciona los valores de verdad de la proposición compuesta, a partir de los valores de verdad de p, q, r, etc. Para construirla se determinan los valores de verdad de las subproposiciones desde las más sencillas hasta las más complejas. Por ejemplo, la tabla de verdad de $(p \vee q) \wedge (p \to \neg q)$ será

p	q	$p \vee q$	$\neg q$	$p \to \neg q$	$(p \vee q) \wedge (p \to \neg q)$
0	0	0	1	1	0
0	1	1	0	1	1
1	0	1	1	1	1
1	1	1	0	0	0

Cada fila de la tabla de verdad de una proposición $\mathcal{P}$ es una **interpretación** de $\mathcal{P}$. Si $\mathcal{P}$ está formada por n proposiciones primitivas, hay 2^n posibles interpretaciones de $\mathcal{P}$ y, en consecuencia 2^n filas en la tabla de verdad de $\mathcal{P}$. En el ejemplo anterior, la primera y la última fila se corresponden con contraejemplos y las otras dos son modelos.

Se pueden "simplificar" las tablas de verdad utilizando otra forma de disponer las proposiciones. Para el ejemplo anterior, nos quedaría:

	p	q	$p \vee q$	$(p \vee q) \wedge (p \to \neg q)$	$p \to \neg q$	$\neg q$
	0	0	0	0	1	1
	0	1	1	1	1	0
	1	0	1	1	1	1
	1	1	1	0	0	0
Paso	1		2	4	3	2

El valor de verdad de cada paso queda determinado por los valores de verdad de los pasos anteriores.

Cuando una proposición compuesta es siempre verdadera, independientemente de los valores de verdad de sus proposiciones componentes, se denomina *tautología* y se denotará T_0 o $\top$. Recíprocamente, cuando una proposición compuesta es siempre falsa, se denominará *contradicción* y se denotará F_0 o $\bot$. Tambien podríamos decir que, si cualquier interpretación de $\mathcal{P}$ es un modelo (resp. un contraejemplo), entonces $\mathcal{P}$ es una tautología (resp. una contradicción).

Ejemplo

I) $p \vee \neg p$ es una tautología.

p	$\neg p$	$p \vee \neg p$
0	1	1
1	0	1

II) $p \wedge \neg p$ es una contradicción.

p	$\neg p$	$p \wedge \neg p$
0	1	0
1	0	0

III) $\neg(p \wedge q) \leftrightarrow (\neg p \vee \neg q)$ es una tautología.

p	q	$p \wedge q$	$\neg(p \wedge q)$	$\neg(p \wedge q) \leftrightarrow (\neg p \vee \neg q)$	$\neg p \vee \neg q$	$\neg p$	$\neg q$
0	0	0	1	1	1	1	1
0	1	0	1	1	1	1	0
1	0	0	1	1	1	0	1
1	1	1	0	1	0	0	0

1.3. Implicaciones y Equivalencias lógicas

Consideremos las dos afirmaciones siguientes:

- El laboratorio está bien distribuido y bien equipado.
- El laboratorio está bien equipado y bien distribuído.

Trivialmente, estas dos afirmaciones tienen siempre los mismos valores de verdad, y, por tanto, decimos que son lógicamente equivalentes. Para hacer más precisa esta idea, traduzcámosla a la lógica. Si P expresa la proposición "el laboratorio está bien distribuido" y Q expresa la proposición "el laboratorio está bien equipado", entonces la primera de las dos afirmaciones se traduce en $P \wedge Q$, mientras que la segunda se traduce $Q \wedge P$. Mediante las tablas de verdad se puede comprobar que estas dos expresiones tienen los mismos valores de verdad para todas las asignaciones posibles; esto es, $(Q \wedge P) \leftrightarrow (P \wedge Q)$ es una tautología.

Definición 1. *Dos proposiciones compuestas $\mathcal{P}$ y $\mathcal{Q}$ son* lógicamente equivalentes*, si tienen los mismos valores de verdad para cada interpretación de los valores de verdad de sus proposiciones componentes. Esta situación la denotaremos:*

$$\mathcal{P} \Longleftrightarrow \mathcal{Q}.$$

Así pues, se tiene que $\mathcal{P}$ y $\mathcal{Q}$ son lógicamente equivalentes si, y sólo si, $\mathcal{P} \leftrightarrow \mathcal{Q}$ es una tautología. La diferencia entre " $\Longleftrightarrow$ " y " $\leftrightarrow$ " es importante ya que, mientras $\mathcal{P} \Longleftrightarrow \mathcal{Q}$ quiere decir que $\mathcal{P} \leftrightarrow \mathcal{Q}$ es una tautología, cuando escribimos $\mathcal{P} \leftrightarrow \mathcal{Q}$ simplemente estamos denotando una proposición compuesta.

Ejemplo La proposición compuesta $p \wedge q \leftrightarrow q$ no es una tautología, por lo que $p \wedge q$ y q no son lógicamente equivalentes.

p	q	$p \wedge q$	$\leftrightarrow$
0	0	0	1
0	1	0	0
1	0	0	1
1	1	1	1

Definición 2. *Dadas dos proposiciones $\mathcal{P}$ y $\mathcal{Q}$, se dice que $\mathcal{P}$* implica lógicamente *$\mathcal{Q}$, o que, de $\mathcal{P}$* se deduce *$\mathcal{Q}$, si $\mathcal{P} \rightarrow \mathcal{Q}$ es una tautología. Esta situación la denotaremos*

$$\mathcal{P} \Longrightarrow \mathcal{Q}$$

y significa que, cuando $\mathcal{P}$ es verdadera, también $\mathcal{Q}$ es verdadera y que cuando $\mathcal{Q}$ es falsa, $\mathcal{P}$ es falsa.

Teorema 1. *Las siguientes tablas recogen algunas equivalencias e implicaciones lógicas:*

Principales equivalencias lógicas	
Leyes Conmutativas	$p \vee q \iff q \vee p$ $p \wedge q \iff q \wedge p$
Leyes Asociativas	$(p \vee q) \vee r \iff p \vee (q \vee r)$ $(p \wedge q) \wedge r \iff p \wedge (q \wedge r)$
Leyes Distributivas	$p \vee (q \wedge r) \iff (p \vee q) \wedge (p \vee r)$ $p \wedge (q \vee r) \iff (p \wedge q) \vee (p \wedge r)$
Ley de la doble negación	$\neg\neg p \iff p$
Leyes de Morgan	$\neg(p \vee q) \iff (\neg p \wedge \neg q)$ $\neg(p \wedge q) \iff (\neg p \vee \neg q)$
Leyes de dominación	$p \vee T_0 \iff T_0$ $p \wedge F_0 \iff F_0$
Leyes de Identidad	$p \wedge T_0 \iff p$ $p \vee F_0 \iff p$
Leyes de la negación	$p \vee \neg p \iff T_0$ $p \wedge \neg p \iff F_0$
Ley de la Contraposición	$(p \to q) \iff (\neg q \to \neg p)$
Leyes de la Implicación	$(p \to q) \iff (\neg p \vee q)$ $(p \to q) \iff \neg(p \wedge \neg q)$
Leyes de la Equivalencia	$(p \leftrightarrow q) \iff [(p \to q) \wedge (q \to p)]$ $(p \leftrightarrow q) \iff (p \wedge q) \vee (\neg p \wedge \neg q)$
Leyes Idempotentes	$p \iff (p \wedge p)$ $p \iff (p \vee p)$
Ley de la reducción al absurdo	$(p \to q) \iff [(p \wedge \neg q) \to F_0]$

Principales implicaciones lógicas	
Modus Ponens	$[(p \to q) \wedge p] \Longrightarrow q$
Modus Tollens	$[(p \to q) \wedge \neg q] \Longrightarrow \neg p$
Silogismo	$[(p \to q) \wedge (q \to r)] \Longrightarrow (p \to r)$
Leyes de simplificación	$(p \wedge q) \Longrightarrow p$ $(p \wedge q) \Longrightarrow q$
Leyes de adición	$p \Longrightarrow (p \vee q)$ $q \Longrightarrow (p \vee q)$
Silogismo disyuntivo	$((p \vee q) \wedge \neg p) \Longrightarrow q$ $((p \vee q) \wedge \neg q) \Longrightarrow p$
Ley de casos	$[(p \to q) \wedge (\neg p \to q)] \Longrightarrow q$
Ley de inconsistencia	$[p \wedge \neg p] \Longrightarrow q$

Las siguientes reglas de "substitución" serán de gran utilidad. Sea $\mathcal{P}$ una proposición compuesta de la cual forma parte la proposición Q

- Si $\mathcal{P}$ es una tautología y cada aparición de Q en $\mathcal{P}$ se substituye por una proposición Q^*, obtenemos una proposición $\mathcal{P}^*$ que resulta ser también una tautología. Sabemos, por ejemplo que:

  $$\neg(p \vee q) \leftrightarrow (\neg p \wedge \neg q)$$

  es una tautología. Si reemplazamos p por $r \vee s$, obtenemos de nuevo una tautología:

  $$\neg((r \vee s) \vee q) \leftrightarrow (\neg(r \vee s) \wedge \neg q)$$

- Si $\mathcal{P}$ es una proposición arbitraria y se substituyen una o más ocurrencias de Q en $\mathcal{P}$ por una proposición Q^* lógicamente equivalente a Q, obtendríamos una proposición $\mathcal{P}^*$ lógicamente equivalente a $\mathcal{P}$. De este modo, si consideramos $\mathcal{P}$

  $$\neg(p \vee q) \rightarrow r$$

  se tiene que $\mathcal{P}$ es lógicamente equivalente a:

  $$(\neg p \wedge \neg q) \rightarrow r$$

Ejemplo 6. *Simplificar* $(p \vee q) \wedge \neg(\neg p \wedge q)$

$$\begin{array}{lll} (p \vee q) \wedge \neg(\neg p \wedge q) & \Longleftrightarrow & \textit{Leyes de Morgan y Doble negación} \\ (p \vee q) \wedge (p \vee \neg q) & \Longleftrightarrow & \textit{Leyes distributivas} \\ p \vee (q \wedge \neg q) & \Longleftrightarrow & \textit{Contradicción} \\ p \vee F_0 & \Longleftrightarrow & p \end{array}$$

1.4. Teoremas y Demostraciones

Un *Teorema* es un enunciado (matemático) cuya veracidad se confirma por medio de un argumento válido o *demostración.*

Un teorema consiste siempre en algunas proposiciones $H_1, H_2, \cdots, H_n$ llamadas *hipótesis o premisas* y una proposición C llamada *conclusión.* El argumento es válido siempre que

$$H_1 \wedge H_2 \wedge \ldots \wedge H_n \Longrightarrow C$$

o, lo que es lo mismo:

$$H_1 \wedge H_2 \wedge \ldots \wedge H_n \rightarrow C$$

es una tautología. Esta situación se denota también

$$\{H_1, H_2, \ldots, H_n\} \Longrightarrow C$$

y diremos que de las premisas se puede deducir la conclusión. Precisamente este es el tipo más natural de demostración llamada *demostración directa.* En particular, todas las implicaciones lógicas ya vistas (modus ponens, modus tollens, silogismo, etc) son ejemplos de argumentos válidos.

Ejemplo 7. *Demostrar la implicación lógica "Modus ponens", es decir*

$$p \wedge (p \to q) \Longrightarrow q$$

p	q	$p \to q$	$p \wedge (p \to q)$	$p \wedge (p \to q) \to q$
0	0	1	0	1
0	1	1	0	1
1	0	0	0	1
1	1	1	1	1

Si llamamos $\mathcal{P} = p \wedge (p \to q)$, es evidente que, de $\mathcal{P}$ se deduce q, ya que siempre que $\mathcal{P}$ es verdadera, también lo es q y, cuando q es falsa, también es falsa $\mathcal{P}$.

También son argumentos válidos la regla de la conjunción $\{p, q\} \Longrightarrow p \wedge q$, la ley de contradicción $\{\neg p \to F_0\} \Longrightarrow p$ y la ley de demostración por casos $\{p \to r, q \to r\} \Longrightarrow p \vee q \to r$.

Ejemplo 8. *Demostrar la implicación*

$$(p \to r) \wedge (q \to r) \Longrightarrow p \vee q \to r$$

p	q	r	$p \to r$	$q \to r$	$p \vee q \to r$
0	0	0	1	1	1
0	0	1	1	1	1
0	1	0	1	0	0
0	1	1	1	1	1
1	0	0	0	1	0
1	0	1	1	1	1
1	1	0	0	0	0
1	1	1	1	1	1

Observando la tabla anterior, podríamos fijarnos únicamente en las filas una, dos, cuatro, seis y ocho ya que el condicional solamente es falso si la premisa es verdadera y la conclusión falsa. Por lo tanto, basta comprobar aquellas filas donde todas las hipótesis son verdaderas, es decir, todos los modelos de las premisas (o hipótesis) han de ser modelos de la conclusión.

Una segunda opción para realizar una demostración directa es utilizar una sucesión de proposiciones, que terminan con la conclusión C, y que se consideran válidas por alguna de las siguientes razones:

I) Es una de las hipótesis.

II) Es una tautología conocida (equivalencia lógica).

III) Es lógicamente equivalente a una proposición anterior.

IV) Se deriva de alguna de las proposiciones anteriores por reglas de sustitución.

v) Se puede inferir de proposiciones anteriores mediante reglas de inferencia.

Las **reglas de inferencia** son técnicas que nos ayudan en las demostraciones de los teoremas. Cada regla de inferencia tiene su origen en una *implicación lógica*.

Ejemplo 9. *Demostrar el argumento*

$$(p \to r) \wedge (q \to r) \Longrightarrow p \vee q \to r$$

I) $p \to r$ *; hipótesis*

II) $q \to r$ *; hipótesis*

III) $\neg p \vee r$ *; equivalencia lógica*

IV) $\neg q \vee r$ *; equivalencia lógica*

V) $(\neg p \vee r) \wedge (\neg q \vee r)$ *; regla de la conjunción*

VI) $(\neg p \wedge \neg q) \vee r$ *; distributiva*

VII) $\neg(p \vee q) \vee r$ *; ley de Morgan*

VIII) $p \vee q \to r$*; equivalencia lógica.*

Un tipo de demostración indirecta es la *contraposición*. Este tipo de demostración esta basada en la tautología: $(p \to q) \leftrightarrow (\neg q \to \neg p)$, en nuestro caso:

$$\neg C \Longrightarrow \neg(H_1 \wedge H_2 \wedge \ldots \wedge H_n)$$

Ejemplo 10. *Si a y b son números naturales y $a + b \geq 25$, entonces $a \geq 13$ ó $b \geq 13$. Denotemos:*

- $p : a \geq 13$
- $q : b \geq 13$
- $r : a + b \geq 25$

Queremos demostrar que $r \Longrightarrow p \vee q$. Para ello veremos que $\neg(p \vee q) \Longrightarrow \neg r$. Tengamos en cuenta que, por las leyes de Morgan

$$\neg(p \vee q) \Longleftrightarrow (\neg p \wedge \neg q).$$

Ahora bien, si $a \leq 12$ y $b \leq 12$, entonces $a + b \leq 24$, es decir, la proposición r es falsa, entonces $r \Longrightarrow p \vee q$ es verdadera.

Otro tipo de demostración indirecta es la demostración por *contradicción o reducción al absurdo*, que consiste en probar

$$\neg C \wedge H_1 \wedge H_2 \wedge \ldots \wedge H_n \Longrightarrow F_0$$

Ejemplo 11. *Si mis cálculos son correctos y pago la cuenta de la electricidad, me quedaré sin dinero. Si no pago la cuenta de la electricidad, me cortarán la corriente. Como no me han cortado la corriente y sigo teniendo dinero, mis cálculos no eran correctos.*

- p : *"Mis cálculos son correctos"*
- q : *"Pago la cuenta de la electricidad"*
- r : *"Me quedo sin dinero"*
- s : *"Me cortan la corriente"*

Se trata de probar que de $\{(p \wedge q) \to r, \neg q \to s, \neg r, \neg s\}$*, se deduce* $\neg p$*, o, equivalentemente, que*

$$((p \wedge q) \to r) \wedge (\neg q \to s) \wedge \neg r \wedge \neg s \wedge p \Longrightarrow F_0$$

Aplicando las equivalencias e implicaciones lógicas (reglas de inferencia), tenemos que:

$$\begin{array}{ll} ((p \wedge q) \to r) \wedge (\neg q \to s) \wedge \neg r \wedge \neg s \wedge p & \Longrightarrow \\ ((p \wedge q) \to r) \wedge q \wedge \neg r \wedge p & \Longrightarrow \\ r \wedge \neg r & \Longleftrightarrow \\ F_0 & \end{array}$$

Nota 1. *Es interesante destacar que el teorema*

$$\{H_1, H_2, \ldots, H_n\} \Longrightarrow C$$

es válido si, y sólo si, el conjunto

$$\{H_1, H_2, \ldots, H_n, \neg C\}$$

es inconsistente.

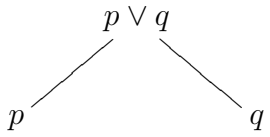
1.5. Tablas o Árboles Semánticos

Precisamente el método de demostración por contradicción o reducción al absurdo nos permite utilizar las llamadas *tablas semánticas*[2] para comprobar si un argumento es o no válido. El método (descubierto en los años cincuenta por **Beth y Hintikka**, independientemente uno del otro) permite saber si una proposición es una contradicción. Para ello, se construye un árbol donde los nodos (finitos) son las proposiciones, el conectivo $\wedge$ se representa por un arista vertical,

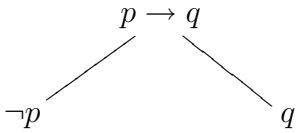
$$\begin{array}{c} p \wedge q \\ | \\ p \\ | \\ q \end{array}$$

[2]Quizás sería más adecuado el nombre de árbol semántico.

y el conectivo $\vee$ por un par de aristas en la forma



El resto de los conectivos se traducen a esa forma. Así, el condicional $p \to q$ se representa como

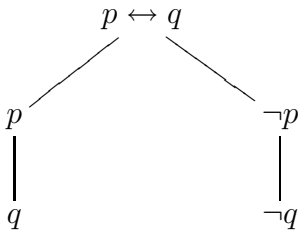


ya que $p \to q \iff \neg p \vee q$

Por otro lado, como

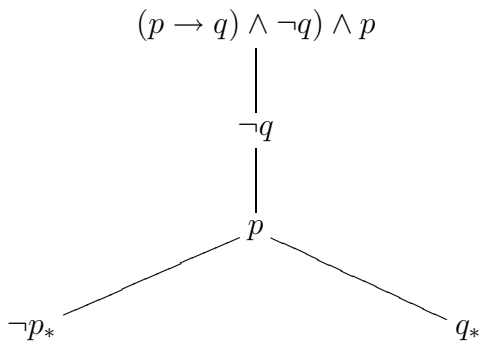
$$\begin{array}{rcl} p \leftrightarrow q & \iff & (\neg p \vee q) \wedge (\neg q \vee p) \\ & \iff & (\neg p \wedge \neg q) \vee (\neg p \wedge p) \vee (q \wedge \neg q) \vee (q \wedge p) \\ & \iff & (\neg p \wedge \neg q) \vee (p \wedge q) \end{array}$$

la bicondicional se representará

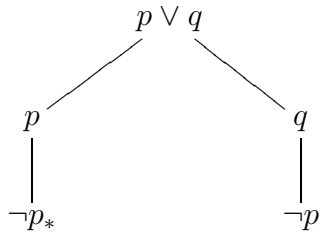


En este método, se van descomponiendo, por turno, cada proposición compuesta, de acuerdo con las reglas anteriores, marcando dicha proposición como ya utilizada. Conviene descomponer primero los bicondicionales y sus negaciones antes que otras conectivas que creen ramas. Si en una sucesión de nodos del árbol (*camino*), aparece una proposición y su negación, se dice que es un camino cerrado y se marca con $*$ el nodo final. Si al final del proceso todos los caminos se cierran, la proposición es una contradicción; en caso contrario, cada camino abierto es un modelo de la proposición inicial. Así pues, si queremos demostrar o refutar un argumento del tipo $H \Longrightarrow C$ calculamos la tabla semántica de $H \wedge \neg C$. Si al finalizar todos los caminos se cierran, tenemos que $H \wedge \neg C$ es una contradicción, es decir, el argumento $H \Longrightarrow C$ es válido. Por el contrario, la existencia de una rama abierta nos llevará a concluir que el argumento no es válido. Del mismo modo, si tenemos un sistema de proposiciones $\{p_1, p_2, \ldots, p_n\}$, sabremos que es consistente, si al construir la tabla semántica de $p_1 \wedge p_2 \wedge \ldots \wedge p_n$, nos queda algún camino abierto que representará un modelo para dicho sistema.

Ejemplo 12. I) *Demostrar el "modus tollens":* $((p \to q) \wedge \neg q) \Longrightarrow \neg p$



II) *Demostrar o refutar* $\{p \vee q\} \Longrightarrow p$



En este ejemplo vemos que $\{\neg p, q\}$ *es un contraejemplo ya que si* p *es falsa y* q *verdadera,* $p \vee q$ *es verdadera y* $(p \vee q) \to p$ *es falsa.*

III) *Si hay probabilidad de lluvia o hace viento, Manuel no cortará el cesped. Siempre que no hay nubes en el cielo, no hay probabilidad de que llueva. Hoy no hace viento y no hay nubes en cielo. Entonces, Manuel cortará el cesped.*

Llamemos:

- p: *"Hay probabilidad de lluvia"*
- q: *"Hace viento"*
- n: *"Hay nubes en el cielo"*
- c: *"Manuel cortará el cesped"*

Se trata de ver si, de las hipótesis:

$$\{((p \vee q) \to \neg c), \neg n \to \neg p, \neg q, \neg n\}$$

se puede deducir c. *Al desarrollar la tabla, se comprueba que*

$$\{\neg p, \neg q, \neg n, \neg c\}$$

es un contraejemplo, ya que aunque no llueva y no haga viento, nadie nos permite asegurar que Manuel vaya a cortar el cesped.

Las principales ventajas de las tablas semánticas respecto a las tablas de verdad son:

i) Es menos costoso de aplicar.

ii) Es una buena base para programar demostradores automáticos.

iii) Puede extenderse a otras lógicas más potentes que la lógica de proposiciones, para las cuales el método de las tablas de verdad deja de tener sentido.

iv) En el caso de que el argumento no sea válido las tablas semánticas nos muestran explicitamente un contraejemplo.

1.6. Cuantificadores

Desde el comienzo del tema sabemos que un enunciado del tipo "$x + 2$ es un número par" no es una proposición, ya que, si $x = 1$, entonces el enunciado es falso y, si $x = 2$, el enunciado es verdadero. Este sería un ejemplo de proposición abierta, cuyo valor de verdad o falsedad depende del valor que tome una (o varias variables) que recorren un cierto conjunto llamado dominio.

Por otro lado, el cálculo proposicional no permite trabajar con una infinidad de proposiciones. Por ejemplo, si denotamos la proposición anterior

$$p(x) : \text{“}(x + 2) \text{ es un número par”}$$

y, queremos expresar que $p(x)$ es cierta cuando x es un entero par, diríamos que:

$$p(2) \wedge p(4) \wedge \ldots$$

es cierta; si lo que queremos es significar que una proposición $p(x)$ es cierta para algún valor natural de x, diríamos que:

$$p(1) \vee p(2) \vee \ldots$$

es cierta.

Para solucionar este problema, se utilizan los *cuantificadores*. Supongamos que $p(x)$ es una proposición si la variable x pertenece a un determinado conjunto U llamado dominio. El *cuantificador universal* "$\forall$" se utiliza para construir proposiciones del tipo

- $$\forall x \; p(x)$$

  que se leen "para todo x, $p(x)$", o bien, "para cada", o bien, "para cualquier". Este tipo de proposición es verdadera cuando $p(x)$ es verdadera para cualquier valor x de U. Es falsa, si para algún valor de U, $p(x)$ es falsa. Por ejemplo: "Todos los alumnos de I.I. tienen más de 16 años" es una proposición verdadera. "Todos los alumnos de I.I. de la Universidad de A Coruña nacieron en A Coruña" es una proposición falsa.

- $$\forall x\ \neg p(x)$$

  que se lee "para todo (cada o cualquiera) x, no se verifica $p(x)$". Será verdadera cuando $p(x)$ sea falsa para todos los valores x de U. Será falsa cuando se verifique $p(x)$, para algún valor x de U.

El *cuantificador existencial* "$\exists$" se utiliza para proposiciones del tipo:

- $$\exists x\ p(x)$$

  que se leen "existe x que verifica $p(x)$". Es verdadera cuando $p(x)$ es verdadera para, al menos, un valor x de U. Es falsa cuando, para todo valor x de U, la proposición $p(x)$ es falsa. "Existe un entero x que sumado con 1 nos da 0" es verdadero. "Existe un entero x que sumado con 1 nos da x" es un argumento falso.

- $$\exists x\ \neg p(x)$$

  que se lee "existe un x tal que no se verifica $p(x)$". Es verdadero cuando $p(x)$ es falsa para algún valor x y es falsa cuando todos los valores de x hacen que $p(x)$ sea verdadera.

Las **leyes de Morgan generalizadas** son ciertas cualquiera que sea el universo del discurso y cualquiera que sea el valor de las proposiciones. Estas son:

I) $\neg[\forall x\, p(x)] \Longleftrightarrow \exists x[\neg p(x)]$

II) $\neg[\exists x\, p(x)] \Longleftrightarrow \forall x[\neg p(x)]$

III) $\neg\exists x\,[\neg p(x)] \Longleftrightarrow \forall x\, p(x)$

IV) $\neg\forall x[\neg p(x)] \Longleftrightarrow \exists x\, p(x)$

Ejemplo 13. *Epiménides de Cnosos (siglo V. a. de C.) decía "Todos los cretenses son mentirosos y yo soy cretense, luego miento". Alguien, a la vista de ello, razona como sigue.*

> Si Epiménides mintió en lo que dijo, entonces los cretenses no eran mentirosos, luego Epiménides, por ser cretense, no mintió en lo que dijo. Se llega pues a una contradicción.

¿Es el razonamiento anterior correcto?

> No, ya que la negación de "Todos los cretenses son mentirosos" es que algún cretense no miente, pero no que todos sean no mentirosos.

Ejemplo 14. *Escribir la negación de la siguiente proposición cuantificada:*

$$\forall x \in A, \exists y \in B, \exists z \in C, \forall t \in B \ p(x, y, z, t).$$

Aplicando las reglas anteriores, tenemos:

$$\begin{aligned}
&\neg[\forall x \in A, \exists y \in B, \exists z \in C, \forall t \in B \ p(x, y, z, t)] \\
\iff &\exists x \in A \ \neg[\exists y \in B, \exists z \in C, \forall t \in B \ p(x, y, z, t)] \\
\iff &\exists x \in A, \forall y \in B \ \neg[\exists z \in C, \forall t \in B \ p(x, y, z, t)] \\
\iff &\exists x \in A, \forall y \in B, \forall z \in C \ \neg[\forall t \in B \ p(x, y, z, t)] \\
\iff &\exists x \in A, \forall y \in B, \forall z \in C, \exists t \in B \ \neg p(x, y, z, t)
\end{aligned}$$

1.7. Álgebras de Boole

En la teoría matemática de las Álgebras de Boole se formalizan las propiedades matemáticas de las funciones lógicas. Los valores de verdad (0 y 1) son las dos únicas posibilidades que hay para el objeto más pequeño e indivisible en una computadora digital: el *bit*. En última instancia, todos los programas y datos se pueden reducir a combinaciones de bits. Los circuitos electrónicos permiten que los dispositivos de almacenamiento de bits de las computadoras digitales se comuniquen entre sí. Los elementos básicos de estos circuitos se llaman puertas lógicas y cada tipo de puerta implementa una operación booleana.

Los circuitos digitales más sencillos se llaman **circuitos combinacionales**.

Definición 3. *Un* Álgebra de Boole *es un conjunto* $A = \{a, b, c, \ldots\}$ *con tres operaciones definidas en él: suma* $+$*, producto* $\cdot$ *y complemento o inversión* $\bar{}$ *; de modo que se cumplen los siguientes axiomas:*

A1: *A es cerrado para las tres operaciones:*

$$\forall a, b \in A \text{ se tiene que } a + b \in A,\ a \cdot b \in A,\ \bar{a} \in A$$

A2: *Existen dos elementos distinguidos 0 y 1 en A tales que:*

$$\forall a \in A,\ a + \mathbf{0} = a \qquad y \qquad \forall a \in A,\ a \cdot \mathbf{1} = a$$

A3: *Todo elemento* $a \in A$ *tiene un complemento* $\bar{a} \in A$ *tal que*

$$a + \bar{a} = \mathbf{1} \qquad a \cdot \bar{a} = \mathbf{0}$$

A4: *Las operaciones suma y producto son conmutativas:*

$$\forall a, b \in A, \qquad a + b = b + a, \qquad a \cdot b = b \cdot a$$

A5: *Las operaciones suma y producto son asociativas:*

$$\forall a, b, c \in A, \qquad a + (b + c) = (a + b) + c, \qquad a \cdot (b \cdot c) = (a \cdot b) \cdot c$$

A6: *Las operación suma es distributiva respecto al producto, y viceversa:*

$$\forall a, b, c \in A,$$
$$a + (b \cdot c) = (a + b) \cdot (a + c), \qquad a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$

La estructura $(A, +, \cdot, \bar{\ })$ *se llama un* **Álgebra de Boole**.

El Álgebra de Boole más sencilla es aquella formada por los elementos $\{0, 1\}$ con las operaciones dadas por los operadores lógicos $\wedge$, $\vee$ y $\neg$ definidos anteriormente.

Una propiedad importante de un Álgebra de Boole es el **principio de Dualidad**. Este principio establece que las expresiones algebraicas deducidas a partir de un Álgebra de Boole permanecen válidas si se intercambian los operadores ($+$ por $\cdot$) y los elementos distinguidos (0 por 1).

Otras propiedades de un Álgebra de Boole se recogen en el siguiente resultado:

Proposición 1. *Sea* $(A, +, \cdot, \bar{\ })$ *un Álgebra de Boole. Para cualesquiera* $a, b \in A$ *se verifica:*

I) *Leyes de Idempotencia:* $a + a = a$ *y* $a \cdot a = a$

II) *Leyes de Acotación:* $a + \mathbf{1} = \mathbf{1}$ *y* $a \cdot \mathbf{0} = \mathbf{0}$

III) *Leyes de Absorción:* $a + a \cdot b = a$ *y* $a \cdot (a + b) = a$

IV) *Leyes de Morgan:* $\overline{a + b} = \bar{a} \cdot \bar{b}$ *y* $\overline{a \cdot b} = \bar{a} + \bar{b}$

V) *El elemento* $\bar{a}$ *de* $a \in A$ *es único. Si existe* $b \in A$ *tal que* $a + b = 1$ *y* $a \cdot b = 0$ *entonces* $b = \bar{a}$.

VI) *Involución:* $\overline{(\bar{a})} = a$

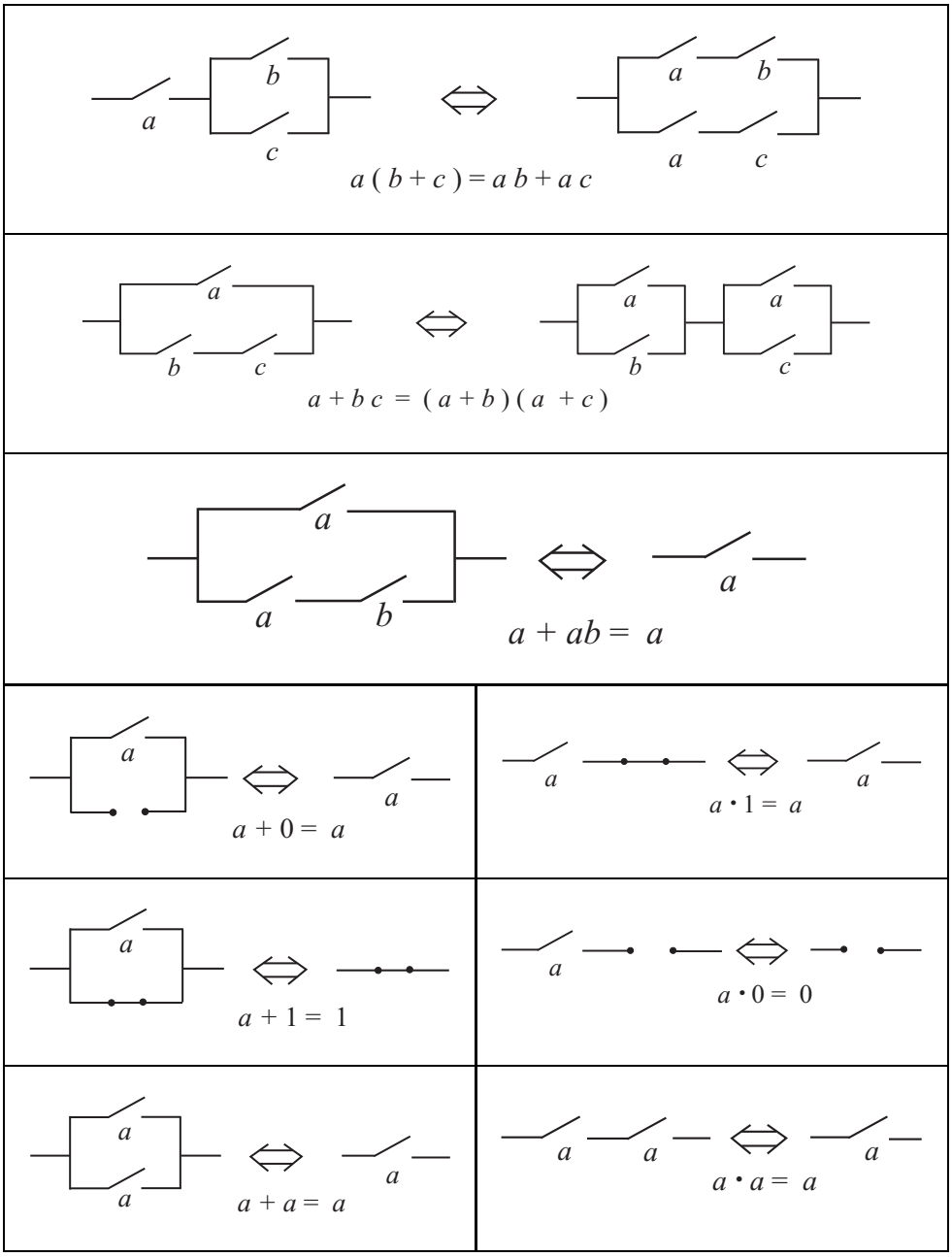
VII) $a + \bar{a} \cdot b = a + b$ *y* $a \cdot (\bar{a} + b) = a \cdot b$.

$$a = a \cdot 1 = a(a + \bar{a}) = a \cdot a + a \cdot \bar{a} = a \cdot a + 0 = a \cdot a$$
$$a = a + 0 = a + a \cdot \bar{a} = (a + a) \cdot (a + \bar{a}) = (a + a) \cdot 1 = a + a$$
$$a + \bar{a}b = (a + \bar{a})(a + b) = 1 \cdot (a + b) = a + b$$

Las propiedades recogidas en la proposición anterior son las equivalentes a las ya estudiadas en lógica proposicional, basta con tener en cuenta la siguiente tabla de equivalencias:

Boole	**Lógica**
$+$	$\vee$
$\cdot$	$\wedge$
$\overline{}$	$\neg$
$\mathbf{0}$	F_0
$\mathbf{1}$	T_0

En la siguiente tabla se presenta la interpretación física de algunas de las propiedades del Álgebra de Boole.



1.8. Funciones de Boole

Definición 4. *Dada un álgebra de Boole binaria* $(\{0,1\},+,\cdot,^-)$*, llamaremos variables booleanas a unos símbolos* x_1*,* y_1*,* z_1*,* x_2*,* y_2*,* z_2*, . . . , que representan a los elementos del conjunto* $\{0,1\}$*.*

Partiendo del alfabeto formado por el conjunto de variables booleanas y los símbolos $+$, $\cdot$ y $^-$, podemos definir un lenguaje que se corresponde con el lenguaje de la lógica de proposiciones cambiando las conectivas $\vee$, $\wedge$ y $\neg$, por los símbolos $+$, $\cdot$ y $^-$, respectivamente.

Las *expresiones booleanas* en los símbolos $x_1, \ldots, x_n$ se definen de manera recursiva como sigue[3]:

$$0,\ 1,\ x_1,\ x_2,\ldots,\ x_n \qquad \text{(variables)}$$

son expresiones booleanas. Si E_1 y E_2 son expresiones booleanas, entonces

a) $\overline{E_1}$ b) $E_1 + E_2$ c) $E_1 \cdot E_2$

son, también, expresiones booleanas. En una expresión booleana en la que no se usan paréntesis para especificar el orden de las operaciones se supone el siguiente orden:

$$1.^{-} \qquad 2.\cdot \qquad 3.+$$

Definición 5. *Una función f de n variables sobre un álgebra de Boole A es una aplicación*

$$f : \overbrace{A \times A \times \cdots \times A}^{n} \to A$$

Si $A = \{0,1\}$ son 2^n las posibles combinaciones de entrada $(x_{n-1}, \ldots, x_1, x_0)$ donde $x_i \in \{0,1\}$.

Una función de Boole puede definirse mediante expresiones del álgebra de Boole o bien dando su tabla de valores.

Ejemplo 15. $f(a,b,c) = \overline{a} + \overline{a}b\overline{c}$

$$\begin{aligned}
f(0,0,0) &= 1 + 1\cdot 0 \cdot 1 = 1\\
f(0,0,1) &= 1 + 1\cdot 0 \cdot 0 = 1\\
f(0,1,0) &= 1 + 1\cdot 1 \cdot 1 = 1\\
f(0,1,1) &= 1 + 1\cdot 1 \cdot 0 = 1\\
f(1,0,0) &= 0 + 0\cdot 0 \cdot 1 = 0\\
f(1,0,1) &= 0 + 0\cdot 0 \cdot 0 = 0\\
f(1,1,0) &= 0 + 0\cdot 1 \cdot 1 = 0\\
f(1,1,1) &= 0 + 0\cdot 1 \cdot 0 = 0
\end{aligned}$$

a	b	c	f
0	*0*	*0*	*1*
0	*0*	*1*	*1*
0	*1*	*0*	*1*
0	*1*	*1*	*1*
1	*0*	*0*	*0*
1	*0*	*1*	*0*
1	*1*	*0*	*0*
1	*1*	*1*	*0*

Dos funciones se dicen iguales si sus tablas de valores son iguales. Por ejemplo, la función anterior f es igual a la función $g(a,b,c) = \overline{a}$. A esta conclusión se puede llegar simplificando la expresión que define f aplicando propiedades del álgebra de Boole: $f(a,b,c) = \overline{a} + \overline{a}b\overline{c} = \overline{a}(1 + b\overline{c}) = \overline{a}$.

Ejemplo 16. *Sean $f_1(a,b,c) = a + b\cdot c$ y $f_2(a,b,c) = a\cdot b + (a+b)\cdot c$ dos funciones booleanas. En la siguiente tabla se recogen sus correspondientes funciones complementarias, su suma y su producto:*

[3]De modo análogo a cómo se definen que las expresiones bien formadas en lógica proposicional

a	**b**	**c**	$\mathbf{f_1}$	$\mathbf{f_2}$	$\overline{\mathbf{f_1}}$	$\overline{\mathbf{f_2}}$	$\mathbf{f_1 + f_2}$	$\mathbf{f_1 \cdot f_2}$
0	*0*	*0*	*0*	*0*	*1*	*1*	*0*	*0*
0	*0*	*1*	*0*	*0*	*1*	*1*	*0*	*0*
0	*1*	*0*	*0*	*0*	*1*	*1*	*0*	*0*
0	*1*	*1*	*1*	*1*	*0*	*0*	*1*	*1*
1	*0*	*0*	*1*	*0*	*0*	*1*	*1*	*0*
1	*0*	*1*	*1*	*1*	*0*	*0*	*1*	*1*
1	*1*	*0*	*1*	*1*	*0*	*0*	*1*	*1*
1	*1*	*1*	*1*	*1*	*0*	*0*	*1*	*1*

Definición 6.

- *Un* **literal** *es una variable booleana (x) o una variable booleana complemento ($\bar{x}$).*

- *Un* **minterm** *en las variables booleanas x_1, x_2, ..., x_n es un producto booleano $y_1 \cdot y_2 \cdot \cdots \cdot y_n$, donde $y_i = x_i$ o $y_i = \overline{x_i}$. Por tanto, un minterm es un producto de n literales con un literal por cada variable.*

Para funciones de tres variables serían minterms: $\overline{x_1}x_2\overline{x_3}$, $x_1x_2x_3$, $x_1\overline{x_2x_3}$. Y no serían minterms: $x_1\overline{x_2}$, $x_1\overline{x_1x_2}x_3$.

Los minterms se denotan de forma simplificada tomando un 1 por cada variable sin negar y 0 por cada variable negada. Los posibles minterms para funciones de dos y tres variables son, respectivamente, los siguientes:

Minterm	Variables	Notación
$\overline{x_1x_2}$	00	0
$\overline{x_1}x_2$	01	1
$x_1\overline{x_2}$	10	2
x_1x_2	11	3

Minterm	Variables	Notación
$\overline{x_1x_2x_3}$	000	0
$\overline{x_1x_2}x_3$	001	1
$\overline{x_1}x_2\overline{x_3}$	010	2
$\overline{x_1}x_2x_3$	011	3
$x_1\overline{x_2x_3}$	100	4
$x_1\overline{x_2}x_3$	101	5
$x_1x_2\overline{x_3}$	110	6
$x_1x_2x_3$	111	7

Definición 7. *Un* **maxterm** *en las variables booleanas x_1, x_2, ..., x_n es una suma booleana $y_1 + y_2 + \ldots + y_n$, donde $y_i = x_i$ o $y_i = \overline{x_i}$. Por tanto, un maxterm es una suma de n literales con un literal por cada variable.*

Para funciones de tres variables serían maxterms: $\overline{x_1}+x_2+\overline{x_3}$, $x_1+x_2+x_3$, $x_1 + \overline{x_2} + \overline{x_3}$. Y no serían maxterms: $x_1 + \overline{x_2}$, $x_1 + \overline{x_1} + \overline{x_2} + x_3$.

Los maxterms se denotan de forma simplificada tomando un 0 por cada variable sin negar y un 1 por cada variable negada. Los posibles maxterms para funciones de dos y tres variables son, respectivamente, los siguientes:

Maxterm	Variables	Notación
$\overline{x_1}+\overline{x_2}$	11	3
$\overline{x_1}+x_2$	10	2
$x_1+\overline{x_2}$	01	1
x_1+x_2	00	0

Maxterm	Variables	Notación
$\overline{x_1}+\overline{x_2}+\overline{x_3}$	111	7
$\overline{x_1}+\overline{x_2}+x_3$	110	6
$\overline{x_1}+x_2+\overline{x_3}$	101	5
$\overline{x_1}+x_2+x_3$	100	4
$x_1+\overline{x_2}+\overline{x_3}$	011	3
$x_1+\overline{x_2}+x_3$	010	2
$x_1+x_2+\overline{x_3}$	001	1
$x_1+x_2+x_3$	000	0

A continuación enunciamos un teorema muy importante relacionado con los maxterms y minterms:

Teorema 2. (Teorema de expansión de Shannon) *Cualquier función binaria puede expresarse en forma de suma de minterms o en forma de producto de maxterms. Estas expresiones, que son únicas, reciben el nombre de* **representaciones canónicas de la función**.

Demostración: Para expresar una función cualquiera $f(x_n, x_{n-1}, \ldots, x_1)$ en forma de suma de minterms, podemos poner la función de la siguiente forma:

$$f(x_n, x_{n-1}, \ldots, x_1) = \overline{x_1} f(x_n, x_{n-1}, \ldots, x_2, 0) + x_1 f(x_n, x_{n-1}, \ldots, x_2, 1)$$

Esta expresión se verifica para los dos valores posibles de x_1. Si $x_1 = 0$ quedará:

$$f(x_n, x_{n-1}, \ldots, x_1) = 1 \cdot f(x_n, x_{n-1}, \ldots, x_2, 0) + 0 \cdot f(x_n, x_{n-1}, \ldots, x_2, 1),$$

y si $x_1 = 1$ tendremos:

$$f(x_n, x_{n-1}, \ldots, x_1) = 0 \cdot f(x_n, x_{n-1}, \ldots, x_2, 0) + 1 \cdot f(x_n, x_{n-1}, \ldots, x_2, 1),$$

Repitiendo el proceso para x_2:

$$\begin{aligned} f(x_n, x_{n-1}, \ldots, x_2, x_1) = {} & \overline{x_2 x_1} f(x_n, x_{n-1}, \ldots, x_3, 0, 0) + \overline{x_2} x_1 f(x_n, x_{n-1}, \ldots, x_3, 0, 1) \\ & + x_2 \overline{x_1} f(x_n, x_{n-1}, \ldots, x_3, 1, 0) + x_2 x_1 f(x_n, x_{n-1}, \ldots, x_3, 1, 1) \end{aligned}$$

y repitiendo el proceso para las n variables:

$$\begin{aligned} f(x_n, x_{n-1}, \ldots, x_2, x_1) = {} & \overline{x_n} \cdots \overline{x_2 x_1} f(0, 0, \ldots, 0, 0) + \overline{x_n} \cdots \overline{x_2} x_1 f(0, \ldots, 0, 1) \\ & + \overline{x_n} \cdots x_2 \overline{x_1} f(0, \ldots, 1, 0) + \overline{x_n} \cdots x_2 x_1 f(0, \ldots, 0, 1, 1) \\ & + \cdots + x_n \cdots \overline{x_2 x_1} f(1, \ldots, 0, 0) + x_n \cdots \overline{x_2} x_1 f(1, \ldots, 0, 1) \\ & x_n \cdots x_2 \overline{x_1} f(1, \ldots, 1, 0) + x_n \cdots x_2 x_1 f(1, \ldots, 1, 1) \end{aligned}$$

Por tanto, en la expresión de la función van a aparecer aquellos minterms que representan combinaciones de entradas para las cuales la función vale 1, los demás desaparecen. ■

De forma similar se haría la demostración para expresar una función en forma de producto de maxterms. En este caso aparecen aquellos maxterms

que representan combinaciones de entrada para los cuales la función vale 0, los demás maxterms desaparecen.

Consideremos ahora la siguiente tabla donde se recogen los valores correspondientes a dos funciones booleanas $F(x, y, z)$ y $G(x, y, z)$

	x	**y**	**z**	**F**	**G**
0	0	0	0	1	0
1	0	0	1	0	1
2	0	1	0	1	1
3	0	1	1	1	0
4	1	0	0	0	0
5	1	0	1	1	0
6	1	1	0	0	1
7	1	1	1	0	1

Para cada una de las funciones, podemos construir una expresión booleana con este conjunto de valores prefijado sin más que realizar la suma booleana de distintos minterms:

$$F(x, y, z) = \bar{x}\bar{y}\bar{z} + \bar{x}y\bar{z} + \bar{x}yz + x\bar{y}z = \sum m(0, 2, 3, 5)$$
$$G(x, y, z) = \bar{x}\bar{y}z + \bar{x}y\bar{z} + xy\bar{z} + xyz = \sum m(1, 2, 6, 7)$$

A la suma de mintérms que representa la función se le llama **forma normal disyuntiva** de la función booleana.

También podemos construir una expresión booleana del conjunto de valores anterior realizando el producto booleano de distintos maxterms:

$$F(x, y, z) = (x + y + \bar{z})\,(\bar{x} + y + z)\,(\bar{x} + \bar{y} + z)\,(\bar{x} + \bar{y} + \bar{z}) = \prod M(1, 4, 6, 7)$$
$$G(x, y, z) = (x + y + z)\,(x + \bar{y} + \bar{z})\,(\bar{x} + y + z)\,(\bar{x} + y + \bar{z}) = \prod M(0, 3, 4, 5)$$

Al producto de maxterms que representa la función se le llama **forma normal conjuntiva** de la función booleana. Este desarrollo se puede obtener, también, a partir de la forma normal disyuntiva tomando el dual.

También, dada una función booleana, podemos calcular su forma normal disyuntiva, sin más que aplicar propiedades del álgebra de Boole. Calculemos, por ejemplo, la forma normal disyuntiva de la función $F(x, y, z) = (x + y)\bar{z}$

$$\begin{aligned}
F(x, y, z) = (x + y)\bar{z} &= x\bar{z} + y\bar{z} && \text{Prop. distributiva}\\
&= x\mathbf{1}\bar{z} + y\mathbf{1}\bar{z} && \text{Prop. de identidad}\\
&= x(y + \bar{y})\bar{z} + y(x + \bar{x})\bar{z} && \text{Existencia de inverso}\\
&= xy\bar{z} + x\bar{y}\bar{z} + xy\bar{z} + \bar{x}y\bar{z} && \text{Distributiva, Conmutativa}\\
&= xy\bar{z} + x\bar{y}\bar{z} + \bar{x}y\bar{z} && \text{Prop. de idempotencia}
\end{aligned}$$

En la tabla $F(x, y, z) = m(2, 4, 6)$ se corresponde con las filas tercera, quinta y séptima

x	y	z	F	
0	0	0	0	0
0	0	1	0	1
0	1	0	1	2
0	1	1	0	3
1	0	0	1	4
1	0	1	0	5
1	1	0	1	6
1	1	1	0	7

Luego, el teorema de expansión de Shannon demuestra que existe una relación sencilla entre la tabla de verdad de una función lógica de Boole y su representación canónica:

Forma normal	Método de obtención	Convenio
Disyuntiva	Suma de productos de variables cuyas combinaciones hacen 1 la función	0 variable negada 1 variable sin negar
Conjuntiva	Producto de sumas de variables cuyas combinaciones hacen 0 la función	0 variable sin negar 1 variable negada

Tabla 1

Vamos a expresar en forma de suma de minterms y en forma de producto de maxterms la siguiente función f:

	a	b	c	f	Minterm	Maxterm
0	0	0	0	0		$a+b+c$
1	0	0	1	0		$a+b+\bar{c}$
2	0	1	0	1	$\bar{a}b\bar{c}$	
3	0	1	1	0		$a+\bar{b}+\bar{c}$
4	1	0	0	0		$\bar{a}+b+c$
5	1	0	1	1	$a\bar{b}c$	
6	1	1	0	0		$\bar{a}+\bar{b}+c$
7	1	1	1	1	abc	

Las representaciones canónicas son, por tanto:

$$f(a,b,c) = \sum m(2,5,7) = \bar{a}b\bar{c} + a\bar{b}c + abc$$

$$f(a,b,c) = \prod M(0,1,3,4,6)$$
$$= (a+b+c)(a+b+\bar{c})(a+\bar{b}+\bar{c})(\bar{a}+b+c)(\bar{a}+\bar{b}+c)$$

1.9. Puertas lógicas básicas

Estos dispositivos de estado sólido, son los bloques elementales para la construcción de circuitos lógicos y son capaces de cambiar los niveles de voltaje (bits). El nombre de *puertas* responde al hecho de que pueden tener

una o varias entradas pero una sóla salida, y esta salida puede tomar el valor lógico 0 o 1, dependiendo de los valores lógicos que tengan las entradas.

Comenzaremos por analizar las tres puertas básicas: AND (y), OR (o) y NOT (no). Su función se corresponde exactamente con la que, simbólicamente, realizan los operadores lógicos $\wedge$, $\vee$ y $\neg$, respectivamente, en lógica de proposiciones.

Si representamos por x_1 y x_2 las entradas de la **puerta** AND, donde x_1 y x_2 son bits, la salida que produce se denota por $x_1 \wedge x_2$, donde:

$$x_1 \wedge x_2 = \begin{cases} 1 & \text{si} \quad x_1 = 1 \text{ y } x_2 = 1 \\ 0 & \text{en otro caso} \end{cases}$$

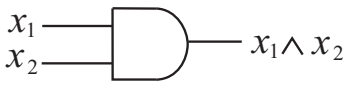


Figura 1. Puerta AND

Una **puerta** OR recibe entradas x_1 y x_2, donde x_1 y x_2 son bits, y produce una salida denotada por $x_1 \vee x_2$, donde

$$x_1 \vee x_2 = \begin{cases} 1 & \text{si} \quad x_1 = 1 \text{ o } x_2 = 1 \\ 0 & \text{en otro caso} \end{cases}$$

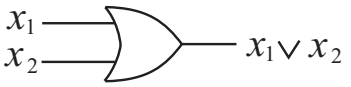


Figura 2. Puerta OR

Una **puerta** NOT (o *inversor*) recibe una entrada x, donde x es un bit, y produce una salida denotada por $\bar{x}$, donde

$$\bar{x} = \begin{cases} 1 & \text{si} \quad x = 0 \\ 0 & \text{si} \quad x = 1 \end{cases}$$

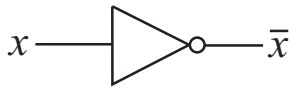


Figura 3. Puerta NOT

La **tabla lógica** de un circuito combinatorio lista todas las entradas posibles junto con las salidas producidas. Las tablas lógicas correspondientes a los circuitos AND, OR y NOT básicos (Figuras 1, 2 y 3, respectivamente) son:

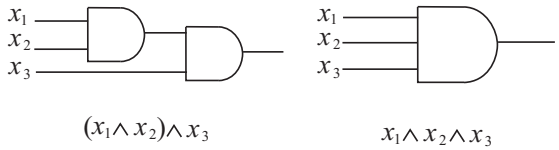
$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_1 \wedge x_2}$
0	0	0
0	1	0
1	1	1
1	0	0

$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_1 \wedge x_2}$
0	0	0
0	1	1
1	1	1
1	0	1

$\mathbf{x}$	$\mathbf{\bar{x}}$
0	1
1	0

Obviamente, son las tablas de verdad de los operadores lógicos a los que corresponden.

Las puertas OR y AND pueden tener más de dos entradas pues las operaciones que realizan son, formalmente, las mismas operaciones conocidas del álgebra de Boole, y, por tanto, tienen las mismas propiedades, y, concretamente, la propiedad asociativa. Esto nos permite decir, por ejemplo, que la función de una puerta AND de tres entradas, es la misma del cirucito formado por dos puertas AND de dos entradas conectadas según indica la figura:

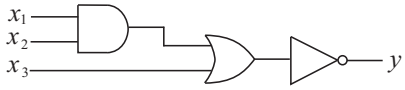


Las puertas se pueden interconectar para obtener circuitos combinatorios más amplios. La salida de una puerta cualquiera puede servir de entrada a una o varias puertas, pero nunca pueden conectarse juntas dos o más salidas. Como existe una correspondencia biunívoca entre las operaciones que realizan las puertas y los operadores del álgebra de Boole, si representamos por variables (x_i, y_i, etc.) los valores lógicos de las entradas del circuito, podremos escribir una fórmula para representar cada salida, en la que intervendrán esas variables y los símbolos " $^-$ ", "+" y "·". Recíprocamente, se puede construir el circuito combinacional correspondiente a una función booleana, para ello se va dibujando el circuito en el orden en el que se realizan las operaciones.

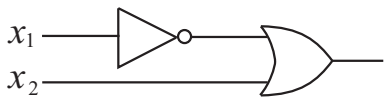
Ejemplo 17. *La función booleana*

$$F(x_1, x_2, x_3) = \overline{(x_1 \cdot x_2) + x_3}$$

se corresponde con el circuito lógico siguiente:



Los operadores lógicos condicional ($\rightarrow$) y bicondicional ($\leftrightarrow$) estudiados en el tema anterior, también se pueden implementar utilizando circuitos combinacionales. El circuito

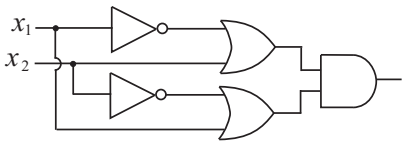


con salida $\neg x_1 \vee x_2$, se corresponde con el operador lógico condicional ($x_1 \rightarrow x_2$).

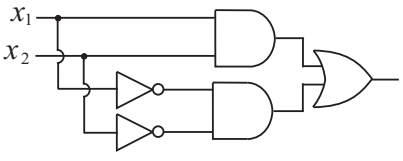
En el caso del operador lógico bicondicional

$$x_1 \leftrightarrow x_2 \Leftrightarrow (x_1 \rightarrow x_2) \wedge (x_2 \rightarrow x_1) \Leftrightarrow (x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2)$$

se puede reprentar por

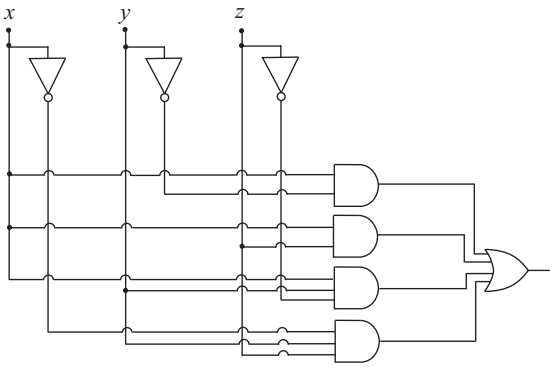


o bien por

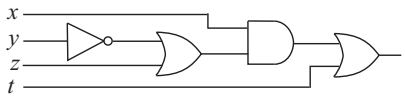


A continuación se dibujan algunos circuitos lógicos, cada uno con la función que describe su comportamiento:

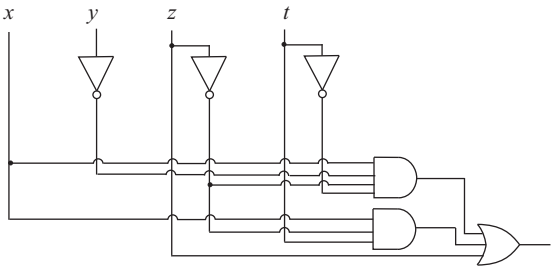
La función $F(x, y, z) = x\bar{y} + xz + \bar{x}y\bar{z} + \bar{x}zy$ se corresponde con el circuito



La función $F(x, y, z, t) = x(\bar{y} + z)t$ se corresponde con el circuito



Por último, la función $F(x, y, z, t) = x\bar{y}\bar{z}\bar{t} + x\bar{z}t + z$ se corresponde con el circuito



Definición 8. *Aquellas combinaciones de entradas para las cuales no nos interesa el valor que pueda tomar la salida se llaman* **indiferencias** *y se dice que las funciones que incluyen indiferencias están* **incompletamente especificadas**.

Esto ocurre en algunas ocasiones cuando el circuito que se diseña forma parte de un sistema mayor en el que ciertas entradas se producen sólo en circunstancias tales que la salida del circuito no influirá en el sistema general. Siempre que la salida no tenga ningún efecto, es evidente que no importa si la salida es un 0 ó un 1. Otra posibilidad es que ciertas combinaciones de entrada no ocurran jamás debido a varias restricciones externas. El circuito responderá de alguna forma a cualquier entrada, pero como esas entradas no se producirán nunca no importa si el circuito final responde con una salida de 0 ó 1.

Las indiferencias se indican en la tabla de verdad anotanto un "–"como valor funcional, en vez de un 0 ó un 1. Este símbolo "–"no es estándar y también son utilizados otros como "$\emptyset$", "**d** "y "**X** ".

Supongamos por ejemplo, que queremos construir un sistema que produzca como salida el cuadrado de los números del 0 al 4. Necesitaremos tres bits como entradas para representar del o al 4 y cinco bits de salida para representar el mayor valor que es $4^2 = 16$. La tabla de verdad de este sistema es la siguiente:

a	b	c	f_4	f_3	f_2	f_1	f_0
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1
0	1	0	0	0	1	0	0
0	1	1	0	1	0	0	1
1	0	0	1	0	0	0	0
1	0	1	–	–	–	–	–
1	1	0	–	–	–	–	–
1	1	1	–	–	–	–	–

Las representaciones canónicas para estas funciones serían, en notación simplificada (en este caso las indiferencias las indicamos con "d" o "D" como un término a parte de los minterms o maxterms):

$$f_4(a,b,c) = \sum m(4) + \sum d(5,6,7)$$
$$f_3(a,b,c) = \sum m(3) + \sum d(5,6,7)$$
$$f_2(a,b,c) = \sum m(2) + \sum d(5,6,7)$$
$$f_1(a,b,c) = \sum d(5,6,7)$$
$$f_0(a,b,c) = \sum m(1,3) + \sum d(5,6,7)$$

o bien,

$$f_4(a,b,c) = \prod M(0,1,2,3) \prod D(5,6,7)$$
$$f_3(a,b,c) = \prod M(0,1,2,4) \prod D(5,6,7)$$
$$f_2(a,b,c) = \prod M(0,1,3,4) \prod D(5,6,7)$$
$$f_1(a,b,c) = \prod M(0,1,2,3,4) \prod D(5,6,7)$$
$$f_0(a,b,c) = \prod M(0,2,4) \prod D(5,6,7)$$

Una indiferencia puede ser considerada como salida 0 ó 1 según convenga. Por ejemplo, en la función f_0, podemos considerar $f_0(1,0,1) = 1$, $f_0(1,1,0) = 0$ y $f_0(1,1,1) = 1$, entonces desarrollando la función en forma de suma de minterms:

$$f_0(a,b,c) = \sum m(1,3,5,7) = \bar{a}\bar{b}c + \bar{a}bc + a\bar{b}c + abc = \bar{a}c\left(\bar{b}+b\right) + ac\left(\bar{b}+b\right)$$
$$= c\left(\bar{a}+a\right) = c$$

Con este ejemplo podemos ver que las indiferencias van a ser muy útiles a la hora de simplificar las expresiones algebraica de una función.

1.10. Minimización de funciones. Minimización de circuitos

Minimizar una función es obtener la expresión más simplificada posible para dicha función. En general, la función minimizada no es única. Una expresión de una función en forma de suma de producto será mínima si:

- No existe otra expresión de la función con menor número de términos producto.
- Cualquier otra expresión con igual número de términos producto tendrá más variables dentro de los productos.

Si la expresión es un producto de términos suma será mínima si cumple las condiciones anteriores cambiando la palabra producto por suma.

Vamos a ver ahora una serie de conceptos que nos ayudarán a obtener las expresiones mínimas de las funciones.

Definición 9. *Dos minterms (maxterms) constituyen un* **implicante de primer orden** *si la expresión de uno de ellos puede obtenerse a partir de la del otro negando una sola variable.*

La suma de los minterms que componen un implicante es simplificable (lo mismo que el producto de maxterms). Por ejemplo, para funciones de tres variables $\bar{a}\bar{b}c$ y $\bar{a}bc$; cuando se suman

$$\bar{a}\bar{b}c + \bar{a}bc = \bar{a}c\left(\bar{b} + b\right) = \bar{a}c \quad \text{(implicante } \bar{a}c\text{)}.$$

Definición 10. *Dos implicantes de primer orden constituyen un* **implicante de segundo orden** *si la expresión de uno de ellos puede obtenerse a partir de la del otro negando una sola variable.*

Por ejemplo $\bar{a}\bar{b}c$ y $\bar{a}bc$ (implicante $\bar{a}c$), $\bar{a}\bar{b}\bar{c}$ y $\bar{a}b\bar{c}$ (implicante $\bar{a}\bar{c}$); si los sumamos

$$\bar{a}c + \bar{a}\bar{c} = \bar{a}\left(c + \bar{c}\right) = \bar{a} \quad \text{(implicante de segundo orden}\bar{a}\text{)}.$$

Definición 11. *Dos implicantes de segundo orden constituyen un* **implicante de tercer orden** *si la expresión de uno de ellos puede obtenerse a partir de la del otro negando una sola variable.*

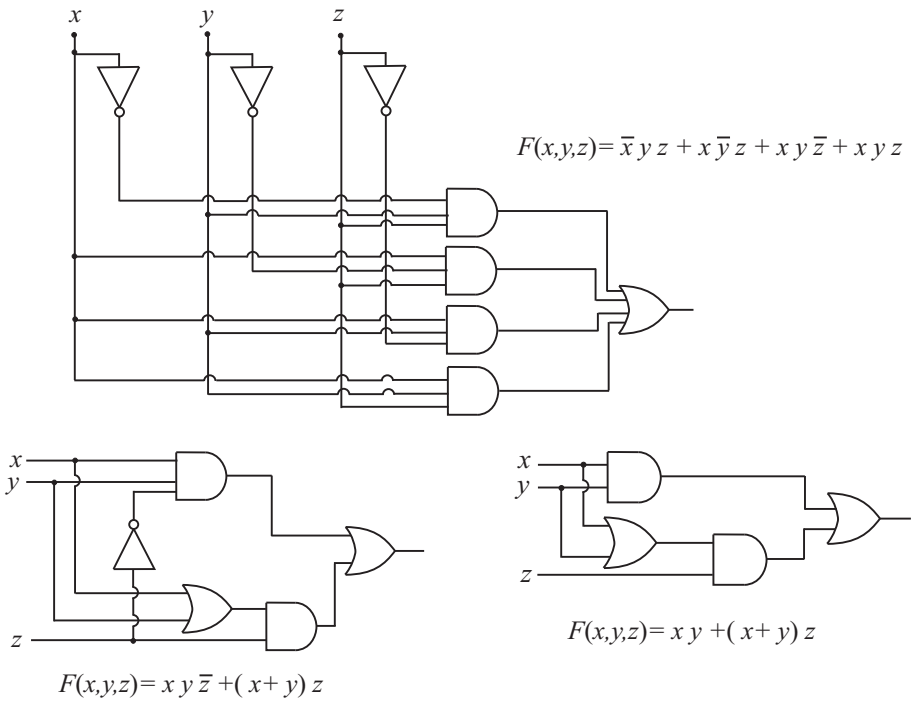
Por ejemplo $\bar{a}\bar{b}c$, $\bar{a}bc$, $\bar{a}\bar{b}\bar{c}$ y $\bar{a}b\bar{c}$ (implicante $\bar{a}$), $a\bar{b}\bar{c}$, $a\bar{b}c$, $ab\bar{c}$ y abc (implicante a); si los sumamos $\bar{a} + a = 1$.

Esto se generaliza sin dificultad para funciones de cualquier número de variables y para implicantes de cualquier orden, es decir, dos implicantes de orden n constituyen uno d orden $n+1$ si la expresión de uno de ellos se puede obtener a partir del otro negando una variable. Es conveniente notar que un implicante de primer orden simplifica una variable, uno de segundo dos, uno de tercero tres, etc. También es importante recalcar que un implicante de orden n está compuesto por 2^n minterms (maxterms), es decir, uno de primer orden está formado por dos minterms (maxterms), uno de segundo por cuatro, uno de tercer orden por ocho, etc.

A la hora de diseñar un circuito lógico para realizar una determinada función, hay que especificar, en primer lugar, la función y la tabla de verdad correspondiente. Como se vio en ejemplos anteriores, la misma tabla de verdad puede realizarse mediante circuitos diferentes; por ejemplo la tabla

x	y	z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

es la correspondiente a cada uno de los tres circuitos siguientes:



Es obvio (por el coste, tamaño, fiabilidad, etc.) que interesará encontrar el circuito que, respetando las especificaciones, tenga el menor número posible de puertas.

Existen dos procedimientos básicos a la hora de simplificar las funciones booleanas y, en consecuencia, los circuitos lógicos correspondientes.

Aquí vamos a ver un método gráfico para simplificar funciones booleanas y, en consecuencia, los circuitos lógicos correspondientes. Este método es el más sencillo aunque, prácticamente, deja de tener utilidad para formas booleanas con más de seis variables: el método de las tablas de Karnaugh.

1.11. Diagramas de Karnaugh

Una tabla o diagrama de Karnaugh no es otra cosa que una presentación alternativa de la misma información contenida en una tabla de verdad. La tabla de Karnaugh es de doble entrada, y tiene las asignaciones colocadas de tal modo que las que corresponden a productos canónicos adyacentes están físicamente contiguas.

Este método se desarrolló en la década de los cincuenta para ayudar a minimizar los circuitos de forma manual.

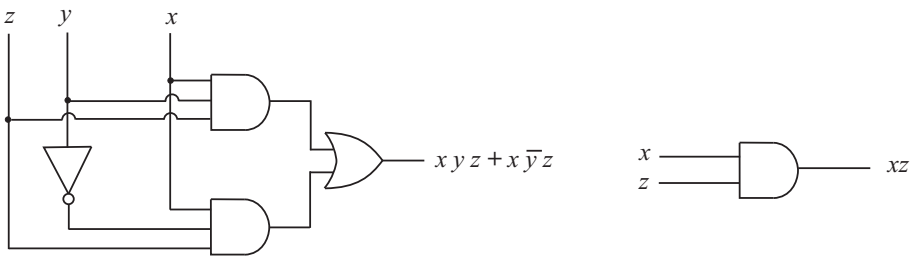
La eficiencia de un circuito combinacional depende del número de puertas que tenga y de la disposición de éstas. El proceso de diseñar un circuito combinacional comienza con la tabla que especifica las salidas para cada combinación de valores de entrada. Para obtener un conjunto de puertas lógicas que implemente este circuito siempre podemos usar la forma normal disyuntiva del circuito. Sin embargo, la forma normal disyuntiva puede tener muchos más sumandos de los necesarios. Se pueden combinar entre sí dos sumandos de una forma normal disyuntiva que difieren en una sola variable de manera que en un sumando aparezca dicha variable y en el otro sumando lo haga su complementario. Por ejemplo, consideremos el circuito cuya tabla asociada es

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

La forma normal disyuntiva de este circuito es $xyz + x\bar{y}z$. Pero

$$xyz + x\bar{y}z = (y + \bar{y})xz = \mathbf{1} \cdot xz = xz$$

Luego, xz es una expresión booleana con menos operadores que representa igualmente al circuito. A continuación mostramos las dos implementaciones diferentes de este circuito:



Este ejemplo muestra que combinar sumandos de la forma normal disyuntiva de un circuito produce una expresión más sencilla del circuito. Minimizar una función booleana permite construir circuitos con el menor número posible de puertas y el menor número posible de entradas a las puertas AND y OR del circuito.

Los K-diagramas proporcionan un método visual para simplificar una forma normal disyuntiva, pero no son adecuados para automatizar este proceso. Un mapa de Karnaugh está constituido por una cuadrícula en forma de encasillado cuyo número de casillas depende del número de variables que

tenga la función a simplificar. Cada una de las casillas representa las distintas combinaciones de las variables que puedan existir.

En cada cuadrado representaremos el valor que toma la función para la combinación de variables que le corresponde, y como hay una correspondencia uno a uno entre los cuadrados y las distintas combinaciones de entrada, el mapa de Karnaugh va a ser otra forma de especificar una función (es un diagrama visual de la tabla de verdad).

El mapa de Karnaugh para una función de 2 variables es:

$a \backslash b$	0	1
0	0	1
1	2	3

$a \backslash b$	0	1
0	$\overline{a}\,\overline{b}$	$\overline{a}\,b$
1	$a\,\overline{b}$	$a\,b$

$a \backslash b$	0	1
0	$a+b$	$a+\overline{b}$
1	$\overline{a}+b$	$\overline{a}+\overline{b}$

2 variables

El mapa de Karnaugh para una función de 3 variables es:

$a \backslash bc$	00	01	11	10
0	0	1	3	2
1	4	5	7	6

3 variables

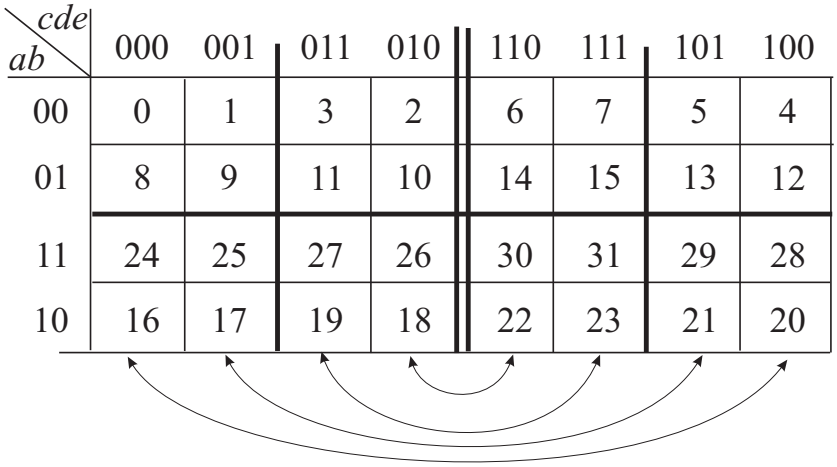
$a \backslash bc$	00	01	11	10
0	$a+b+c$	$a+b+\overline{c}$	$a+\overline{b}+\overline{c}$	$a+\overline{b}+c$
1	$\overline{a}+b+c$	$\overline{a}+b+\overline{c}$	$\overline{a}+\overline{b}+\overline{c}$	$\overline{a}+\overline{b}+c$

$a \backslash bc$	00	01	11	10
0	$\overline{a}\,\overline{b}\overline{c}$	$\overline{a}\,\overline{b}c$	$\overline{a}\,bc$	$\overline{a}\,b\overline{c}$
1	$a\,\overline{b}\overline{c}$	$a\,\overline{b}c$	$a\,bc$	$a\,b\overline{c}$

Los mapas de Karnaugh para funciones de 4 y 5 variables son:

$ab \backslash cd$	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

4 variables



5 variables

En algunos de los casos anteriores, hemos indicado el minterm (maxterm) que correspondería a cada casilla.

Importante: el orden para los pares de variables es, **obligatoriamente** 00, 01, 11 y 10 y para el grupo de tres variables 000, 001, 011, 010, 110, 111, 101 y 100.

En el caso de querer obtener una expresión mínima como suma de términos producto, en el mapa colocaremos los unos de la función, además de las indiferencias, omitiendo los ceros. Si lo que nos proponemos es calcular la expresión mínima como producto de términos suma, entonces pondremos los ceros y las indiferencias, excluyendo los unos.

El segundo paso en la minimización es localizar los implicantes de la función en el mapa de Karnaugh. Los implicantes deben estar formados por celdas adyacentes y pueden contener tantos unos como indiferencias (deben contener al menos un 1).

Los implicantes de primer orden ocupan dos casillas adyacentes. La adyacencia de dos casillas puede ser horizontal o vertical, nunca diagonal. Las casillas de la primera y última fila son adyacentes y lo mismo ocurre con la primera y última columna. Un implicante de segundo orden está formado por dos implicantes de primer orden adyacentes (2^2 casillas), uno de tercero por dos de segundo orden adyacentes (2^3 casillas), etc.

Los mapas para funciones de cinco variables se construyen a partir de mapas de cuatro variables. Las celdas simétricas respecto al eje central representan también adyacencias, como se indica en la figura.

Dos celdas **adyacentes** del mapa son aquellas cuyos minterms difieren en, exactamente, un literal.

Siempre que haya unos en dos celdas adyacentes del K-diagrama, los minterms representados por estas celdas se pueden combinar entre sí dando lugar a un término que depende sólo de una de las variables (En el ejemplo anterior $xy + \bar{x}y = x$).

Se procede rodeando con un círculo los bloques de celdas del K-diagrama que representan minterms que se pueden combinar y después calculamos la correspondiente suma de productos.

De modo análogo al caso de dos variables, se dice que dos celdas son **adyacentes** si los minterms que representan difieren en, exactamente, un literal.

En el caso de querer obtener una expresión mínima como suma de términos producto, en el mapa colocaremos los unos de la función, además de las indiferencias, omitiendo los ceros. Si lo que nos proponemos es calcular la expresión mínima como producto de términos suma, entonces pondremos los ceros y las indiferencias, excluyendo los unos.

El segundo paso en la minimización es localizar los implicantes de la función en el mapa de Karnaugh. Los implicantes deben estar formados por celdas adyacentes y pueden contener tanto unos como indiferencias (deben contener al menos un 1). Los implicantes de primer orden ocupan dos casillas adyacentes. La adayacencia de dos casillas puede ser horizontal o vertical, nunca diagonal. Las casillas de la primera y última fila son adyacentes y lo mismo ocurre con la primera y la última columna. Un implicante de segundo orden está formado por dos implicantes de primer orden adyacentes, uno de tercero por dos de segundo adyacentes, etc.

Por último, hay que destacar que cuando vayamos a representar una función booleana, ésta tiene que estar en su forma canónica (minterms y maxterms) completa y, por tanto, todos los términos han de contener todas las variables que intervienen en la función. En el caso de tener que representar funciones incompletas, habrá que obtener, previamente, la forma canónica completa.

• Minimización en suma de productos

El objetivo es incluir todos los unos del mapa con el menor número de implicantes y del mayor orden posible. Las indiferencias ayudan a completar los implicantes. No importa que los implicantes se solapen unos a otros, lo que si interesa es no incluir más implicantes de los necesarios.

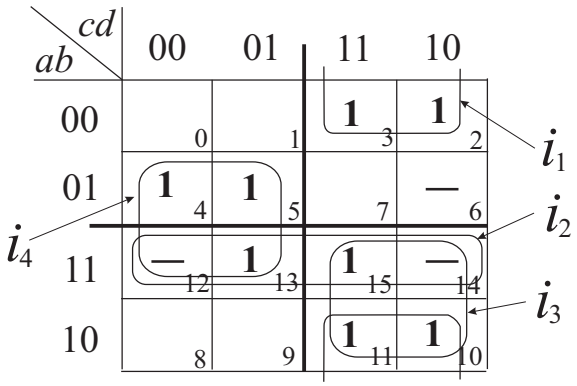
El procedimiento a seguir puede resumirse en las siguientes reglas:

- Extraer todos los implicantes de cualquier orden que no estén incluidos en otros y que contengan al menos un 1 (se llaman *implicantes primos* de la función. Se empieza por los de mayor orden.
- Escoger de los implicantes primos aquellos que contengan unos en exclusiva (son los *implicantes primos esenciales* de la función).
- En casi de que quede algún 1 por cubrir, escoger el menor número de implicantes (y del mayor orden posible) necesario para cubrir todos los unos.

Simplifiquemos, por ejemplo, la función:

$$f(a,b,c,d) = \sum m(2,3,4,5,10,11,13,15) + \sum d(6,12,14)$$

Una vez representada la función en el mapa de Karnaugh, extraemos los implicantes primos de la función:



donde se consideran las indiferencias 12 y 14 iguales a 1 ($f(1,1,0,0) = 1$ y $f(1,1,1,0) = 1$) y la tercera indiferencia (casilla 6) igual a 0.

Los implicantes primos esenciales son:

$$i_1 :\ b = 0,\ c = 1 \quad i_1 = \bar{b}c \qquad i_4 :\ b = 1,\ c = 0 \quad i_4 = b\bar{c}$$

nos queda por cubrir el 1 correspondiente a la casilla 15 que se puede cubrir con i_3 o con i_2:

Si se cubre con $i_3 : a = 1,\ b = 1$, luego $i_3 = ab$

Si se cubre con $i_2 : a = 1,\ c = 1$, luego $i_2 = ac$

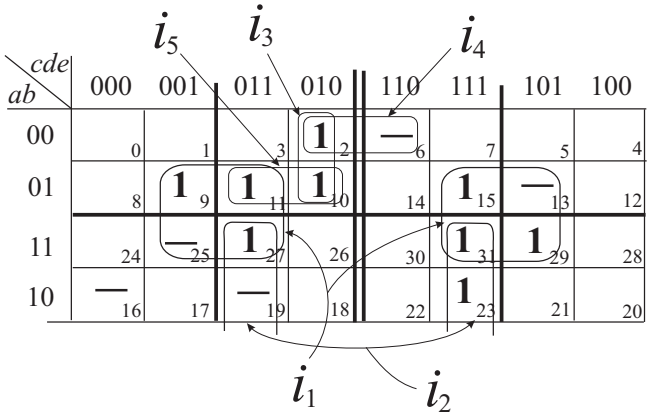
Así hay dos expresiones alternativas para la función:

- Implicantes i_1, i_4 e i_3: $f(a,b,c,d) = \bar{b}c + b\bar{c} + ab$.
- Implicantes i_1, i_4 e i_2: $f(a,b,c,d) = \bar{b}c + b\bar{c} + ac$.

Consideremos ahora una función de cinco variables:

$$f(a,b,c,d,e) = \sum m(2,9,10,11,15,23,27,29,31) + \sum d(6,13,16,19,25)$$

Representamos la función en el mapa de Karanugh y extraemos los implicantes primos de la función:



Los implicantes primos esenciales son:

$$i_1 :\ b = 1,\ e = 1 \quad i_1 = bec \qquad i_2 :\ a = d = e = 1 \quad i_2 = ade$$

nos quedan por cubrir las casillas 2 y 10 que se puede cubrir con un único implicante i_3: $a = c = e = 0$, $d = 1$ luego $i_3 = \bar{a}\bar{c}\bar{e}d$. Por tanto, la expresión simplificada de f es $f(a, b, c, d, e) = bec + ade + \bar{a}\bar{c}\bar{e}d$.

• Minimización en producto de sumas

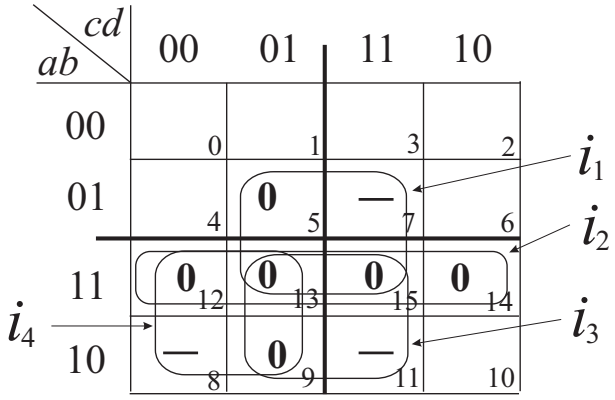
El método es totalmente equivalente al anterior. Las modificaciones que han de realizarse son las siguientes:

- Colocar en el mapa de Karnaugh los ceros y las indiferencias de la función.
- Los implicantes estarán formados por celdas adyacentes que pueden contener tanto ceros como indiferencias (deben contener al menos un cero).
- La obtención de la expresión de cada implicante es la siguiente: por cada variable que no cambien en un implicante, si es 0 tomar la variable tal cual y si es 1 tomar la variable negada. Sumar las variables obtenidas de esta forma.

Veamos el siguiente ejemplo de cuatro variables:

$$\begin{aligned} f(a, b, c, d) &= \sum m(0, 1, 2, 3, 4, 6, 10) + \sum d(7, 8, 11) \\ &= \prod M(5, 9, 12, 13, 14, 15) \prod D(7, 8, 11) \end{aligned}$$

Representamos la función en el mapa de Karnaugh y buscamos los implicantes primos



Los implicantes primos esenciales son:

$$i_1 :\ b = 1,\ d = 1 \quad i_1 = \bar{b} + \bar{c} \qquad i_2 :\ a = 1,\ b = 1 \quad i_2 = \bar{a} + \bar{b}.$$

nos queda por cubrir el 0 correspondiente a la casilla 9 que se puede cubrir con el implicante primo i_3 o con el implicante primo i_4:

Si se cubre con $i_3 : a = 1,\ d = 1$, luego $i_3 = \bar{a} + \bar{d}$

Si se cubre con $i_4 : a = 1,\ c = 0$, luego $i_2 = \bar{a} + c$

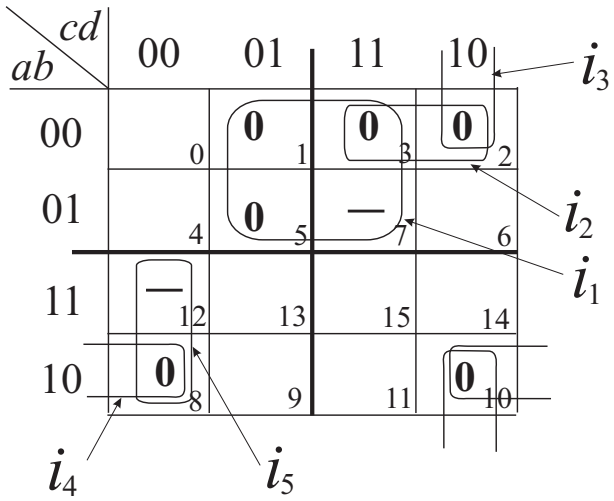
Así hay dos expresiones alternativas para la función:

- Implicantes i_1, i_2 e i_3: $f(a,b,c,d) = \left(\bar{b}+\bar{c}\right)\left(\bar{a}+\bar{b}\right)\left(\bar{a}+\bar{d}\right)$.
- Implicantes i_1, i_2 e i_4: $f(a,b,c,d) = \left(\bar{b}+\bar{c}\right)\left(\bar{a}+\bar{b}\right)(\bar{a}+c)$.

Por último, consideremos otro ejemplo en cuatro variables

$$f(a,b,c,d) = \sum m(0,4,6,9,11,13,14,15) + \sum d(7,12) = \prod M(1,2,3,5,8,10) \prod D(7,12)$$

Representamos la función en el mapa de Karnaugh y buscamos los implicantes primos



El único implicante primo esencial es i_1, siendo $a = 0$ y $d = 1$; luego $i_1 = a+\bar{d}$. Nos quedan por cubrir los ceros correspondientes a las casillas 2, 8 y 10 que se pueden cubrir de la forma siguiente:

- Si se cubre el 0 de la casilla 2 con $i_2 = a + b + \bar{c}$ ($a = b = 0$, $c = 1$) podemos cubrir los dos ceros restantes, casillas 8 y 10, con un único implicante $i_4 = \bar{a} + b + d$, ($a = 1$, $b = d = 0$).
- Si se cubre el 0 de la casilla 2 con $i_3 = b + \bar{c} + d$, ($b = 0$, $c = 1$, $d = 0$), nos queda por cubrir la casilla 8 que se puede hacer con el implicante $i_4 = \bar{a}+b+d$ o bien con el implicante $i_5 = \bar{a}+c+d$, ($a = 1$, $c = d = 0$).

Así hay tres expresiones alternativas para la función:

- Implicantes i_1, i_2 e i_4: $f(a,b,c,d) = \left(a+\bar{d}\right)(a+b+\bar{c})(\bar{a}+b+d)$.
- Implicantes i_1, i_3 e i_4: $f(a,b,c,d) = \left(a+\bar{d}\right)(b+\bar{c}+d)(\bar{a}+b+d)$.
- Implicantes i_1, i_3 e i_5: $f(a,b,c,d) = \left(a+\bar{d}\right)(b+\bar{c}+d)(\bar{a}+c+d)$.

1.12. Completitud funcional

Siempre con el objetivo final de reducir el tamaño y el coste de un circuito, a parte de las puertas vistas, se suelen emplear otras como la puerta NOR y la puerta NAND.

Toda función booleana se puede expresar como una suma booleana de minterms, es decir, toda función booleana se puede representar utilizando los operadores booleanos $+$, $\cdot$ y $^-$. Así, diremos que el conjunto $\{+, \cdot, ^-\}$ es **funcionalmente completo**.

Si utilizamos las leyes de De Morgan, podemos eliminar todas las sumas booleanas utilizando la propiedad

$$x + y = \overline{\bar{x} \cdot \bar{y}}$$

obteniéndose que $\{\cdot, ^-\}$ es funcionalmente completo. De la misma forma

$$x \cdot y = \overline{\bar{x} + \bar{y}}$$

con lo que $\{+, ^-\}$ también es funcionalmente completo.

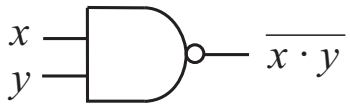
El conjunto $\{+, \cdot\}$ no es funcionalmente completo puesto que es imposible expresar la función booleana $F(x) = \bar{x}$ utilizando estos dos operadores.

Los conjuntos anteriores se pueden reducir a conjuntos de un solo elemento. Consideremos el operador $\uparrow$ o **NAND** y el operador $\downarrow$ o **NOR** definidos, respectivamente, por las tablas

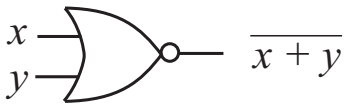
x	y	x ↑ y
0	0	1
0	1	1
1	0	1
1	1	0

x	y	x ↓ y
0	0	1
0	1	0
1	0	0
1	1	0

y representados en los circuitos combinacionales por



Puerta NAND



Puerta NOR

Los conjuntos $\{\uparrow\}$ y $\{\downarrow\}$ son funcionalmente completos puesto que

$$\bar{x} = x \uparrow x, \qquad x \cdot y = (x \uparrow y) \uparrow (x \uparrow y), \qquad x + y = (x \uparrow x) \uparrow (y \uparrow y).$$