Tema 1

Lógica y Álgebras de Boole

Recuerda que estas notas son, únicamente, parte del material de trabajo de los profesores de esta asignatura. Los contenidos de este tema se pueden ver en los siguientes libros:

- K.H. Rosen, *Matemática Discreta y sus aplicaciones* (5 ed.): capítulo 1 (secciones 1.1 a 1.5), capítulo 3 (secciones 3.1, 3.2 y 3.3) y capítulo 10.
- R.P. Grimaldi, *Matemática Discreta y combinatoria* (3 ed.): capítulo 2, capítulo 4 (sección 4.1) y capítulo 15.
- E. Paniagua Arís, *Lógica Computacional*: capítulos 1, 2, 3 y 4.

Cuando se plantea qué matemáticas va a necesitar un informático, la primera respuesta es: lógica y matemática discreta. Ese tipo de matemática está justificado pues la computación básicamente trata cantidades que no varían de forma continua, que son discretas. Y fuertes son las razones por las que un informático necesita estudiar lógica: en ella se encuentran las raíces de la ciencia de la computación (los trabajos de Church y Turing estaban motivados por el problema de decisión para la lógica de primer orden) y, de forma recíproca, la informática ha generado una explosión de interés en lógica, desde la búsqueda de métodos de automatización del razonamiento hasta la verificación de la corrección de un software. Eso es el equivalente al test de prueba de un producto de ingeniería; pero, dado que el software es un producto matemático, el test va más allá: es preciso demostrar que no tiene errores y que realiza lo especificado.

La lógica trata de la formalización del lenguaje y los métodos correctos de razonamiento, reglas y técnicas para determinar si un argumento dado es válido o no. En su forma tradicional, la lógica tiene una historia mucho más larga que la computación, remontándose al siglo IV a.C. con la codificación del silogismo, por Aristóteles (384–322 a.C.). Con ello se determinaban las formas del razonamiento válidas, frente a las no válidas de los sofistas. Ello, junto con el objetivo de entender la estructura de los lenguajes naturales, sigue siendo objeto de la lógica y es relevante en inteligencia artificial (para representación del conocimiento), en la construcción de programas informáticos y en la verificación de la corrección de ellos.

En el siglo XVII, el filósofo alemán Leibniz (1646–1716) sugirió incorporar símbolos matemáticos a la lógica con el fin de mecanizar el proceso de razonamiento deductivo. Durante el XIX estas ideas se materializan en los trabajos de De Morgan (1806–1871), Frege (1848–1925) y, en especial, de G. Boole (1815–1864) en su libro "Las leyes del pensamiento" (1854), donde desarrolla un sistema de reglas que le permitan expresar, manipular y simplificar problemas lógicos mediante cálculo simbólico. Boole también definió una estructura algebraica importante: las álgebras de Boole. En 1938 Shannon (1916–2001) mostró cómo utilizar las reglas de la lógica para diseñar circuitos, y que las propiedades fundamentales de dichos circuitos se pueden representar usando álgebras de Boole.

1.1 Lógica Proposicional

1.1.1 Proposiciones

Los objetos básicos de la lógica son las proposiciones, que se definen de la forma siguiente.

Definición 1. *Una* ***proposición*** *es una oración declarativa que es verdadera o falsa, pero no ambas cosas a la vez. Le asignaremos uno, y solo uno, de los valores de verdad: verdadero* (1) *o falso* (0)*.*

Ejemplo 1. *Los siguientes enunciados son proposiciones:*

- *Los triángulos tienen cuatro vértices.* (0)
- *Ronaldo Del Carmen es uno de los codirectores de* "Inside Out" (1)
- $2+2=4$. (1)
- $2\times 4=7$. (0)

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

- *Ojalá no llueva.*
- *¿A qué hora salimos de clase?*
- *¡Prohibido!*
- $x+3=4$.

  Las tres primeras frases no son proposiciones porque no son declarativas; la primera es desiderativa, la segunda interrogativa y la tercera es imperativa. La cuarta frase tampoco es es proposición porque no es ni verdadera ni falsa, pues su valor de verdad depende del valor que le asignemos a x*. Este tipo de frases se estudiarán en la sección 1.2.*

1.1.2 Operadores lógicos. Sintaxis

Las proposiciones **primitivas** (**simples** o **atómicas**) son aquellas proposiciones que no se pueden descomponer en otras más simples (como las que aparecen en el ejemplo 1); suelen designarse con letras minúsculas p, q, r, etc. Para obtener nuevas proposiciones, llamadas **compuestas**, se utilizan los **conectivos** u **operadores lógicos** siguientes: $\neg$, $\wedge$, $\vee$, $\rightarrow$ y $\leftrightarrow$.

- **Negación** ($\neg$) Si p es una proposición, la negación de p se denota por $\neg p$ y se lee:

  "no p" "no ocurre p" "no es cierto p"

- **Conjunción** ($\wedge$) Si p y q son proposiciones, la conjunción de p y q se denota por $p \wedge q$ y se lee:

  "p y q" "p, no obstante q" "p, pero q"
  "p, sin embargo q" "p, q" "p, aunque q"

- **Disyunción** ($\vee$) Si p y q son proposiciones, la disyunción de p y q se denota por $p \vee q$ y se lee:

  "p o q (o ambos)" "al menos p o q" "como mínimo p o q"

- **Condicional** ($\rightarrow$) Si p y q son proposiciones, el condicional $p \rightarrow q$ se lee:

  "Si p, entonces q"
  "Si p, q" "q si p"
  "p solo si q" "q siempre que p"
  "cuando p, entonces q" "q cuando p"
  "p es suficiente para q" "q es necesario para p"

- **Bicondicional** ($\leftrightarrow$) Si p y q son proposiciones, el bicondicional $p \leftrightarrow q$ se lee:

  "p si, y solo si, q" "p es suficiente y necesario para q"

Definición 2. *Al conectar entre sí las proposiciones mediante los operadores lógicos, se obtienen* ***expresiones*** *o* ***fórmulas bien formadas*** *(f.b.f.). Las reglas para construir una expresión bien formada son:*

1. *Las proposiciones primitivas o atómicas son f.b.f.*
2. *Si $\mathcal{P}$ es una f.b.f., $\neg\mathcal{P}$ es una f.b.f.*
3. *Si $\mathcal{P}$ y $\mathcal{Q}$ son f.b.f., entonces $\mathcal{P} \wedge \mathcal{Q}$, $\mathcal{P} \vee \mathcal{Q}$, $\mathcal{P} \rightarrow \mathcal{Q}$ y $\mathcal{P} \leftrightarrow \mathcal{Q}$ son f.b.f.*
4. *No hay más reglas.*

Como símbolos auxiliares se utilizan paréntesis o corchetes, que sirven para agrupar y evitar ambigüedades. 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.

Ejemplo 3. *La expresión $p \rightarrow (\neg q)$ es una f.b.f., mientras que $p \rightarrow (\wedge q)$ no lo es.*

Se establece una jerarquía de prioridades entre las conectivas, que permitirá suprimir paréntesis. 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. Así, $p \to (q \wedge r)$ puede escribirse como $p \to q \wedge r$, y $p \to q \wedge \neg r$ es $p \to (q \wedge (\neg r))$. La prioridad dentro del mismo nivel se indica con paréntesis.

Ejemplo 4. *En la proposición compuesta* $\neg p \wedge (r \to t)$ *no podemos eliminar paréntesis ya que, si los suprimimos, quedaría la secuencia* $\neg p \wedge r \to t$ *que representa a* $(\neg p \wedge r) \to t$. *En* $\neg p \wedge (q \vee r)$ *tampoco se pueden suprimir los paréntesis, ya que la conjunción y la disyunción están en el mismo nivel de evaluación.*

Ejemplo 5. *Para formalizar la frase* "Si Juan va al cine pero no compra palomitas, ahorra 2 euros"*, consideramos las proposiciones atómicas:*
p: Juan va al cine*,* *q:* Juan compra palomitas *y* *r:* Juan ahorra 2 euros*.*
El enunciado dado es $p \wedge \neg q \to r$.

1.1.3 Operadores lógicos. Semántica

Al igual que a las proposiciones primitivas, a las proposiciones compuestas se les asigna un valor de verdad. Los valores de verdad de estas dependen de los valores de verdad de las proposiciones primitivas que las componen.

Dadas las proposiciones p y q,

- La **negación** $\neg p$ es verdadera únicamente cuando p es falsa.
- La **conjunción** $p \wedge q$ es verdadera cuando tanto p como q son verdaderas, y falsa en cualquier otro caso.
- La **disyunción** $p \vee q$ es falsa cuando tanto p como q son falsas, y verdadera en cualquier otro caso.
- El **condicional** $p \to q$ es falso cuando p es verdadera y q es falsa, y verdadero en cualquier otro caso.
- El **bicondicional** $p \leftrightarrow q$ es verdadero cuando p y q tienen los mismos valores de verdad, y falso en los otros casos.

A continuación se muestran en unas tablas (llamadas tablas de verdad) los valores de las proposiciones anteriores:

p	$\neg p$
0	1
1	0

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

Comprender el significado (la semántica) del condicional es muy importante, para ello veamos el siguiente ejemplo.

Ejemplo 6. *El padre de Juan le dice a este: "Si apruebas* Matemática Discreta*, te compro un ordenador". En este caso, lo simbolizaríamos como* $p \to q$*, donde*

p : *"Apruebas* Matemática Discreta*"* $\quad\quad$ q : *"Te compro un ordenador"*

Se pueden distinguir cuatro casos:

1. p y q son ambas verdaderas, es decir Juan aprueba *Matemática Discreta* y su padre le compra el ordenador. Se cumple la promesa y el valor de verdad del condicional es 1.

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

3. p y q son ambas falsas (Juan suspende *Matemática Discreta* 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 $p \to q$ es 1.

4. p es falsa (Juan suspende) y q es verdadera (su padre le compra el ordenador). Aunque el padre le compra el ordenador a Juan, tampoco, en este caso, se rompe la promesa y el valor de verdad del condicional es 1.

Es decir, el padre rompe su promesa **únicamente** cuando Juan aprueba y, sin embargo, no le compra el ordenador. Es decir, el valor del condicional $p \to q$ es 0 únicamente cuando p es verdadera y q es falsa.

Ejemplo 7. *Fijémonos en la diferencia entre el argumento anterior y el siguiente. El padre de Juan le dice a este: "Te compro un ordenador si, y solamente si, apruebas* Matemática Discreta*". En este caso, lo simbolizaríamos como $q \leftrightarrow p$. Veamos aquí los cuatro casos posibles:*

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

2. *p es verdadera* (*Juan aprueba* Matemática Discreta) *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.

3. *p y q son ambas falsas* (*Juan suspende* Matemática Discreta *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.

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

Definición 3. *Una* ***tabla de verdad*** *para una proposición compuesta es un método que proporciona los valores de verdad de la proposición dada, a partir de los valores de verdad de las proposiciones atómicas que la conforman.*

Para construirla se determinan los valores de verdad de sus subproposiciones componentes, desde las más sencillas hasta las más complejas.

Ejemplo 8. *La tabla de verdad de* $(p \vee q) \wedge (p \to \neg q)$ *es:*

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**	$\leadsto$ contraejemplo
0	1	1	0	1	**1**	$\leadsto$ modelo
1	0	1	1	1	**1**	$\leadsto$ modelo
1	1	1	0	0	**0**	$\leadsto$ contraejemplo

Una **interpretación** de una fórmula $\mathcal{P}$ es una asignación de verdad de las proposiciones atómicas que aparecen en $\mathcal{P}$. Cada fila de la tabla de verdad de $\mathcal{P}$ se corresponde con una interpretación de $\mathcal{P}$. Si en $\mathcal{P}$ intervienen n proposiciones primitivas, hay 2^n posibles interpretaciones de $\mathcal{P}$ (lo cual muestra que es un procedimiento de complejidad exponencial). También se puede definir una interpretación de $\mathcal{P}$ como una aplicación del conjunto de átomos que intervienen en $\mathcal{P}$ al conjunto $\{0, 1\}$.

Si para una interpretación de $\mathcal{P}$, el valor de $\mathcal{P}$ es 1 se dice que esa interpretación es un **modelo**; si, por el contrario, el valor de $\mathcal{P}$ es 0 se dice que esa interpretación es un **contraejemplo**. En el caso anterior, la primera y la última fila se corresponden con contraejemplos y las otras dos son modelos. Es decir, $\{\neg p, q\}$ y $\{p, \neg q\}$ son modelos de $(p \vee q) \wedge (p \to \neg q)$, mientras que $\{\neg p, \neg q\}$ y $\{p, q\}$ son contraejemplos.

Se puede obtener la tabla de verdad de una proposición $\mathcal{P}$ utilizando otra forma de disponer sus subproposiciones. Para el ejemplo anterior, 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*	*5*	*4*	*3*

El valor de verdad de cada paso queda determinado por los valores de verdad de los pasos anteriores, como vemos en la siguiente tabla donde se calculan los valores de verdad de la proposición $(p \wedge q) \vee \neg p \to \neg q$:

	p	q	$p \wedge q$	$\neg p$	$(p \wedge q) \vee \neg p$	$(p \wedge q) \vee \neg p \to \neg q$	$\neg q$
	0	0	0	1	1	**1**	1
	0	1	0	1	1	**0**	0
	1	0	0	0	0	**1**	1
	1	1	1	0	1	**0**	0
Paso	*1*		*2*	*3*	*4*	*6*	*5*

Se dice que $\{\neg p, \neg q\}$ y $\{p, \neg q\}$ son modelos de $(p \wedge q) \vee \neg p \to \neg q$, mientras que $\{\neg p, q\}$ y $\{p, q\}$ son contraejemplos.

Definición 4. *Una fórmula que es siempre verdadera se denomina* ***tautología****, y se denotará* $\top$. *Una fórmula que es siempre falsa se denomina* ***contradicción****, y se denotará* $\perp$. *Finalmente, una fórmula que no es ni una tautología ni una contradicción se llama* ***contingencia****.*

Ejemplo 9.

- $p \vee \neg p$ *es una tautología.*

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

- $p \wedge \neg p$ *es una contradicción.*

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

Ejemplo 10.

- $p \vee \neg(p \wedge q)$ *es una tautología.*
- $(p \wedge q) \wedge \neg(p \vee \neg q)$ *es una contradicción.*

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

Definición 5. *Un conjunto de proposiciones* $\{\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_n\}$ *se dice* ***consistente*** *(o* ***satisfacible****) si la conjunción de todas ellas* $\mathcal{P}_1 \wedge \mathcal{P}_2 \wedge \cdots \wedge \mathcal{P}_n$ *admite al menos un modelo*[1]. *En caso contrario, es decir, si* $\mathcal{P}_1 \wedge \mathcal{P}_2 \wedge \cdots \wedge \mathcal{P}_n$ *es una contradicción, el conjunto* $\{\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_n\}$ *se dice inconsistente.*

Ejemplo 11.

- *El conjunto* $\{p \to q,\ p \wedge q\}$ *es consistente ya que, cuando p y q son verdaderas, las fórmulas* $p \to q$ *y* $p \wedge q$ *son verdaderas, es decir,* $(p \to q) \wedge (p \wedge q)$ *admite un modelo (cuando p y q son verdaderas,* $\{p, q\}$*).*

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

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

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

En la sección 1.1.5 veremos otro método, el de las tablas semánticas, para estudiar si un conjunto de fórmulas es consistente o no.

[1] Esto indica que hay una interpretación que es modelo de todas las proposiciones $\mathcal{P}_1$, $\mathcal{P}_2, \dots, \mathcal{P}_n$.

1.1.4 Implicaciones y Equivalencias lógicas

Consideremos las dos afirmaciones siguientes: “Marta es estudiante y juega al tenis” y “Marta juega al tenis y es estudiante”. Obviamente, ambas tienen siempre los mismos valores de verdad. Para hacer más precisa esta idea, sea p la proposición “Marta es estudiante” y sea q la proposición “Marta juega al tenis”, entonces la primera de las dos afirmaciones se traduce en $p \wedge q$, mientras que la segunda se traduce en $q \wedge p$. 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 6. *Dos proposiciones compuestas $\mathcal{P}$ y $\mathcal{Q}$ se dicen **lógicamente equivalentes** si ambas tienen los mismos valores de verdad para cada interpretación de los valores de verdad de sus proposiciones componentes (es decir, la fórmula $\mathcal{P} \leftrightarrow \mathcal{Q}$ es una tautología). Ea situación se representa por*

$$\mathcal{P} \equiv \mathcal{Q}$$

*También puede denotarse por $\mathcal{P} \iff \mathcal{Q}$. Es importante diferenciar “$\iff$” de “$\leftrightarrow$”: $\mathcal{P} \iff \mathcal{Q}$ indica que la fórmula $\mathcal{P} \leftrightarrow \mathcal{Q}$ es una tautología, mientras que cuando se escribe $\mathcal{P} \leftrightarrow \mathcal{Q}$ simplemente se está dando una proposición compuesta. Nótese que $\mathcal{P} \iff \mathcal{Q}$ **no** es una f.b.f.*

Ejemplo 12. *Las proposiciones $\neg(p \wedge q)$ y $\neg p \vee \neg q$ son lógicamente equivalentes:*

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

Ejemplo 13. *La proposición compuesta $p \wedge q \leftrightarrow q$ no es una tautología, por lo que las proposiciones $p \wedge q$ y q no son lógicamente equivalentes.*

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

Para cualesquiera fórmulas $\mathcal{P}$, $\mathcal{Q}$, $\mathcal{R}$, se verifica que:

- $\mathcal{P} \equiv \mathcal{P}$
- si $\mathcal{P} \equiv \mathcal{Q}$, entonces $\mathcal{Q} \equiv \mathcal{P}$
- si $\mathcal{P} \equiv \mathcal{Q}$ y $\mathcal{Q} \equiv \mathcal{R}$, entonces $\mathcal{P} \equiv \mathcal{R}$

Al final de esta sección, la tabla 1.1 recoge algunas equivalencias lógicas.

Las equivalencias lógicas de la tabla, así como cualquier otra que se haya probado, permiten construir otras equivalencias lógicas. Ello se debe a que una proposición en una fórmula se puede

substituir por otra que sea lógicamente equivalente sin alterar el valor de la fórmula:
Sea $\mathcal{P}$ una proposición compuesta, de la cual forma parte una proposición Q, y sea Q^* una proposición lógicamente equivalente a Q. Si se substituye alguna ocurrencia de Q en $\mathcal{P}$ por Q^*, se obtiene una proposición $\mathcal{P}^*$ lógicamente equivalente a $\mathcal{P}$.

Ejemplo 14. *Sea $\mathcal{P}$ la fórmula $\neg(p \vee q) \to r$. Puesto que la subfórmula $\neg(p \vee q)$ es lógicamente equivalente a $\neg p \wedge \neg q$, se tiene que $\mathcal{P} \equiv \mathcal{P}^*$, siendo $\mathcal{P}^*$ la fórmula $\neg p \wedge \neg q \to r$.*

Ejemplo 15. *Estas equivalencias permiten simplificar la fórmula $(p \vee q) \wedge \neg(\neg p \wedge q)$ de la forma siguiente:*

$$\begin{array}{rcll}
(p \vee q) \wedge \neg(\neg p \wedge q) & \equiv & (p \vee q) \wedge (p \vee \neg q) & \textit{Leyes de De Morgan y Doble negación} \\
& \equiv & p \vee (q \wedge \neg q) & \textit{Ley distributiva} \\
& \equiv & p \vee \perp & \textit{Contradicción} \\
& \equiv & p & \textit{Ley Identidad}
\end{array}$$

Definición 7. *Dadas dos proposiciones $\mathcal{P}$ y $\mathcal{Q}$, se dice que $\mathcal{P}$* ***implica lógicamente*** *$\mathcal{Q}$, o que de $\mathcal{P}$ se deduce $\mathcal{Q}$, cuando la proposición $\mathcal{P} \to \mathcal{Q}$ es una tautología. Es decir, cuando $\mathcal{P}$ es verdadera, también $\mathcal{Q}$ es verdadera y que cuando $\mathcal{Q}$ es falsa, $\mathcal{P}$ es falsa. En este caso, se escribe*

$$\mathcal{P} \models \mathcal{Q}$$

También se denota por $\mathcal{P} \Longrightarrow \mathcal{Q}$. De nuevo, es importante diferenciar "$\Longrightarrow$" y "$\to$" : mientras $\mathcal{P} \Longrightarrow \mathcal{Q}$ quiere decir que la fórmula $\mathcal{P} \to \mathcal{Q}$ es una tautología, cuando se escribe $\mathcal{P} \to \mathcal{Q}$ simplemente se está dando una proposición compuesta. Nótese que $\mathcal{P} \Longrightarrow \mathcal{Q}$ ***no*** *es una f.b.f.*

Ejemplo 16. *Para demostrar que $(p \to q) \wedge p \models q$ es suficiente probar que la fórmula $[(p \to q) \wedge p] \to q$ es una tautología. Si se construye su tabla de verdad, se observa que todos sus valores son 1:*

	p	q	$p \to q$	$(p \to q) \wedge p$	$[(p \to q) \wedge p] \to q$	q
	0	0	1	0	**1**	0
	0	1	1	0	**1**	1
	1	0	0	0	**1**	0
	1	1	1	1	**1**	1
Paso	*1*		*2*	*3*	*4*	*1*

Al final de esta sección, la tabla 1.2 recoge algunas implicaciones lógicas.

Principales equivalencias lógicas	
Leyes Conmutativas	$p \vee q \equiv q \vee p$ $p \wedge q \equiv q \wedge p$
Leyes Asociativas	$(p \vee q) \vee r \equiv p \vee (q \vee r)$ $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$
Leyes Distributivas	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$
Ley de la Doble Negación	$\neg\neg p \equiv p$
Leyes de De Morgan	$\neg(p \vee q) \equiv (\neg p \wedge \neg q)$ $\neg(p \wedge q) \equiv (\neg p \vee \neg q)$
Leyes de Dominación	$p \vee \top \equiv \top$ $p \wedge \bot \equiv \bot$
Leyes de Identidad	$p \wedge \top \equiv p$ $p \vee \bot \equiv p$
Leyes de la Negación	$p \vee \neg p \equiv \top$ $p \wedge \neg p \equiv \bot$
Ley de la Contraposición	$(p \to q) \equiv (\neg q \to \neg p)$
Leyes de la Implicación	$(p \to q) \equiv (\neg p \vee q)$ $\neg(p \to q) \equiv (p \wedge \neg q)$
Leyes de la Equivalencia	$(p \leftrightarrow q) \equiv [(p \to q) \wedge (q \to p)]$ $(p \leftrightarrow q) \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$
Leyes Idempotentes	$p \equiv (p \wedge p)$ $p \equiv (p \vee p)$
Leyes de Absorción	$p \wedge (p \vee q) \equiv p$ $p \vee (p \wedge q) \equiv p$
Ley de la Reducción al absurdo	$(p \to q) \equiv [(p \wedge \neg q) \to \bot]$

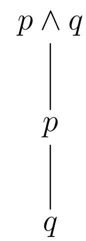
Table 1.1: Tabla de equivalencias

Principales implicaciones lógicas	
Modus Ponens	$[(p \to q) \wedge p] \models q$
Modus Tollens	$[(p \to q) \wedge \neg q] \models \neg p$
Silogismo	$[(p \to q) \wedge (q \to r)] \models (p \to r)$
Leyes de Simplificación	$(p \wedge q) \models p$ $(p \wedge q) \models q$
Leyes de Adición	$p \models (p \vee q)$ $q \models (p \vee q)$
Silogismo Disyuntivo	$((p \vee q) \wedge \neg p) \models q$ $((p \vee q) \wedge \neg q) \models p$
Ley de Casos	$[(p \to q) \wedge (\neg p \to q)] \models q$
Ley de Inconsistencia	$[p \wedge \neg p] \models q$

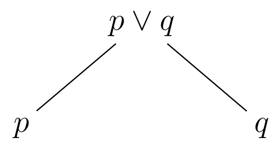
Table 1.2: Tabla de implicaciones

1.1.5 Tablas semánticas

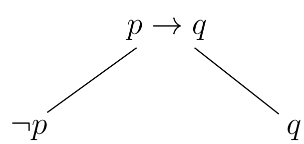
El método de las **tablas semánticas**[2] o **árbol semántico** fue descubierto a mediados del siglo pasado por **Beth** y **Hintikka**, independientemente uno del otro, y permite demostrar si un conjunto de fórmulas es consistente o no. Para ello, se construye un árbol donde los nodos (finitos) son las proposiciones, el conectivo $\wedge$ se representa por un arista vertical,



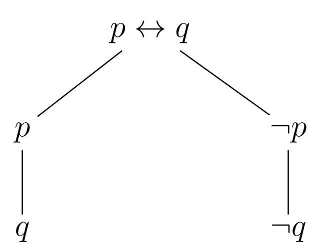
y el conectivo $\vee$ por un par de aristas en la forma



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



ya que $p \to q \equiv \neg p \vee q$. Por otro lado, como $p \leftrightarrow q \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$, el bicondicional se representará por



Para comprobar si un conjunto de proposiciones $\{\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_n\}$ es consistente, se construye la tabla semántica de $\mathcal{P}_1 \wedge \mathcal{P}_2 \wedge \cdots \wedge \mathcal{P}_n$ y se van descomponiendo, una a una, todas las proposiciones compuestas, de acuerdo con las reglas anteriores, marcando las proposiciones ya utilizadas. 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 (vértice) final.

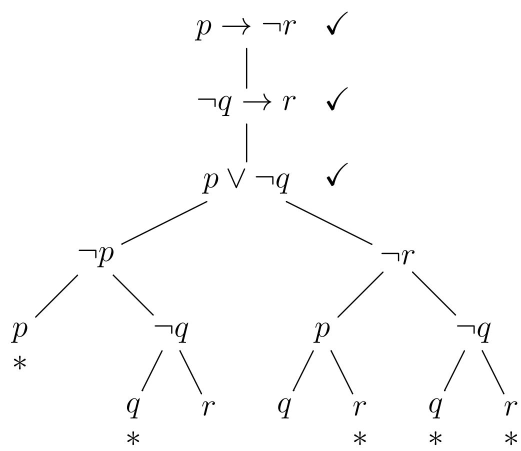
Si al final del proceso *todos* los caminos se cierran, la fómula $\mathcal{P}_1 \wedge \mathcal{P}_2 \wedge \cdots \wedge \mathcal{P}_n$ es una contradicción (el conjunto $\{\mathcal{P}_1,\ \mathcal{P}_2, \dots,\ \mathcal{P}_n\}$ es inconsistente); en caso contrario, cada camino abierto representa un modelo de $\mathcal{P}_1 \wedge \mathcal{P}_2 \wedge \cdots \wedge \mathcal{P}_n$ (es decir, $\{\mathcal{P}_1,\ \mathcal{P}_2, \dots,\ \mathcal{P}_n\}$ es consistente).

[2] O *tableaux, del inglés "cuadro escénico".*

Ejemplo 17. *Para comprobar si el conjunto* $\{p \to \neg r,\ \neg q \to r,\ p \vee \neg q\}$ *es o no consistente, se construye el árbol semántico de* $(p \to \neg r)\ \wedge\ (\neg q \to r)\ \wedge\ (p \vee \neg q)$

$$\begin{array}{c} p \to \neg r \\ | \\ \neg q \to r \\ | \\ p \vee \neg q \end{array}$$

Se descompone cada una de las fórmulas compuestas, sin importar el orden, pero una vez descompuesta una fórmula se marca ($\checkmark$), *y se van señalando las ramas que se cierran* ($*$):



aplicando $p \to \neg r\ \equiv \neg p \vee \neg r$

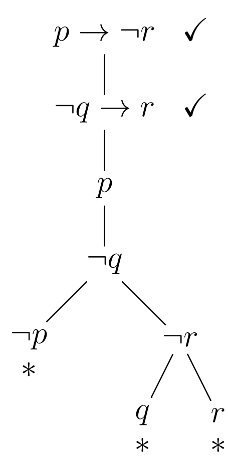
aplicando $\neg q \to r\ \equiv\ q \vee r$

Las ramas abiertas, $\{\neg p, \neg q, r\}$ *y* $\{p, q, \neg r\}$, *proporcionan los modelos de*

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

Es decir, cuando p *y* q *son falsas y* r *verdadera, o bien cuando* p *y* q *son verdaderas y* r *falsa, la proposición* $(p \to \neg r) \wedge (\neg q \to r) \wedge (p \vee \neg q)$ *es verdadera o, lo que es lo mismo, todas las fórmulas del conjunto* $\{p \to \neg r,\ \neg q \to r,\ p \vee \neg q\}$ *tienen valor de verdad* 1.

Ejemplo 18. *Consideremos ahora el conjunto de fórmulas* $\{p \to \neg r,\ \neg q \to r,\ p \wedge \neg q\}$. *El árbol semántico correspondiente a* $(p \to \neg r)\ \wedge\ (\neg q \to r)\ \wedge\ (p \wedge \neg q)$ *es:*



aplicando $p \to \neg r\ \equiv \neg p \vee \neg r$

aplicando $\neg q \to r\ \equiv\ q \vee r$

Como todos los caminos se cierran, el conjunto de fórmulas dado es inconsistente, es decir, $(p \to \neg r)\ \wedge\ (\neg q \to r)\ \wedge\ (p \wedge \neg q)$ *no tiene ningún modelo (es una contradicción).*

1.1.6 Argumentos y métodos de demostración

Definición 8. *Dado un conjunto de proposiciones* $\{H_1, H_2, \ldots, H_n\}$*, se dice que una proposición* C *es consecuencia de ellas si* $H_1 \wedge H_2 \wedge \ldots \wedge H_n \models C$ *(también se dice que este es un argumento válido, y se puede representar por* $\{H_1, H_2, \ldots, H_n\} \models C$*).*

Las proposiciones $H_1, H_2, \ldots, H_n$ *se llaman* ***hipótesis*** *o* ***premisas*** *y* C *se llama* ***conclusión****. El argumento es válido si, siempre que todas las hipótesis son verdaderas, también lo es la conclusión.*

Se puede realizar una **demostración directa** de la validez de un argumento mediante tablas de verdad.

Ejemplo 19. *Para demostrar la implicación lógica "Modus Tollens", es decir,*

$$\{p \to q,\ \neg q\} \models \neg p$$

se prueba que $(p \to q) \wedge \neg q \to \neg p$ *es una tautología:*

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

Ejemplo 20. *Si queremos demostrar que*

$$\{p \to r,\ q \to r\} \models p \vee q \to r$$

llamamos

$$H_1 :\ p \to r \qquad H_2 :\ q \to r \qquad C :\ p \vee q \to r$$

y construimos la tabla de verdad de $H_1 \wedge H_2 \to C$ *para comprobar que es una tautología:*

p	q	r	$H_1 : p \to r$	$H_2 : q \to r$	$H_1 \wedge H_2$	$H_1 \wedge H_2 \to C$	$C :\ p \vee q \to r$
0	0	0	1	1	1	**1**	1
0	0	1	1	1	1	**1**	1
0	1	0	1	0	0	**1**	0
0	1	1	1	1	1	**1**	1
1	0	0	0	1	0	**1**	0
1	0	1	1	1	1	**1**	1
1	1	0	0	0	0	**1**	0
1	1	1	1	1	1	**1**	1

Teniendo en cuenta que el condicional $H_1 \wedge H_2 \to C$ es falso únicamente cuando la conjunción $H_1 \wedge H_2$ es verdadera (esto es, si las hipótesis, H_1 y H_2, son ambas verdaderas) y la conclusión C es falsa, para estudiar si

$$H_1 \wedge H_2 \models C$$

es un argumento válido, basta calcular únicamente los valores de verdad de $H_1 \wedge H_2$ en los casos en los que C tiene valor de verdad 0. Esto es, en la tabla del ejemplo anterior los cálculos se reducen a las filas tercera, quinta y séptima. Incluso en esas filas, si una de las hipótesis es falsa ya no será necesario hallar el valor de verdad de las demás (pues la conjunción de las hipótesis ya es falsa):

p	q	r	$H_1 : p \to r$	$H_2 : q \to r$	$H_1 \wedge H_2$	$H_1 \wedge H_2 \to C$	C
0	0	0	-	-	-	**1**	1
0	0	1	-	-	-	**1**	1
0	1	0	1	0	0	**1**	**0**
0	1	1	-	-	-	**1**	1
1	0	0	0	-	0	**1**	**0**
1	0	1	-	-	-	**1**	1
1	1	0	0	-	0	**1**	**0**
1	1	1	-	-	-	**1**	1

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

1. Es una de las hipótesis.
2. Es una tautología conocida (equivalencia lógica).
3. Es lógicamente equivalente a una proposición anterior.
4. Se deriva de alguna de las proposiciones anteriores por reglas de sustitución.
5. 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 21. *Utilizando reglas de inferencia, probaremos cómo de las hipótesis* "Si no llueve o no hay niebla, entonces se celebrará la competición de barcos y se hará una demostración de los salvavidas", "Si se celebra la competición de barcos, se entregará un trofeo" y "El trofeo no se entregó" *se deduce que* "Llovió".

En primer lugar, consideramos las proposiciones primitivas siguientes:

p: "Llueve"; n: "Hay niebla"; b: "Se celebra competición de barcos";
s: "Se hace demostración de salvavidas"; t: "Se entrega un trofeo".

Las hipótesis y la conclusión del argumento dado son:

$H_1 :\ \neg p \vee \neg n \to b \wedge s$
$H_2 :\ b \to t$
$H_3 :\ \neg t$
$C :\ p$

Teniendo presente que el objetivo es deducir p, aplicamos reglas de inferencia de la forma siguiente:

(1)	$\neg p \vee \neg n \to b \wedge s$	*es una hipótesis*
(2)	$b \to t$	*es una hipótesis*
(3)	$\neg t$	*es una hipótesis*
(4)	$\neg b$	*aplicando Modus Tollens a (2) y (3)*
(5)	$\neg b \vee \neg s$	*aplicando ley de adición a (4)*
(6)	$\neg(b \wedge s)$	*aplicando ley de De Morgan a (5)*
(7)	$\neg(\neg p \vee \neg n)$	*aplicando Modus Tollens a (1) y (6)*
(8)	$p \wedge n$	*aplicando las leyes de De Morgan y la doble negación a (7)*
(9)	p	*aplicando la ley de simplificación a (8)*

Un tipo de **demostración indirecta** es la **contraposición**. Este tipo de demostración está basada en la equivalencia $(p \to q) \equiv (\neg q \to \neg p)$. En nuestro caso, demostrar

$$H_1 \wedge H_2 \wedge \cdots \wedge H_n \models C$$

equivale a demostrar que

$$\neg C \models \neg(H_1 \wedge H_2 \wedge \cdots \wedge H_n), \text{ esto es, } \quad \neg C \models \neg H_1 \vee \neg H_2 \vee \cdots \vee \neg H_n$$

Ejemplo 22. *Si a y b son números naturales y $a + b \geq 25$, entonces $a \geq 13$ o $b \geq 13$. Consideramos las proposiciones primitivas:*

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

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

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

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

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

$$H_1 \wedge H_2 \wedge \cdots \wedge H_n \wedge \neg C \models \perp$$

Puesto que demostrar que el argumento $H_1 \wedge H_2 \wedge \ldots \wedge H_n \models C$ es válido significa probar que

$$H_1 \wedge H_2 \wedge \cdots \wedge H_n \to C \quad \text{es una tautología}$$

es decir, que su negación

$$H_1 \wedge H_2 \wedge \cdots \wedge H_n \wedge \neg C \quad \text{es una contradicción}^3$$

o que $\{H_1, H_2, \ldots, H_n, \neg C\}$ es un conjunto inconsistente.

[3]Téngase en cuenta que, al aplicar la segunda ley de la implicación, se obtiene que

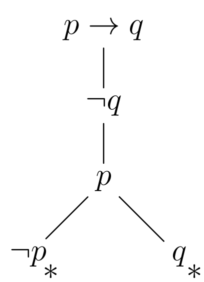
$$\neg(H_1 \wedge H_2 \wedge \cdots \wedge H_n \to C) \equiv H_1 \wedge H_2 \wedge \cdots \wedge H_n \wedge \neg C.$$

$$\boxed{\begin{array}{c}\text{El argumento } \{H_1,\ H_2,\ldots,\ H_n\} \models C \text{ es válido si, y solo si,}\\ \text{el conjunto } \{H_1,\ H_2,\ldots,\ H_n,\ \neg C\} \text{ es inconsistente.}\end{array}}$$

Este método de demostración por contradicción o reducción al absurdo nos permite utilizar las tablas semánticas para comprobar si un argumento es o no válido. Si queremos demostrar, o refutar, un argumento $H_1 \wedge H_2 \wedge \cdots \wedge H_n \models C$ calculamos la tabla semántica de $H_1 \wedge H_2 \wedge \cdots \wedge H_n \wedge \neg C$. Si, al finalizar, todos los caminos se cierran, tenemos que $H_1 \wedge H_2 \wedge \cdots \wedge H_n \wedge \neg C$ es una contradicción, es decir, el argumento $H_1 \wedge H_2 \wedge \cdots \wedge H_n \models C$ es válido. Por el contrario, la existencia de una rama abierta en el árbol de $H_1 \wedge H_2 \wedge \cdots \wedge H_n \wedge \neg C$ nos da un modelo de $H_1 \wedge H_2 \wedge \cdots \wedge H_n \wedge \neg C$, es decir, un contraejemplo de $H_1 \wedge H_2 \wedge \cdots \wedge H_n \rightarrow C$ y nos indica que el argumento $H_1 \wedge H_2 \wedge \cdots \wedge H_n \models C$ no es válido.

Ejemplo 23.

1. *Demostrar el "modus tollens",* $((p \rightarrow q) \wedge \neg q) \models \neg p$*, equivale a probar que el conjunto* $\{p \rightarrow q, \neg q, \neg\neg p\}$ *es inconsistente, es decir, que todas las ramas de la tabla semántica de* $(p \rightarrow q) \wedge \neg q \wedge p$ *se cierran.*



2. *Si llueve o hace viento, Manuel no corta el césped. Siempre que no hay nubes en el cielo, no llueve. Hoy no hace viento y no hay nubes en cielo. Entonces, Manuel corta el césped.*

   Llamemos:

p: "Llueve"	*n: "Hay nubes en el cielo"*
q: "Hace viento"	*c: "Manuel corta el césped"*

   Se trata de ver si, del conjunto de hipótesis $\{(p \vee q) \rightarrow \neg c, \neg n \rightarrow \neg p, \neg q, \neg n\}$ *se puede deducir c, es decir, queremos comprobar si*

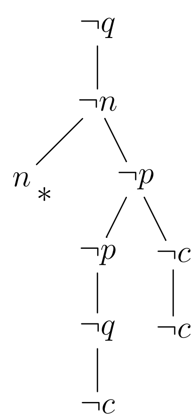
   $$\{(p \vee q) \rightarrow \neg c, \neg n \rightarrow \neg p, \neg q, \neg n\} \models c \quad \textit{es un argumento válido.}$$

   Haciendo una demostración por reducción al absurdo, tendremos que ver si

   $$(p \vee q \rightarrow \neg c) \wedge (\neg n \rightarrow \neg p) \wedge \neg q \wedge \neg n \wedge \neg c \ \textit{es una contradicción.}$$

   Para ello construimos el árbol semántico de la proposición

   $$(p \vee q \rightarrow \neg c) \wedge (\neg n \rightarrow \neg p) \wedge \neg q \wedge \neg n \wedge \neg c$$



Donde hemos utilizado que:

$$\neg n \to \neg p \;\equiv\; n \vee \neg p$$
$$p \vee q \to \neg c \;\equiv\; (\neg p \wedge \neg q) \vee \neg c$$

Como vemos, $\{\neg p, \neg q, \neg n, \neg c\}$ es un modelo de $H_1 \wedge H_2 \wedge H_3 \wedge H_4 \wedge \neg C$, donde

$$H_1 :\; p \vee q \to \neg c, \qquad H_2 :\; \neg n \to \neg p, \qquad H_3 :\; \neg q, \qquad H_4 :\; \neg n, \qquad C :\; c,$$

por lo que es un contraejemplo de su negación $\neg(H_1 \wedge H_2 \wedge H_3 \wedge H_4 \wedge \neg C)$, es decir, de $H_1 \wedge H_2 \wedge H_3 \wedge H_4 \longrightarrow C$. Se obtiene así un contraejemplo del argumento dado y se concluye que este no es válido.

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

1. Es menos costoso de aplicar.

2. Es una buena base para programar demostradores automáticos.

3. Puede extenderse a otras lógicas, para las cuales el método de las tablas de verdad deja de tener sentido.

4. En el caso de que el argumento no sea válido las tablas semánticas nos muestran explícitamente un contraejemplo.

1.2 Lógica de Predicados

Como hemos comentado al inicio del tema, un enunciado del tipo "$x + 2$ *es un número par*" no es una proposición porque no es ni verdadero ni falso si no se especifican los valores de la variable. Por ejemplo, si $x = 1$, entonces el enunciado es falso; mientras que si $x = 2$, el enunciado es verdadero.

Este sería un ejemplo de predicado (afirmación que expresa una propiedad de un objeto o una relación entre objetos) y su valor de verdad depende del valor que tome una variable (o varias variables) que recorre un cierto conjunto llamado **dominio** o **universo**. Por ejemplo, si consideramos el predicado $Q(x, y) :\; x + y = 5$ en el universo de los números enteros positivos, $Q(x, y)$ se convierte en una proposición verdadera cuando asignamos los valores $x = 1, y = 4$; $x = 2, y = 3$; $x = 3, y = 2$; o bien $x = 4, y = 1$ a las variables x e y; y es una proposición falsa en los demás casos.

De hecho, un predicado se transforma en una proposición sustituyendo todas las variables por los elementos del dominio o utilizando cuantificadores. A continuación veremos el *cuantificador universal* y el *cuantificador existencial*.

El **cuantificador universal** "$\forall$" se utiliza para construir proposiciones del tipo siguiente:

- $\forall x\ p(x)$

  Se lee "para todo x, $p(x)$", o bien, "para cada x, $p(x)$", o bien, "para cualquier x, $p(x)$".

  Este tipo de proposición es verdadera cuando $p(a)$ es verdadera para cualquier valor a del dominio U, y es falsa si $p(a)$ es falsa para algún valor de U.

  Por ejemplo: "Todos los alumnos de esta Facultad tienen más de 16 años", es una proposición verdadera. "Todos los alumnos de esta Facultad nacieron en Coruña" es una proposición falsa.

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

- $\exists x\ p(x)$

  Se lee "existe x que verifica $p(x)$".

  Esta proposición es verdadera cuando $p(a)$ es verdadera para, al menos, un valor a de U. Es falsa cuando, para todo valor a de U, la proposición $p(a)$ es falsa.

  Por ejemplo: "Existe un entero x que sumado con 1 nos da 0" es verdadera. "Existe un entero x que sumado con 1 nos da x" es una proposición falsa.

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

$$\neg\,[\forall x\ \ p(x)]\ \equiv\ \exists x\ \ \neg p(x) \qquad\qquad \neg\,[\exists x\ \ p(x)]\ \equiv\ \forall x\ \ \neg p(x)$$

Ejemplo 24. *Escribamos la negación de la siguiente proposición cuantificada:*

$$\forall x\,\exists y\ \ [p(x) \wedge \neg q(x) \rightarrow r(x, y)]$$

Aplicando las reglas anteriores, tenemos:

$$\begin{aligned}
&\neg\,[\forall x\ \exists y\quad p(x) \wedge \neg q(x) \rightarrow r(x, y)]\\
\equiv\ &\exists x\ \ \neg\,[\exists y\quad p(x) \wedge \neg q(x) \rightarrow r(x, y)]\\
\equiv\ &\exists x\ \ \forall y\quad \neg\,[p(x) \wedge \neg q(x) \rightarrow r(x, y)]\\
\equiv\ &\exists x\ \ \forall y\quad \neg\,[\neg(p(x) \wedge \neg q(x)) \vee r(x, y)]\\
\equiv\ &\exists x\ \ \forall y\quad [p(x) \wedge \neg q(x) \wedge \neg r(x, y)]
\end{aligned}$$

Ejemplo 25. *Consideremos los predicados $A(x)$: "*x es estudiante de informática*" y $B(x)$: "*x tiene un ordenador*" en el universo de los estudiantes de la universidad de A Coruña.*

La frase "todos los estudiantes de informática tienen ordenador" *se formaliza como:*

$$\forall x\ \ (A(x) \rightarrow B(x))$$

mientras que la frase "algún estudiante de informática tiene ordenador" *se formaliza como:*

$$\exists x\ \ (A(x) \wedge B(x))$$

1.2.1 Argumentos en lógica de predicados

Formalicemos el siguiente argumento: "Todo el que tiene los ojos azules no es moreno. Pepe tiene los ojos azules. Entonces Pepe no es moreno". Sobre el universo de los humanos construimos los siguientes predicados:

$$A(x)\text{: “}x\text{ tiene los ojos azules”} \qquad M(x)\text{: “}x\text{ es moreno”}$$

En este argumento, las premisas o hipótesis son $\forall x\ [A(x) \rightarrow \neg M(x)]$ y $A(Pepe)$; mientras que la conclusión es $\neg M(Pepe)$.

Para demostrar argumentos expresados en lógica de predicados, se utilizan las reglas siguientes:

1. **Especificación universal** (EU)

   Si la proposición $\forall x\ F(x)$ es verdad, entonces se puede deducir que la proposición $F(a)$ es verdad para cualquier elemento a del universo del discurso. Esta regla la podemos representar en las tablas semánticas de la forma

   $$\begin{array}{c} \forall\, x\ F(x) \\ | \\ F(a) \end{array}$$

   Donde $F(a)$ significa sustituir simultáneamente cada aparición de x en F por la constante a.

2. **Generalización universal**

   Si la proposición $F(a)$ es verdad para cualquier elemento a del universo del discurso, entonces se concluye que $\forall x\ F(x)$ es verdad.

3. **Especificación existencial** (EE)

   Si la proposición $\exists x\ F(x)$ es verdad, entonces existe un elemento a en el universo del discurso tal que $F(a)$ es verdad. Esta regla la podemos representar en las tablas semánticas de la forma:

   $$\begin{array}{c} \exists x\ F(x) \checkmark \\ | \\ F(a) \end{array}$$

   Con a en el universo U y **distinto de todos los nombres** pertenecientes a U que aparecen en la misma rama donde va a estar $F(a)$.

4. **Generalización existencial** Si $F(a)$ es verdad para algún elemento a del universo del discurso, entonces la proposición $\exists x\ F(x)$ es verdad.

Debe notarse que en la especificación universal **no se marca como utilizada** la fórmula, pues se puede volver a utilizar. La justificación es que si la sentencia $\forall x\ F(x)$ es parte de un conjunto consistente, añadir uno o más casos de F al conjunto no cambiaría ese carácter. Sin embargo, en la especificación existencial **sí se marca** la fórmula al ser utilizada (con el símbolo $\checkmark$) y tiene como restricción que el nombre a de la variable **debe ser nuevo en cada rama**. Ello garantiza que el nombre está escogido en forma completamente arbitraria del contexto, lo que concuerda con la semántica: se conoce que es verdad al menos en un caso, aunque no sepamos en cuál. Por lo tanto, **se debe aplicar la Especificación existencial antes de utilizar la Especificación**

universal. Es decir, al contrario que en el cálculo proposicional, donde no importa el orden de aparición de las reglas para probar que un conjunto es inconsistente, **en el cálculo de predicados sí influye**.

Ejemplo 26. *Demostrar el argumento dado al inicio de la sección,*

$$\{\forall x\ (A(x) \to \neg M(x)),\ A(Pepe)\} \models \neg M(Pepe),$$

equivale a probar que el conjunto $\{\forall x\ (A(x) \to \neg M(x)),\ A(Pepe),\ \neg\neg M(Pepe)\}$ *es inconsistente. Para ello construimos una tabla semántica de*

$$\forall x\ (A(x) \to \neg M(x))\ \wedge\ A(Pepe)\ \wedge\ M(Pepe)$$

y veamos que todas las ramas se cierran:

(1)	$(\forall x\,[A(x) \to \neg M(x)])\ \wedge\ A(Pepe)$		(premisas)
(2)	$\neg(\neg M(Pepe))$		($\neg$ conclusión)
(3)	$A(Pepe)\ \checkmark$		(de 1)
(4)	$\forall x\,[A(x) \to \neg M(x)]$		(de 1)
(5)	$A(Pepe) \to \neg M(Pepe)\ \checkmark$		(de 4 y EU)
(6)	$\neg A(Pepe)_{\,*}$	$\neg M(Pepe)_{\,*}$	(de 5)

Ejemplo 27. *Consideremos el argumento:* "Todo el mundo grita o llora. No todo el mundo llora. Así que algunas personas gritan y no lloran". *Para formalizarlo definimos, sobre el universo de los humanos, los predicados*

$$G(x)\text{: "}x\text{ grita"} \qquad L(x)\text{: "}x\text{ llora"}$$

Las premisas del argumento son $\forall x\ [G(x) \vee L(x)]$ *y* $\neg\forall x\ L(x)$*; mientras que la conclusión es* $\exists x\ [G(x) \wedge \neg L(x)]$.

Para comprobar que el argumento $\{\forall x\,[G(x) \vee L(x)],\, \neg\forall x\, L(x)\} \models \exists x\,[G(x) \wedge \neg L(x)]$ *es válido, comprobamos que se cierran todas las ramas de la tabla semántica de:*

$$\forall x\,[G(x) \vee L(x)]\ \wedge \neg\forall x\, L(x)\ \wedge\ \neg\,\exists x\,[G(x) \wedge \neg L(x)]$$

(1)	$(\forall x\,[G(x) \vee L(x)])\ \wedge\ \neg\forall x\, L(x)$	(premisas)
(2)	$\neg(\exists x\,[G(x) \wedge \neg L(x)] \equiv \forall x\,[\neg G(x) \vee L(x)]$	($\neg$ conclusión)
(3)	$\neg\forall x\, L(x) \equiv \exists x\ \neg L(x)\quad \checkmark$	(de 1)
(4)	$\neg L(c)$	(de 3 y EE)
(5)	$\forall x\,[G(x) \vee L(x)]$	(de 1)
(6)	$G(c) \vee L(c)\ \checkmark$	(de 5 y EU)
(7)	$G(c)$ $\qquad$ $L(c)_{*}$	(de 6)
(8)	$\neg G(c) \vee L(c)\quad \checkmark$	(de 2 y EU)
(9)	$\neg G(c)_{*}$ $\qquad$ $\neg(\neg L(c))_{*}$	(de 8)

Es **importante** destacar que es necesario usar la *Especificación existencial* en la segunda premisa (paso (3) a (4)) antes de utilizar la *Especificación universal* en la primera premisa (paso (5) a (6)).

En el ejemplo siguiente vemos que no se puede intercambiar este orden; es decir, siempre hay que utilizar las especificaciones existenciales antes que las universales.

Ejemplo 28. *Consideremos el argumento* "Algunos poetas fueron románticos. Algunos románticos se suicidaron. Por lo tanto, algunos poetas se suicidaron"*. Si definimos, sobre el universo de los humanos, los predicados siguientes:*

$$P(x)\text{: “}x\text{ es poeta”} \qquad R(x)\text{: “}x\text{ es romántico”} \qquad S(x)\text{: “}x\text{ se suicida”},$$

el argumento dado es: $[\exists x\,(P(x) \wedge R(x))] \wedge [\exists x\,(R(x) \wedge S(x))] \models [\exists x\,(P(x) \wedge S(x))]$.

Teniendo en cuenta que $\neg[\exists x\,(P(x) \wedge S(x))] \equiv \forall x\,(\neg P(x) \vee \neg S(x))$, *si representamos*

$$\{P(x) \wedge R(x),\ R(x) \wedge S(x),\ \neg P(x) \vee \neg S(x)\}$$

se tiene que

$$
\begin{array}{ccc}
 & P(x) & \\
 & | & \\
 & R(x) & \\
 & | & \\
 & R(x) & \\
 & | & \\
 & S(x) & \\
 / & & \backslash \\
\neg P(x) & & \neg S(x) \\
* & & *
\end{array}
$$

Obviamente, esta representación **no es correcta** *(nos hemos olvidado de los cuantificadores y hemos representado los predicados para una variable arbitraria x).*

Hagamos la tabla semántica de forma correcta y veamos que el argumento no es válido. La tabla comienza de la forma siguiente

$$
\begin{array}{ccc}
(1) & \exists x\,(P(x) \wedge R(x)) & \text{(premisa)} \\
 & | & \\
(2) & \exists x\,(R(x) \wedge S(x)) & \text{(premisa)} \\
 & | & \\
(3) & \neg[\exists x\,(P(x) \wedge S(x))] & (\neg\ \text{conclusión})
\end{array}
$$

Para continuar, haremos primero la especificación existencial de (1)

$$
\begin{array}{cc}
\exists x\,(P(x) \wedge R(x)) & \checkmark \\
| & \\
P(a) \wedge R(a) & \checkmark \\
| & \\
P(a) & \\
| & \\
R(a) &
\end{array}
$$

Continuamos con la especificación existencial de (2)

$$
\begin{array}{cc}
\exists x\,(R(x) \wedge S(x)) & \checkmark \\
| & \\
R(b) \wedge S(b) & \checkmark \\
| & \\
R(b) & \\
| & \\
S(b) &
\end{array}
$$

y, por último, aplicando la especificación universal de (3) *a cada una de las variables, teniendo en* *cuenta que* $\neg[\exists x\,(P(x)\wedge S(x))]\equiv\forall x\,[\neg P(x)\vee\neg S(x)]$, *se obtiene la tabla siguiente:*

$$
\begin{array}{ccc}
 & P(a)\wedge R(a)\ \checkmark & \\
 & | & \\
 & P(a) & \\
 & | & \\
 & R(a) & \\
 & | & \\
 & R(b)\wedge S(b)\ \checkmark & \\
 & | & \\
 & R(b) & \\
 & | & \\
 & S(b) & \\
 & | & \\
 & \neg P(a)\vee\neg S(a) & \\
 \swarrow & & \searrow \\
 \neg P(a)_{*} & & \neg S(a) \\
 & & | \\
 & & \neg P(b)\vee\neg S(b)\checkmark \\
 & \swarrow & \searrow \\
 & \neg P(b) & \neg S(b)_{*}
\end{array}
$$

De la tabla semántica anterior, se deduce que el argumento no es válido y un contraejemplo se obtiene siguiendo la rama que queda abierta: así a es un poeta romántico que ***no*** *se suicidó (pues en dicha rama aparecen* $P(a)$, $R(a)$ *y* $\neg S(a)$*) y b es un romántico que se suicida y que* ***no*** *es poeta (*$\neg P(b)$, $R(b)$ *y* $S(b)$*).*

1.2.2 Demostración por inducción

Un predicado básico en muchas teorías es la igualdad "=", que verifica dos axiomas

$$\forall x\ (x=x) \quad \text{y} \quad \forall x\ \forall y\ \big((x=y)\wedge P(x))\to P(y)\big).$$

Además, en los números naturales, existe una regla de demostración específica, conocida como *Principio de inducción*: $\{P(1),\forall x\,(P(x)\to P(x+1))\}\models\forall x\,P(x)$. Este principio se basa en una característica fundamental de los números naturales: que cualquiera de ellos puede ser obtenido a partir del *uno* mediante una suma reiterada de este.

Es decir, para demostrar, en el universo $\mathbb{N}$ de los números naturales, que $\forall n\ P(n)$ es una proposición verdadera, usamos el *principio de inducción* que se enuncia del siguiente modo: Si

- (**Base inductiva**) $P(1)$ es cierta, y
- (**Paso inductivo**) $\forall k\ \ (P(k)\models P(k+1))$

entonces, $\forall n\ P(n)$ es una proposición verdadera.

Por lo tanto, si $P(1)$ es verdadera y, además se cumple que la propiedad P se transmite de cualquier número natural k a su sucesor, $k+1$, entonces todos los números naturales satisfacen la propiedad P.

Hay muchos tipos de imágenes gráficas que pueden ayudar a comprender el principio anterior. Por ejemplo, usando fichas de dominó. Si las colocamos en fila una tras otra, de tal modo que si una ficha cae empuja, y hace caer, a la siguiente (paso inductivo) y tiramos la primera (base inductiva), todas las fichas caerán. Hay que tener en cuenta que una sola de estas condiciones no es suficiente para que todas las fichas caigan.

Ejemplo 29.

1. *Para todo natural* n*, se verifica que* $1+2+\cdots+n=\displaystyle\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.

   - Base inductiva $n=1$: $1=\dfrac{1(1+1)}{2}$
   - Paso inductivo: Sea $k \in \mathbb{N}$ cualquiera y supongamos que el resultado es cierto para k,

   $$\sum_{i=1}^{k} i = 1+2+\cdots+k = \frac{k(k+1)}{2},$$

   y comprobemos si se cumple para $k+1$

   $$\begin{aligned}\sum_{i=1}^{k+1} i &= 1+2+\cdots+k+(k+1)\\ &= \frac{k(k+1)}{2}+(k+1) = \frac{k(k+1)+2(k+1)}{2} = \frac{(k+1)(k+2)}{2}\end{aligned}$$

2. *Probar que para todo natural* n*, se verifica que* $1+2+\cdots+2^n=\sum_{i=0}^{n} 2^i = 2^{n+1}-1$.

   - Base inductiva $n=1$: $1+2=2^{1+1}-1$
   - Paso inductivo: Sea $k \in \mathbb{N}$ cualquiera y supongamos que el resultado es cierto para k,

   $$\sum_{i=0}^{k} 2^i = 1+2+\cdots+2^k = 2^{k+1}-1,$$

   y comprobemos si se cumple para $k+1$

   $$\begin{aligned}\sum_{i=0}^{k+1} 2^i &= \left(\sum_{i=0}^{k} 2^i\right)+2^{k+1} = \left(1+2+\cdots+2^k\right)+2^{k+1}\\ &= \left(2^{k+1}-1\right)+2^{k+1} = 2\cdot 2^{k+1}-1 = 2^{(k+1)+1}-1\end{aligned}$$

En ocasiones, para facilitar la demostración, debemos aplicar el llamado *principio de inducción fuerte o completa*, que es equivalente al anterior, y que se enuncia como sigue:

Sea P una propiedad definida sobre los naturales tal que

- (**Base inductiva**) $P(1)$ es cierta, y

- (**Paso inductivo completo**) para todo natural k se cumple que $\{P(1), P(2), \ldots, P(k)\} \models P(k+1)$

entonces, $\forall n\ P(n)$ es una proposición verdadera.

Ejemplo 30. *Después de transcurrir n meses en un experimento de invernadero, el número p_n de plantas de un tipo particular satisface las ecuaciones $p_1 = 3$, $p_2 = 7$ y*

$$p_n = 3p_{n-1} - 2p_{n-2}$$

para todo $n \geq 3$. Probemos que $p_n = 2^{n+1} - 1$.

En primer lugar, es claro que para los casos $n = 1, 2$ se verifica. Además si $n \geq 3$ y, se supone que para todo $1 \leq k \leq n$ se verifica la hipótesis, entonces

$$p_{n+1} = 3p_n - 2p_{n-1} = 3(2^{n+1} - 1) - 2(2^n - 1) = 3 \cdot 2^{n+1} - 2^{n+1} - 1 = 2^{n+2} - 1.$$

con lo que queda probado.

Los principios de inducción simple y fuerte son válidos también para demostrar que una propiedad es cierta para todos los números enteros mayores que un entero dado n_0, en este caso sin más que cambiar el universo $\mathbb{N}$ por

$$U = \{n \in \mathbb{Z} \ | \ n \geq n_0\}.$$

Nota 1. *Es interesante destacar que tanto la base inductiva como el paso inductivo son necesarios ya que, en ausencia de alguno de ellos el principio de inducción no es cierto.*

Ejemplo 31. *Para cada n, sea $P(n)$ la propiedad que afirma que*

$$\sum_{i=1}^{n} i = \frac{\left(n + \frac{1}{2}\right)^2}{2}$$

Es fácil comprobar que si $P(k)$ es cierta, entonces también es cierta $P(k+1)$:

$$\sum_{i=1}^{k+1} i = \sum_{i=1}^{k} i + (k+1) = \frac{\left(k + \frac{1}{2}\right)^2}{2} + (k+1) = \frac{\left(k + 1 + \frac{1}{2}\right)^2}{2}.$$

Es decir, esta propiedad verifica el paso inductivo *y parecería que es cierta para todos los naturales. Sin embargo, no se cumple para $n = 1$*

$$1 \neq \frac{\left(1 + \frac{1}{2}\right)^2}{2}.$$

De hecho, no es cierta para ningún natural n ya que

$$\sum_{i=1}^{n} i = \frac{n\,(n+1)}{2}, \ \textit{pero}\ \frac{n\,(n+1)}{2} \neq \frac{\left(n + \frac{1}{2}\right)^2}{2}.$$

1.3 Álgebras de Boole

Las Álgebras de Boole se aplican en el análisis de circuitos electrónicos y, por lo tanto, en el diseño de dispositivos digitales como ordenadores, teléfonos móviles, etc. 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 9. *Un* ***Álgebra de Boole*** *es un conjunto* $A = \{a, b, c, \ldots\}$ *con dos operaciones binarias, suma* $(+)$ *y producto* $(\cdot)$*, y una operación unaria, complemento o inversión* $(\bar{\ })$*, que cumplen los siguientes axiomas:*

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

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

A2: *Existen dos elementos distinguidos* $\mathbf{0}$ *y* $\mathbf{1}$ *en A tales que, para todo $a \in A$:*

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

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: para cualesquiera $a, b \in A$*

$$a + b = b + a, \qquad a \cdot b = b \cdot a$$

A5: *Las operaciones suma y producto son asociativas: para cualesquiera $a, b, c \in A$*

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

A6: *La 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 $(A, \mathbf{0}, \mathbf{1}, +, \cdot, \bar{\ })$ *se llama* ***Álgebra de Boole****.*

El Álgebra de Boole más sencilla es la formada por el conjunto $A = \{0, 1\}$ con las operaciones dadas por las siguientes tablas:

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

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

x	$\overline{x}$
0	1
1	0

También es un Álgebra de Boole el conjunto de las proposiciones (o fórmulas bien formadas) con los operadores lógicos $\vee$, $\wedge$ y $\neg$ definidos al inicio del tema, siendo $\bot$ el elemento $\mathbf{0}$ y $\top$ el elemento $\mathbf{1}$.

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 entre sí los operadores ($+$ y $\cdot$) y los elementos distinguidos ($\mathbf{0}$ y $\mathbf{1}$).

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

Proposición 1. *Sea*$(A, \mathbf{0}, \mathbf{1}, +, \cdot, \bar{\ })$ *un Álgebra de Boole. Para cualesquiera* $a, b \in A$ *se verifica:*

1. *Leyes de Idempotencia:* $a + a = a$ *y* $a \cdot a = a$
2. *Leyes de Acotación:* $a + \mathbf{1} = \mathbf{1}$ *y* $a \cdot \mathbf{0} = \mathbf{0}$
3. *Leyes de Absorción:* $a + (a \cdot b) = a$ *y* $a \cdot (a + b) = a$
4. *El elemento* $\bar{a}$ *de* $a \in A$ *es único. Si existe* $b \in A$ *tal que* $a + b = \mathbf{1}$ *y* $a \cdot b = \mathbf{0}$ *entonces* $b = \bar{a}$.
5. *Leyes de Morgan:* $\overline{a + b} = \bar{a} \cdot \bar{b}$ *y* $\overline{a \cdot b} = \bar{a} + \bar{b}$
6. *Involución:* $\overline{(\bar{a})} = a$
7. $a + (\bar{a} \cdot b) = a + b$ *y* $a \cdot (\bar{a} + b) = a \cdot b$.

Para demostrar las Leyes de Idempotencia aplicamos, de forma sucesiva, los axiomas **A2**, **A3**, **A6**, **A3** y **A2** de Álgebra de Boole:

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

De forma dual se obtiene

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

La última propiedad se prueba a partir de los axiomas **A6**, **A3** y **A2** como sigue:

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

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 correspondencias:

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

1.3.1 Funciones de Boole

Partiendo del alfabeto formado por un conjunto de variables y los símbolos $+$, $\cdot$ y $^{-}$, podemos definir un lenguaje, que se corresponde con el lenguaje de la lógica de proposiciones cambiando, como ya se ha comentado, 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[4]:

- $0, 1, x_1, x_2, \ldots, x_n$ (variables) son expresiones booleanas.
- Si E_1 y E_2 son expresiones booleanas, entonces $\overline{E_1}$, $E_1 + E_2$, $E_1 \cdot E_2$ son, también, expresiones booleanas.

Se establece una jerarquía de prioridad entre las operaciones booleanas que permite suprimir paréntesis en una expresión booleana. El orden es el siguiente:

1. Complemento $^{-}$ 2. Producto $\cdot$ 3. Suma $+$

Así, $(x_1 \cdot x_2) + x_3$ se escribe $x_1 \cdot x_2 + x_3$, pero en la expresión $(x_1 + x_2) \cdot x_3$ no pueden suprimirse los paréntesis (en caso de suprimirse, se obtedría $x_1 + x_2 \cdot x_3$, que corresponde a $x_1 + (x_2 \cdot x_3)$).

Definición 10. *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 $A = \{0, 1\}$ hay 2^n posibles combinaciones de entrada $(x_1, x_2, \ldots, x_n)$ 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 32. $f(a, b, c) = \overline{a} + \overline{a} \cdot b \cdot \overline{c}$

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

a	b	c	$f(a,b,c)$	$\overline{a}$
0	0	0	1	1
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	0	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 también simplificando la expresión que define f aplicando propiedades del álgebra de Boole:

$$f(a, b, c) = \overline{a} + \overline{a} \cdot b \cdot \overline{c} = \overline{a} \cdot (1 + b \cdot \overline{c}) = \overline{a} \cdot 1 = \overline{a}.$$

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

Para simplificar la notación, podemos suprimir el $\cdot$ de la operación producto; es decir, escribiremos $x\,y$ para representar $x \cdot y$. Además, hay que tener especial cuidado en la escritura pues:

$$\overline{x\,y} = \overline{x} + \overline{y} \quad \text{es distinto de} \quad \overline{x}\,\overline{y} = \overline{x+y}.$$

Definición 11. *Un* ***literal*** *es una variable booleana (literal positivo: x) o el complemento de una variable booleana (literal negativo: $\bar{x}$).*

Escribiremos una función booleana utilizando **sumas de literales** o **productos de literales** a partir de su tabla de valores. Por ejemplo, en la tabla de valores de $f(x,y,z) = (x+y)\,\bar{z}$ señalamos los productos de literales para los que $f(x,y,z) = 1$ y las sumas de literales para los que $f(x,y,z) = 0$:

	x	y	z	$f(x,y,z)$	
$x+y+z \longleftarrow$	0	0	0	0	
$x+y+\bar{z} \longleftarrow$	0	0	1	0	
	0	1	0	1	$\longrightarrow \bar{x}\,y\,\bar{z}$
$x+\bar{y}+\bar{z} \longleftarrow$	0	1	1	0	
	1	0	0	1	$\longrightarrow x\,\bar{y}\,\bar{z}$
$\bar{x}+y+\bar{z} \longleftarrow$	1	0	1	0	
	1	1	0	1	$\longrightarrow x\,y\,\bar{z}$
$\bar{x}+\bar{y}+\bar{z} \longleftarrow$	1	1	1	0	

Se verifica que $f(x,y,z) = \bar{x}\,y\,\bar{z} + x\,\bar{y}\,\bar{z} + x\,y\,\bar{z}$. Esto es debido a que la suma final es 1 cuando **alguno de los sumandos** es 1. Por otra parte, cada uno de estos sumandos es un producto; y un producto es 1 cuando **todos sus factores** son 1.
Análogamente, $f(x,y,z) = (x+y+z)(x+y+\bar{z})(x+\bar{y}+\bar{z})(\bar{x}+y+\bar{z})(\bar{x}+\bar{y}+\bar{z})$, puesto que el producto final es 0 cuando **alguno de los factores** es 0. Por otro lado, cada uno de los factores es una suma; y una suma es 0 cuando **todos sus sumandos** son 0.

Teorema 1. ***(Teorema de expansión de Shannon)*** *Cualquier función boolena puede expresarse en forma de suma de productos de literales (**Forma Normal Disyuntiva**, FND[5]) o en forma de producto de sumas de literales (**Forma Normal Conjuntiva**, FNC[6]).*

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	$F(x,y,z)$	$G(x,y,z)$
0	0	0	1	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	0	1
1	1	1	0	1

[5] En inglés disjunctive normal form (DNF).
[6] En inglés conjuctive normal form (CNF).

Si elegimos, para cada una de las funciones, los casos en que su valor es 1, obtenemos:

$$F(x,y,z) = \bar{x}\bar{y}\bar{z} + \bar{x}y\bar{z} + \bar{x}yz + x\bar{y}z$$
$$G(x,y,z) = \bar{x}\bar{y}z + \bar{x}y\bar{z} + xy\bar{z} + xyz$$

Esta representación de la función se llama **forma normal disyuntiva** (FND) de la función booleana. Otras formas de escribir $F(x,y,z)$ y $G(x,y,z)$ como suma de productos es:

$$F(x,y,z) = \bar{x}\bar{y}\bar{z} + \bar{x}y + x\bar{y}z = \bar{x}\bar{z} + \bar{x}yz + x\bar{y}z$$
$$G(x,y,z) = \bar{x}\bar{y}z + y\bar{z} + xyz = \bar{x}\bar{y}z + \bar{x}y\bar{z} + xy$$

También podemos elegir los casos en los que F y G tienen valor 0, obteniéndose:

$$F(x,y,z) = (x+y+\bar{z})(\bar{x}+y+z)(\bar{x}+\bar{y}+z)(\bar{x}+\bar{y}+\bar{z})$$
$$G(x,y,z) = (x+y+z)(x+\bar{y}+\bar{z})(\bar{x}+y+z)(\bar{x}+y+\bar{z})$$

Esta representación de la función se llama **forma normal conjuntiva** (FNC) de la función booleana. Otra forma de escribir $F(x,y,z)$ y $G(x,y,z)$ como producto de sumas es:

$$F(x,y,z) = (x+y+\bar{z})(\bar{x}+y+z)(\bar{x}+\bar{y}) = (x+y+\bar{z})(\bar{x}+z)(\bar{x}+\bar{y}+\bar{z})$$
$$G(x,y,z) = (y+z)(x+\bar{y}+\bar{z})(\bar{x}+y+\bar{z}) = (x+y+z)(x+\bar{y}+\bar{z})(\bar{x}+y)$$

En resumen:

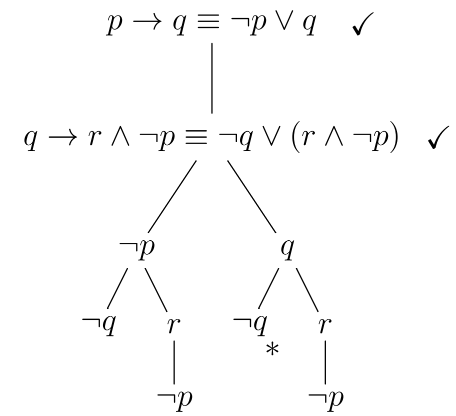
Forma normal	Método de obtención	Convenio
Disyuntiva	Suma de productos de literales cuyas combinaciones hacen 1 la función	0 variable negada 1 variable sin negar
Conjuntiva	Producto de sumas de literales cuyas combinaciones hacen 0 la función	0 variable sin negar 1 variable negada

Tabla 1

De forma análoga, se dice que una fórmula proposicional está en forma normal disyuntiva cuando es una disyunción de conjunciones de literales, y en forma normal conjuntiva si es una conjunción de disyunciones de literales. Por ejemplo, $(p \vee \neg q) \wedge (\neg p \vee r)$ está en forma normal conjuntiva y $(q \wedge \neg r) \vee (p \wedge r)$ en forma normal disyuntiva.

Se pueden utilizar tablas semánticas para obtener una FND de una fórmula proposicional, haciendo la disyunción (suma) de las ramas abiertas, cada rama es la conjunción (producto) de los literales que aparecen en ella. Si todas las ramas se cierran, la fórmula es una contradicción.

Ejemplo 33. *La tabla semántica de la fórmula* $(p \to q) \wedge (q \to r \wedge \neg p)$ *es la siguiente:*



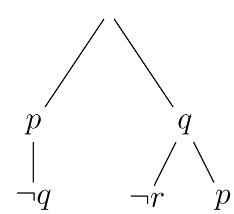
Esta tabla tiene tres ramas abiertas con los conjuntos de átomos $\{\neg q, \neg p\}$, $\{r, \neg p\}$ *y* $\{q, r, \neg p\}$, *por lo que* $(p \to q) \wedge (q \to r \wedge \neg p)$ *es ógicamente equivalente a la siguiente disyunción de conjunciones:*

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

Nótese que las ramas abiertas proporcionan los modelos de la fórmula dada.

Dualmente, para obtener una FNC de $\mathcal{P}$ se construye la tabla semántica de $\neg\mathcal{P}$ y se hace la conjunción de la diyunción de los complementarios de los átomos que aparecen en las ramas abiertas de dicha tabla. Volvamos al ejemplo anterior: $(p \to q) \wedge (q \to r \wedge \neg p)$.

Ejemplo 34. *Se halla primero la tabla semántica de su negación,* $\neg\big((p \to q) \wedge (q \to r \wedge \neg p)\big)$, *que es:*



puesto que

$$\neg((p \to q) \wedge (q \to r \wedge \neg p)) \;\equiv\; (p \wedge \neg q) \vee (q \wedge (\neg r \vee p)).$$

Como vemos, se obtienen tres ramas abiertas con los conjuntos de átomos $\{p, \neg q\}$, $\{q, \neg r\}$ *y* $\{p, q\}$. *Construyendo la conjunción de la diyunción de los complementarios de los átomos que aparecen en esas ramas se obtiene una FNC de la fórumula inicial,* $(p \to q) \wedge (q \to r \wedge \neg p)$*:*

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

Nótese que las ramas abiertas en la tabla de $\neg\mathcal{P}$, *es decir,* $\{p, \neg q\}$, $\{q, \neg r\}$ *y* $\{p, q\}$, *proporcionan los contramodelos (contrajemplos) de la fórmula inicial* $\mathcal{P}$ *(los modelos de* $\neg\mathcal{P}$ *son los contramodelos de* $\mathcal{P}$*).*

1.3.2 Puertas lógicas básicas

Estos dispositivos de estado sólido, son los bloques elementales para la construcción de circuitos lógicos. El nombre de *puertas* responde al hecho de que pueden tener una o varias entradas pero una única 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:

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

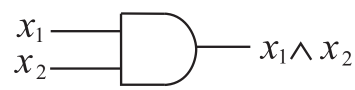


Figura 1. Puerta AND

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

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

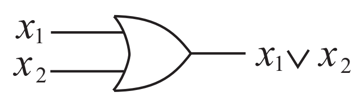


Figura 2. Puerta OR

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

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

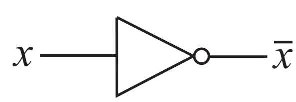


Figura 3. Puerta NOT

La **tabla lógica** de un circuito combinatorio muestra la salida que se obtiene para cada una de las posibles entradas.

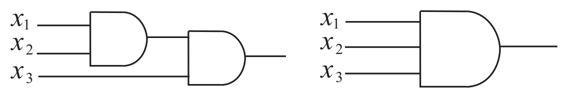
x_1	x_2	$x_1 \wedge x_2$
0	0	0
0	1	0
1	1	1
1	0	0

x_1	x_2	$x_1 \vee x_2$
0	0	0
0	1	1
1	1	1
1	0	1

x	$\overline{x}$
0	1
1	0

Son las tablas 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, en concreto, la propiedad asociativa. Esto nos permite decir, por ejemplo, que la función de una puerta AND de tres entradas, es la misma del circuito formado por dos puertas AND de dos entradas conectadas según indica la figura:



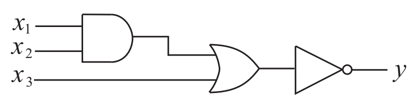
$(x_1 \wedge x_2) \wedge x_3$ $\qquad$ $x_1 \wedge x_2 \wedge x_3$

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, podremos escribir un expresión booleana para representar la salida. 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.

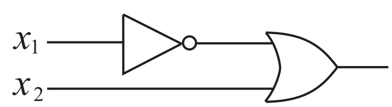
Ejemplo 35. *La función booleana*

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

se corresponde con el circuito lógico siguiente:



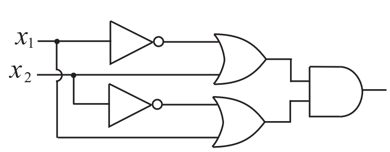
Los operadores lógicos condicional ($\rightarrow$) y bicondicional ($\leftrightarrow$) estudiados, también se pueden implementar utilizando circuitos combinacionales. El circuito



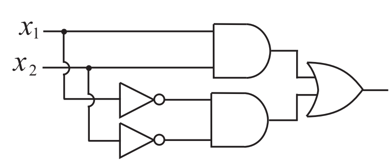
con salida $\neg x_1 \vee x_2$, se corresponde con el operador lógico condicional ($x_1 \rightarrow x_2$). En el caso del operador lógico bicondicional

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

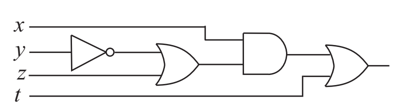
se puede representar por



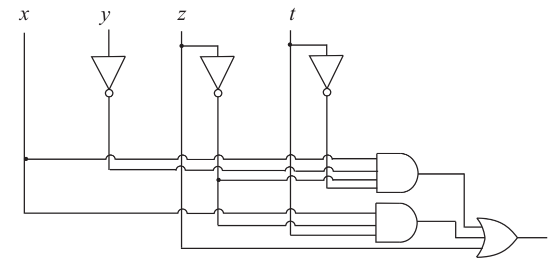
o bien por



A continuación se dibujan un par de circuitos lógicos, cada uno con la función que describe su comportamiento. Por ejemplo, la función $F(x, y, z, t) = x(\bar{y} + z) + t$ se corresponde con el circuito



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



1.3.3 Minimización de funciones. Minimización de circuitos

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

- No existe otra expresión de la función con menor número de sumandos.
- Cualquier otra expresión con el mismo número de sumandos tendrá más variables dentro de alguno de esos sumandos.

Si la expresión es un producto de términos suma, esta será minimal si cumple las condiciones anteriores cambiando la palabra sumando por factor.

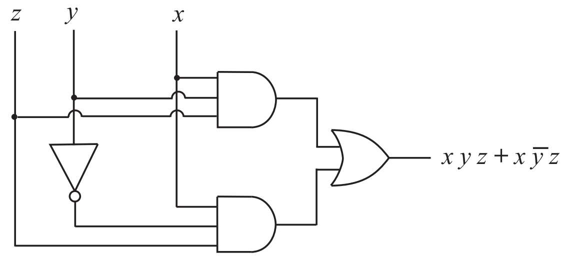
La eficiencia de un circuito combinacional depende del número de puertas que tenga y de la disposición de estas. El proceso de diseñar un circuito combinacional comienza con la tabla que especifica las salidas para cada combinación de valores de entrada. Para obtener un conjunto de puertas lógicas que implemente este circuito siempre podemos usar la forma normal disyuntiva del circuito. Pero, la forma normal disyuntiva puede tener 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

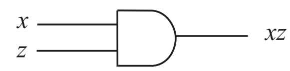
x	y	z	$f(x,y,z)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

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

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

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





Este ejemplo muestra que combinar sumandos de una forma normal disyuntiva de un circuito puede dar lugar una expresión más sencilla del mismo. 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 **diagramas** o **mapas de Karnaugh** son un método desarrollado en la década de los cincuenta del siglo pasado para ayudar a minimizar los circuitos de forma manual; este método proporciona un método visual para simplificar la forma normal disyuntiva (o la forma normal conjuntiva, como veremos en algunos ejemplos, aunque nos centraremos en la FND) de la función representada por el circuito.

Un diagrama de Karnaugh de una función booleana es una presentación alternativa de la misma información contenida en una tabla de valores de la función; está constituido por una cuadrícula en forma de encasillado cuyo número de casillas depende del número de variables que tenga la función a simplificar.

En cada cuadrado del diagrama de Karnaugh representaremos el valor que toma la función para la combinación de variables que le corresponde, puesto que hay una correspondencia uno a uno entre los cuadrados y las distintas combinaciones de entrada. Por ejemplo, si consideramos la función de dos variables $f(x, y) = \bar{x} + y$, la correspondencia entre su tabla de valores y su diagrama de Karnaugh es la siguiente:

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

$x \backslash y$	0	1
0	1	1
1	0	1

En una FND de una función booleana de **dos variables** puede haber hasta cuatro posibles productos de literales, utilizando en cada sumando las dos variables (cada una ellas puede aparecer negada o no):

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

Por tanto, un diagrama de Karnaugh para una función de dos variables consta de cuatro celdas (tantas como posibles productos de literales con las dos variables). Se coloca un 1 en la celda que representa a un producto de literales si este aparece en la FND de la función.

Por ejemplo, para la función anterior $f(x, y) = \bar{x} + y$, la FND $f(x, y) = \bar{x}\,\bar{y} + \bar{x}\,y + x\,y$ se corresponde con su mapa de Karnaugh:

	x	y	f
$\bar{x}\,\bar{y} \longrightarrow$	0	0	1
$\bar{x}\,y \longrightarrow$	0	1	1
	1	0	0
$x\,y \longrightarrow$	1	1	1

$x \backslash y$	0	1
0	1	1
1	0	1

En el caso dual, una FNC de una función booleana de **dos variables** puede tener hasta cuatro posibles sumas de literales (cada una de las dos variables puede aparecer negada o no):

$$x + y, \qquad \bar{x} + y, \qquad \bar{x} + \bar{y}, \qquad x + \bar{y}.$$

En el mapa de Karnaugh se coloca un 0 en la celda que representa a una suma de literales si esta aparece en la FNC de la función.

Por ejemplo, se puede obtener una FNC de la función $g(x, y) = x\,y$ a partir de su tabla de valores o su correspondiente mapa de Karnaugh. Resultando: $g(x, y) = (x + y)\,(\bar{x} + y)\,(x + \bar{y})$.

	x	y	g
$x + y \longrightarrow$	0	0	0
$x + \bar{y} \longrightarrow$	0	1	0
$\bar{x} + y \longrightarrow$	1	0	0
	1	1	1

$x \backslash y$	0	1
0	0	0
1	0	1

Diremos que dos **celdas** son **adyacentes** si los productos (resp. sumas) de literales que representan difieren exactamente en un literal. Por ejemplo, la celda que representa a $\bar{x}y$ es adyacente a la celda que representa $\bar{x}\bar{y}$ (y también a xy) pero $\bar{x}\bar{y}$ y xy no son adyacentes (resp. la celda $x + y$ es adyacente a la celda $x + \bar{y}$, y también a $\bar{x} + y$, pero $x + \bar{y}$ y $\bar{x} + y$ no son adyacentes). En el mapa de Karnaugh, las celdas adyacentes son celdas consecutivas horizontal o verticalmente:

$x \backslash y$	0	1
0	1 ($\bar{x}\,\bar{y}$)	1 ($\bar{x}\,y$)
1	0 ($x\,\bar{y}$)	1 ($x\,y$)

Son adyacentes:
$\bar{x}\,\bar{y} + \bar{x}\,y = \bar{x}$

$$f(x, y) = \bar{x}\,\bar{y} + \bar{x}\,y + x\,y$$
$$= \bar{x} + xy$$

$x \backslash y$	0	1
0	0 ($x + y$)	0 ($x + \bar{y}$)
1	0 ($\bar{x} + y$)	1 ($\bar{x} + \bar{y}$)

Son adyacentes:
$(x + y)\,(\bar{x} + y) = y$

$$g(x, y) = (x + y)\,(\bar{x} + y)\,(x + \bar{y})$$
$$= y\,(x + \bar{y}) = y\,x + y\,\bar{y} = y\,x$$

Ejemplo 36. *Los diagramas de Karnaugh de funciones siguientes*

1. $f(x, y) = xy + \bar{x}y$
2. $f(x, y) = x\bar{y} + \bar{x}y$
3. $f(x, y) = x\bar{y} + \bar{x}y + \bar{x}\bar{y}$

son, respectivamente:

$x \backslash y$	0	1
0		1
1		1

$x \backslash y$	0	1
0		1
1	1	

$x \backslash y$	0	1
0	1	1
1	1	

Para simplificar una función representada en un mapa de Karnaugh, agruparemos dentro de una línea cerrada las casillas que se pueden combinar (adyacentes) y luego calcularemos la correspondiente simplificación (un producto de literales). El objetivo es identificar los bloques de mayor tamaño posible y cubrir todos los *unos* del diagrama con el menor número posible de bloques, usando los bloques de **mayor tamaño posible**.

Si combinamos **dos celdas adyacentes, reducimos una variable**:

$x \backslash y$	0	1
0		1
1		1

$f(x,y) = \bar{x}\,y + x\,y = (\bar{x}+x)\,y = y$

$x \backslash y$	0	1
0	1	1
1	1	

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

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

$f(x,y) = \bar{x}\,\bar{y} + \bar{x}\,y + x\,\bar{y} = \bar{x} + \bar{y}$

Para funciones de **tres variables**, el diagrama de Karnaugh tiene $8 = 2^3$ celdas (pues 8 son los productos de literales distintos que se pueden hacer):

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

Las celdas adyacentes representan productos que difieren en un único literal.

Nota: Este diagrama puede verse como si estuviese dibujado sobre un cilindro; la celda $\bar{x}\,y\,\bar{z}$ es adyacente a $\bar{x}\,\bar{y}\,\bar{z}$, igualmente $x\,y\,\bar{z}$ es adyacente a $x\,\bar{y}\,\bar{z}$.

Los bloques de *dos celdas adyacentes* se pueden combinar para dar lugar a un producto de dos literales, es decir, *se reduce una variable*. Por ejemplo: $x\,y\,z + x\,\bar{y}\,z = x\,z$.

Los bloques de celdas de tamaños 2×2 y 4×1 representan productos que se pueden combinar entre sí para dar lugar a un único literal, es decir, se *reducen dos variables*. Por ejemplo:

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

$\bar{x}$

$\bar{y}$

Las operaciones correspondientes son:

para obtener $\bar{x}$

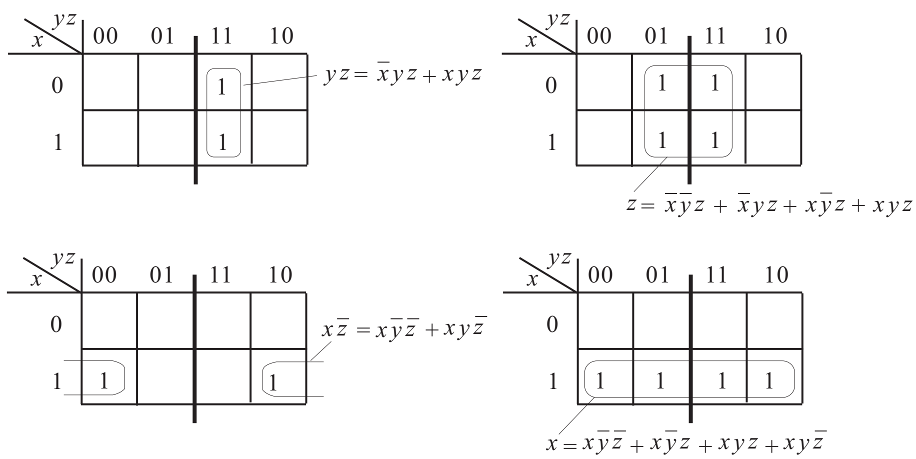
$$\bar{x}\,\bar{y}\,\bar{z}+\bar{x}\,\bar{y}\,z+\bar{x}\,y\,z+\bar{x}\,y\,\bar{z}=\bar{x}\,\bar{y}\,(\bar{z}+z)+\bar{x}\,y\,(z+\bar{z})=\bar{x}\,\bar{y}+\bar{x}\,y=\bar{x}\,(\bar{y}+y)=\bar{x};$$

y para obtener $\bar{y}$

$$\bar{x}\,\bar{y}\,\bar{z}+\bar{x}\,\bar{y}\,z+x\,\bar{y}\,\bar{z}+x\,\bar{y}\,z=\bar{x}\,\bar{y}\,(\bar{z}+z)+x\,\bar{y}\,(\bar{z}+z)=\bar{x}\,\bar{y}+x\,\bar{y}=(\bar{x}+x)\,\bar{y}=\bar{y}.$$

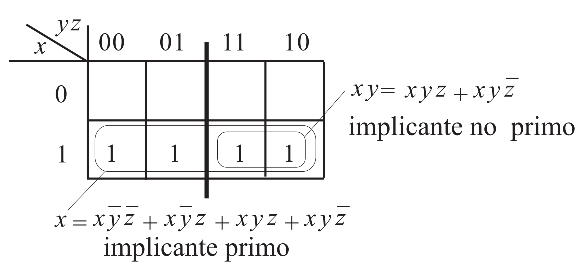
El bloque de las 8 celdas representa un término sin literales, es decir, la función 1.

Ejemplo 37.

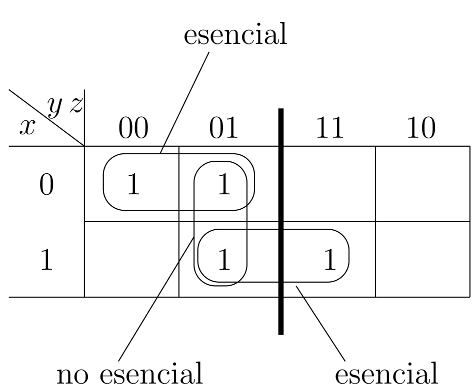


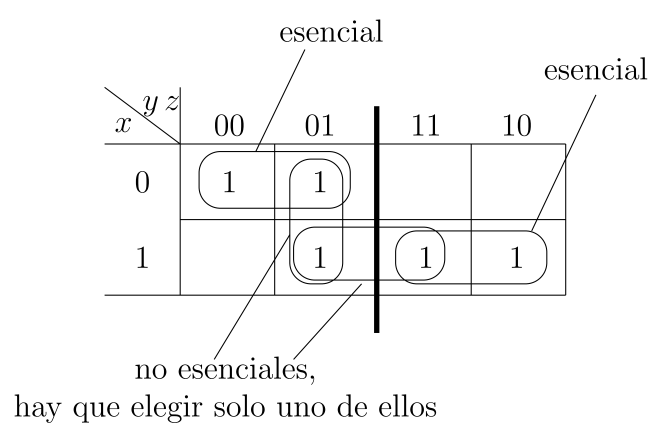
Al producto de literales obtenido al combinar las celdas correspondientes a un bloque de *unos* en el diagrama se le llama **implicante** de la función que se va a minimizar. Se dice que es un **implicante primo** si ese bloque no está contenido en ningún bloque de *unos* mayor.

Ejemplo 38.



Un implicante primo se dice **implicante primo esencial** si cubre una casilla del diagrama de forma exclusiva:



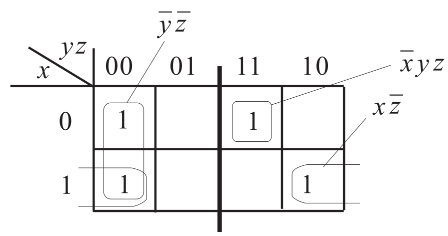


Para simplificar una función, siempre se eligen los bloques de mayor tamaño posible, pero siempre debemos elegir un bloque si es un **implicante primo esencial**. Cubrir todos los *unos* del diagrama con bloques correspondientes a implicantes primos nos permite expresar la suma de productos como una suma de implicantes primos.

Puede haber más de una forma de cubrir todos los *unos* utilizando el menor número posible de bloques, dando así lugar a distintas simplificaciones para una misma función. En la figura anterior, podemos elegir uno de los dos implicantes no esenciales, por lo que se obtienen dos simplificaciones: $f(x,y,z) = \bar{x}\,\bar{y} + \bar{y}\,z + x\,y$ y $f(x,y,z) = \bar{x}\,\bar{y} + x\,z + x\,y$.

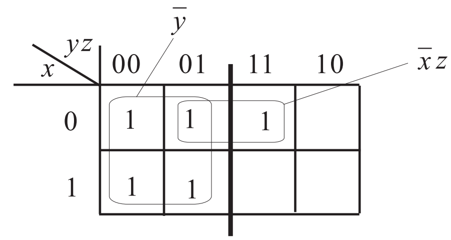
Ejemplo 39.

1. $f(x,y,z) = x\,y\,\bar{z} + x\,\bar{y}\,\bar{z} + \bar{x}\,y\,z + \bar{x}\,\bar{y}\,\bar{z}$



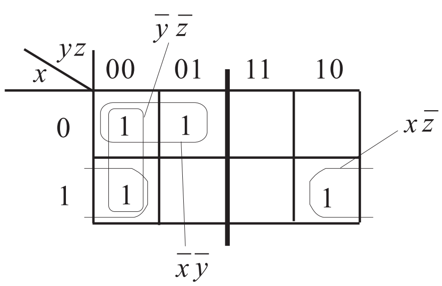
$f(x,y,z) = \bar{x}\,y\,z + \bar{y}\,\bar{z} + x\,\bar{z}$

2. $f(x,y,z) = \bar{x}\,\bar{y}\,\bar{z} + \bar{x}\,\bar{y}\,z + \bar{x}\,y\,z + x\,\bar{y}\,\bar{z} + x\,\bar{y}\,z$



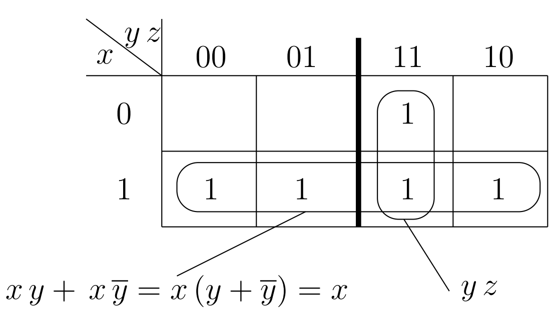
$f(x,y,z) = \bar{y} + \bar{x}\,z$

3. $f(x,y,z) = \bar{x}\,\bar{y}\,\bar{z} + \bar{x}\,\bar{y}\,z + \bar{x}\,y\,z + x\,\bar{y}\,\bar{z} + x\,\bar{y}\,z + x\,y\,z + x\,y\,\bar{z}$



Los implicantes $\bar{x}\,\bar{y}$ y $x\,\bar{z}$ son esenciales, pero $\bar{y}\,\bar{z}$ es un implicante primo no esencial, pues las celdas que cubre quedan cubiertas por los otros dos implicantes primos. Por lo tanto: $f(x,y,z) = x\,\bar{z} + \bar{x}\,\bar{y}$.

4. $f(x,y,z) = (x + z)\ y\ +\ x\ \bar{y}$

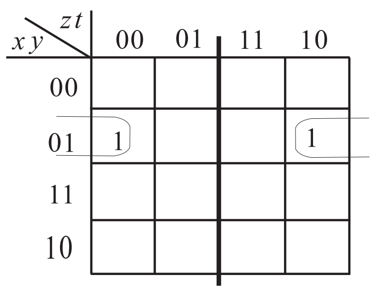


$f(x,y,z) = x\,y + z\,y + x\,\bar{y}$ toma valor 1 cuando es 1 cualquiera de sus sumandos. Cada sumando es un producto de literales, por lo tanto, un sumando es 1 cuando todos los literales tienen valor 1. Por ejemplo, $x\,y$ es 1 cuando lo son x e y, independientemente del valor de z. La forma simplificada es: $f(x,y,z) = x + y\,z$.

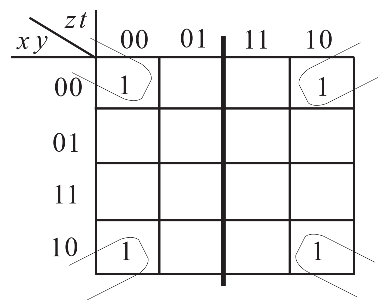
Un diagrama de Karnaugh para **4 variables** es un cuadrado dividido en 16 celdas, cada una de ellas representa uno de los 16 productos posibles de 4 literales.

$xy \backslash zt$	00	01	11	10
00	$\bar{x}\bar{y}\bar{z}\bar{t}$	$\bar{x}\bar{y}\bar{z}t$	$\bar{x}\bar{y}zt$	$\bar{x}\bar{y}z\bar{t}$
01	$\bar{x}y\bar{z}\bar{t}$	$\bar{x}y\bar{z}t$	$\bar{x}yzt$	$\bar{x}yz\bar{t}$
11	$xy\bar{z}\bar{t}$	$xy\bar{z}t$	$xyzt$	$xyz\bar{t}$
10	$x\bar{y}\bar{z}\bar{t}$	$x\bar{y}\bar{z}t$	$x\bar{y}zt$	$x\bar{y}z\bar{t}$

Dos celdas son adyacentes si, y solo si, difieren en un único literal. Por tanto, cada celda es adyacente a otras cuatro celdas. El diagrama en 4 variables puede considerarse como si estuviese dibujado sobre un toro[7], de modo que las celdas adyacentes tengan una frontera común. La simplificación en este caso se realiza identificando aquellos bloques de 2, 4, 8 o 16 celdas que representan términos que se pueden combinar entre sí, como se ve en el ejemplo siguiente:

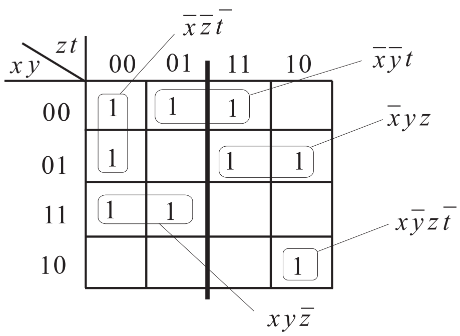


$\bar{x}y\bar{z}\bar{t} + \bar{x}yz\bar{t} = \bar{x}y\bar{t}$

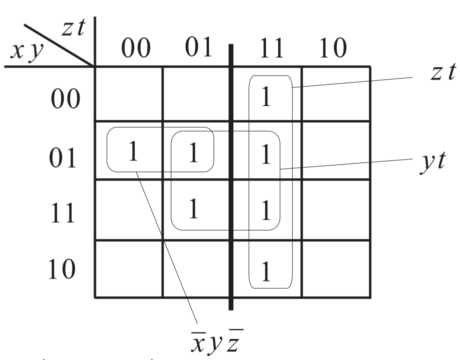


$\bar{x}\bar{y}\bar{z}\bar{t} + \bar{x}\bar{y}z\bar{t} + x\bar{y}\bar{z}\bar{t} + x\bar{y}z\bar{t} = \bar{y}\bar{t}$

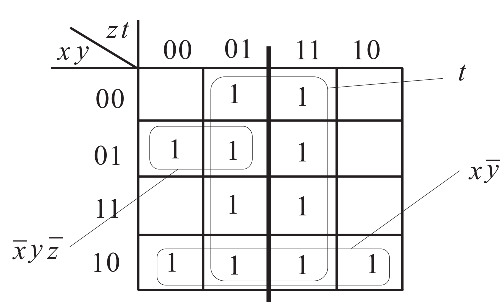
Ejemplo 40.



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



$f(x,y,z,t) = z\,t + y\,t + \bar{x}\,y\,\bar{z}$



$f(x,y,z,t) = t + \bar{x}\,y\,\bar{z} + x\,\bar{y}$

[7] http://es.wikipedia.org/wiki/Toro_%28geometr%C3ADa%29

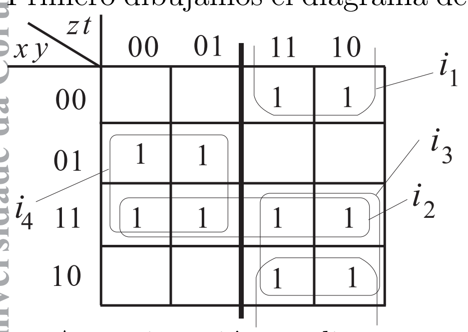
Supongamos que queremos simplificar la función $f(x, y, z, t)$ cuya tabla de valores es

x	y	z	t	$f(x,y,z,t)$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

a partir de la FND:

$$f(x,y,z,t) = \bar{x}\,\bar{y}\,z\,\bar{t} + \bar{x}\,\bar{y}\,z\,t + \bar{x}\,y\,\bar{z}\,\bar{t} + \bar{x}\,y\,\bar{z}\,t + x\,\bar{y}\,z\,\bar{t} + x\,\bar{y}\,z\,t + x\,y\,\bar{z}\,\bar{t} + x\,y\,\bar{z}\,t + x\,y\,z\,\bar{t} + x\,y\,z\,t$$

Primero dibujamos el diagrama de Karnaugh correspondiente identificando los implicantes primos:



El implicante $i_1 = \bar{y}\,z$ es un implicante primo esencial; si no se considera, quedarán *unos* sin cubrir (tiene *unos* no compartidos).
El implicante $i_4 = y\,\bar{z}$ también es un implicante primo esencial. Por tanto, comenzamos eligiendo i_1 e i_4.

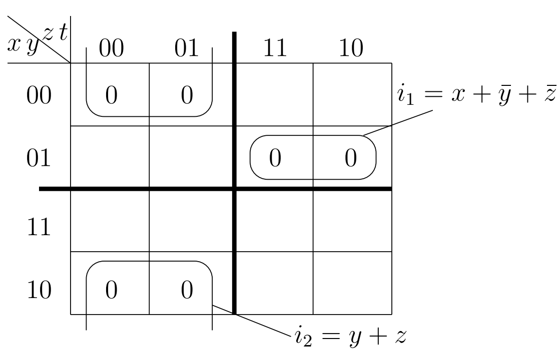
A continuación se elige entre $i_2 = x\,y$ y $i_3 = x\,z$ para cubrir los *unos* correspondientes a las casillas 15 ($f(1,1,1,1) = 1$) y 14 ($f(1,1,1,0) = 1$). Quedarían pues dos simplificaciones posibles

$$f(x,y,z,t) = i_1 + i_4 + i_2 = \bar{y}\,z + y\,\bar{z} + x\,y \quad \text{y} \quad f(x,y,z,t) = i_1 + i_4 + i_3 = \bar{y}\,z + y\,\bar{z} + x\,z.$$

Si queremos simplificar $f(x, y, z, t)$ a partir de la FNC siguiente

$$f(x,y,z,t) = (x+y+z+t)\,(x+y+z+\bar{t})\,(x+\bar{y}+\bar{z}+t)\,(x+\bar{y}+\bar{z}+\bar{t})\,(\bar{x}+y+z+t)\,(\bar{x}+y+z+\bar{t})\,,$$

el diagrama de Karnaugh correspondiente es:



Como los dos implicantes primos que hay son implicantes primos esenciales, la única simplificación posible es

$$f(x,y,z,t) = i_1 \cdot i_2 = (x + \bar{y} + \bar{z})\,(y+z)$$

1.3.4 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 productos de literales, 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.

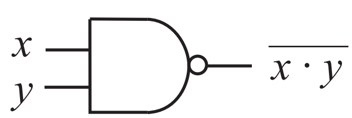
El conjunto $\{+, \cdot\}$ no es funcionalmente completo puesto que es imposible expresar la función booleana $F(x) = \bar{x}$ utilizando estos dos operadores.

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

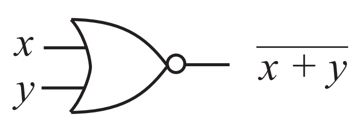
x	y	$x \uparrow y$
0	0	1
0	1	1
1	0	1
1	1	0

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

y representados en los circuitos combinacionales por



Puerta NAND



Puerta NOR

El conjunto $\{\uparrow\}$ es funcionalmente completo puesto que.

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

Análogamente, el conjunto $\{\downarrow\}$ es funcionalmente completo (se deja como ejercicio demostrar que esta afirmación es cierta).