El Álgebra de Boole y el Diseño de Computadoras

René A Hernández Toledo
Colegio Universitario de Cayey

25 de febrero de 1997

1. Introducción

Se oye, a menudo, decir que el Álgebra de Boole y las computadoras están relacionadas. Pero no se oye tanto una explicación acerca de esa relación. ¿Qué hizo Boole? ¿Como se aplica su álgebra al diseño de computadoras?

Queremos, en este artículo, responder de manera simple a esas preguntas. Veremos, en la sección 2, un recordatorio del Álgebra de Boole y sus relaciones con la Lógica. Luego, en la sección 3, hablaremos de "máquinas" que implementan los conectivos lógicos, para continuar en la próxima sección con la implementación de cualquier función booleana. En la sección 5, última de las secciones, aplicaremos lo anterior a la construcción de una máquina electrónica de sumar.

2. La Lógica y el Álgebra de Boole

George Boole (1815–1864) fue un matemático inglés que junto con sus colegas—de la llamada escuela inglesa de Álgebra, intentaban algebrizar todo lo conocido. De los trabajos de esos matemáticos surgieron las teoría de los sistemas de ecuaciones lineales, incluyendo las matrices y los determinantes; así como el estudio de los vectores y el análisis vectorial. Le correspondió a Boole intentar la algebrización de la Lógica. Los resultados de sus investigaciones aparecieron en un libro titulado "An Investigation on the Laws of Thought". A continuación, presentaremos en forma breve los elementos de la lógica y el álgebra de Boole necesarios para el entendimiento posterior.

2.1. La Lógica

El estudio de la Lógica comienza con las *proposiciones*, que son aquellas expresiones verbales o escritas que aseveran o niegan un hecho o relación. De forma más precisa, las proposiciones son aquellas oraciones para las cuales tiene sentido preguntarse si son verdaderas o falsas. Por ejemplo,

- Ayer llovió en San Juan
- Hoy es jueves.
- $2 + 2 = 5$.

Un primer punto a destacar es que hay frases u oraciones que no son del tipo anterior, por ejemplo:

¿Qué día es hoy? o ¡Oh, bella rosa!

no son proposiciones; ya que no puedo decir si son verdaderas o falsas. La algebrización de la Lógica comienza estableciendo que representaremos las proposiciones, al igual que los números en álgebra, por letras; digamos p, q, r. Supondremos que cada proposición puede ser verdadera o falsa (pero no ambas). Representaremos esas situaciones, asignando el valor 1 a una proposición verdadera ($p = 1$) y el valor 0 a una proposición falsa.

Así, como hay operaciones en el álgebra usual, hay "operaciones" en Lógica, a las que llamamos *conectivas*. Usando conectivas construiremos nuevas proposiciones a partir de dos proposiciones dadas. Los conectivos usuales son "o", "y", "si ... entonces ...". Consideremos el conectivo "y", llamado técnicamente la *conjunción*. Si tenemos dos proposiciones, digamos p = "hoy es jueves", q = "hoy está lloviendo", entonces la proposición p y q nos dice que "hoy es jueves y (hoy) está lloviendo". Las conjunciones se consideran verdaderas, cuando sus dos componentes lo son. En el ejemplo, p y q será verdadera cuando, y sólo cuando, se cumpla que es jueves y esté lloviendo. Este convenio se presenta en la siguiente tabla.

p	q	p AND q
0	0	0
0	1	0
1	0	0
1	1	1

El uso de **AND** para este conectivo es tradicional en la electrónica digital. A continuación, presentaremos las tablas de otros conectivos:

p	q	p OR q	p XOR q
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

En esa tabla, OR y XOR denotan las dos modalidades lógicas correspondientes al "o" de nuestro idioma. OR es la modalidad *incluyente*: uno o el otro o ambos; mientras que XOR es la modalidad *excluyente*: uno o el otro, pero no ambos.

Finalmente, es importante para nuestros efectos considerar el conectivo asociado a la negación. Donde por negación de una proposición queremos decir la proposición cuyo valor es precisamente el opuesto al valor de la proposición original.

p	NOT p
0	1
1	0

2.2. El Álgebra de Boole

Boole se dio cuenta que podía asociar operaciones algebraicas a las conectivas de la siguiente manera. Si consideramos la tabla del AND, veremos que los resultados se pueden obtener por la multiplicación usual de los valores de las componentes. Por lo que podemos decir que el AND es representable algebraicamente por la multiplicación. La suma representará entonces al OR o al XOR. Sin embargo, aquí la situación es un poco más complicada debido a la entrada de $1 + 1$, que debiera ser 1 para el OR y 0 para el XOR. Si dejamos por un momento esta situación en suspenso, podemos observar que con esas interpretaciones (usando + ya sea para OR o para XOR) se cumplen la totalidad de las propiedades algebraicas importantes. A saber,

Conmutatividad:	$x + y = y + x$	$xy = yx$
Asociatividad:	$x + (y + z) = (x + y) + z$	$x(yz) = (xy)z$
Distributividad:	$x(y + z) = xy + xz$	
Neutros:	$x + 0 = x$	$x1 = x$

¿Cómo podemos verificar que tales relaciones son ciertas para las proposiciones? Sustituyendo los posibles valores o combinaciones de valores para ambos lados, y viendo que siempre dan lo mismo. Como la Lógica no es lo mismo que la aritmética, Boole buscó propiedades que las distinguieran. Resultó que basta con una sola propiedad. Si decimos,"algo y algo", realmente somos redundantes, bastaría con decir "algo". En términos lógicos, si x es verdadero también lo es x AND x; igualmente, cuando x es falso, también lo es x AND x. Algebraicamente, tendremos el postulado de Boole,

$$x \cdot x = x$$

¿Qué conclusiones podremos sacar de esta propiedad adicional?

1. Notemos que $xx = x$ implica que $4 = 2 * 2 = 2$, de donde restando, 2 en ambos lados; obtendremos que $2 = 0$.

   Por lo tanto, $3 = 2 + 1 = 0 + 1 = 1$, $4 = 2 = 0$, $5 = 4 + 1 = 0 + 1 = 1$, etc. Todos los números se reducen a 0 o 1.

2. Como $x^2 = x$, $x^3 = x^2x = xx = x$, $x^4 = x^3x = xx = x$, etc. O sea, todas las potencias naturales de x son iguales a x.

3. Finalmente, como $x + x = 2x = 0$, concluimos que $x = -x$.

De la relación $1 + 1 = 2 = 0$, concluimos que el $+$ representa al XOR. Sin embargo, esto no quiere decir que no podamos representar algebraicamente al OR. Ya que esto lo haremos con la ayuda de la siguiente definición.

$$x \oplus y := x + y + xy,$$

(uno o el otro o ambos[1]) La negación de x puede representarse como $x' = 1 + x$.

[1] Actualmente, en ingeniería, se usa el convenio inverso, usualmente $+$ es el OR y $\oplus$ es XOR

Podemos resumir la discusión anterior en la siguiente definición.

Definición. El Álgebra de Boole consiste de un conjunto $\{0, 1\}$, junto con las operaciones de suma, resta y multiplicación, donde además de las propiedades usuales, se cumple que: $xx = x$

2.3. Aplicaciones del Álgebra de Boole a la Lógica

El Álgebra de Boole puede aplicarse a la determinación de proposiciones válidas o tautologías. Una proposición es una tautología cuando independiente del valor de sus partes, su valor siempre es verdadero.

Ejemplo 2.1. El *principio del tercero excluido*, afirma que algo es o no es (y que no hay, por lo tanto, otras posibilidades). Simbólicamente, tenemos que estamos afirmando p OR p'.

$$\begin{aligned} p \text{ OR } p' &= p + p' + pp' \\ &= p + 1 + p + p(1 + p) \\ &= p + 1 + p + p + pp \\ &= 1 + p + p + p + p \\ &= 1 + 4p \\ &= 1 \end{aligned}$$

3. Los Circuitos Lógicos

Una de las maneras más efectivas de realizar de forma concreta las operaciones del Álgebra de Boole es mediante los circuitos lógicos. Los circuitos lógicos son circuitos electrónicos especializados, que para nuestros efectos consistirán inicialmente de una fuente de potencia (típicamente una batería), un indicador (una bombilla) e interruptores. Mirar la figura 1.

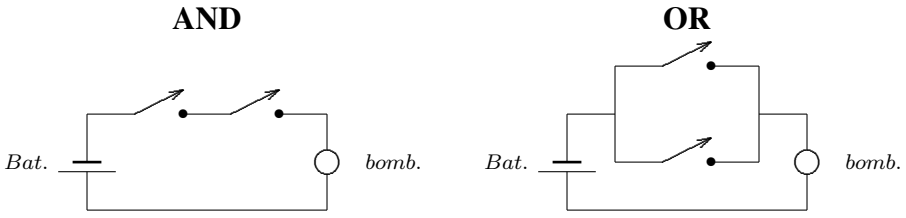

Figura 1: Los circuitos AND y OR

En esos circuitos, supondremos que los interruptores, que son las flechitas, representan las proposiciones. Interruptor cerrado querrá decir verdadero (1), mientras que interruptor abierto es falso (0). Una *tautología* o expresión lógica verdadera consistirá de una combinación de interruptores que hace que la bombilla esté prendida, independientemente del valor de los interruptores.

Consideremos, ahora, a los circuitos mostrados en la figura 1. En el circuito de la izquierda, la bombilla enciende cuando, y sólo cuando, ambos interruptores están cerrados; por lo que podremos usarlo para representar el AND. Por su parte, el circuito de la derecha representa un OR, ya que la bombilla se prenderá, cuando al menos uno de los interruptores esté cerrado. En caso contrario, se apagará la bombilla. Finalmente, la negación puede realizarse con un par de interruptores, acoplados de tal manera que cuando uno se cierra el otro se abre, y viceversa.

Históricamente, los interruptores se construyeron usando electroimanes (relays); para posteriormente pasar a construirse mediante válvulas electrónicas; continuando con transistores y finalmente mediante circuitos integrados (llamados *fichas* o *chips*). En tiendas especializadas tales como RADIO SHACK es posible comprar ANDs, ORs, y NOTs, bajo el nombre de compuertas (en inglés, "gates") lógicas.

4. La realización de funciones booleanas

Una *función booleana* es cualquier expresión válida en el Álgebra de Boole. Por ejemplo,

$$f(x, y, z) = x'y + z \qquad g(x, y) = x' \oplus y.$$

Una función tal como la última g se implementa de la manera mostrada en la figura 2.

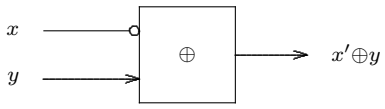

Figura 2: Implementación de $g(x, y) = x' \oplus y$

En tales representaciones, las cajitas cuadradas representan la conectiva cuya operación está indicada. Las líneas que conectan las cajas son los alambres de las conexiones. El círculo a la entrada o salida de una caja indica que la señal correspondiente está negada.

4.1. El análisis de circuitos

El análisis de un circuito consiste en determinar la función realizada por dicho circuito. Por ejemplo, consideremos el circuito ilustrado en la figura 3. Mirando a las entradas y rastreando las señales, podremos determinar la función booleana que se realiza. En el ejemplo, el circuito realiza la función $f(x, y) = xy' \oplus x'y$.

4.2. La síntesis de circuitos

Llamamos síntesis al problema opuesto al del análisis. Aquí, se trata de determinar el circuito que realizará una función dada previamente. Usualmente esa función se presentará por su tabla de valores.

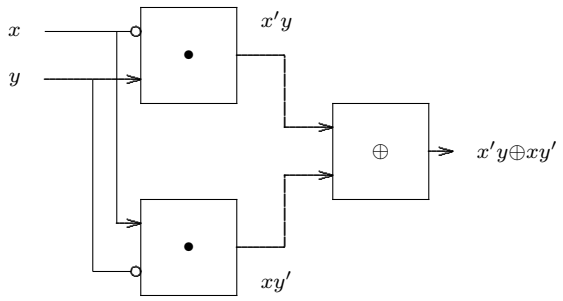

Figura 3: Un circuito lógico

Ejemplo 4.1. Suponer que queremos construir un circuito para realizar la función dada en la tabla siguiente:

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

Al examinar la tabla, nos damos cuenta que no se trata de una de las funciones básicas. Trataremos, por lo tanto, primeramente de expresar esa función en términos de nuestras expresiones básicas. Observando que la tabla contiene un único 1, nos recordamos de la tabla del AND. Ahora bien, un AND tiene un 1 solamente cuando ambos componentes son iguales a 1. En nuestro caso, esto lo podemos obtener negando x (para obtener un 1 en la primera columna y multiplicando por y. Verifiquemos: $x'y$ será 1 cuando, y sólo cuando, $x' = 1$, $y = 1$. O sea, cuando y sólo cuando, $x = 0$, $y = 1$. El circuito ilustrado en la figura 4 muestra una implantación de lo anterior.

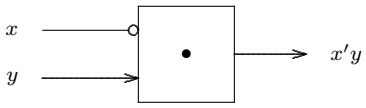

Figura 4: Realización de $f(x,y) = x'y$

Ejemplo 4.2.

x	y	$g(x,y)$
0	0	0
0	1	0
1	0	1
1	1	0

La construcción del correspondiente circuito lo dejaremos al cuidado del lector.

Ejemplo 4.3.

x	y	$h(x,y)$
0	0	0
0	1	1
1	0	1
1	1	0

En este caso, tenemos dos 1 en la tabla de valores de la función. Las funciones para cada uno de esos 1 aislados, se hallaron en los ejemplos anteriores. Observando que esta función h tienen un 1, donde lo tiene la función f o la función g, concluimos que $h = f$ OR $g = x'y \oplus xy'$. El circuito correspondiente apareció en la figura 3.

Tenemos, así, un método para asociar a cada tabla una función booleana. Tales expresiones se convierten entonces en circuitos.

El procedimiento es el siguiente:

1. Asignar, por cada 1 de la tabla de valores, un monomio formado por el producto de todas las variables o sus negaciones. Se usará la variable cuando en la columna correspondiente aparezca un 1; en caso contrario, usaremos su negación.

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

nos producirá el monomio $xy'zuv'$.

2. La expresión final se obtiene sumando todos los monomios anteriores.

En tales expresiones, cada suma, multiplicación o negación representa una ficha. La simplificación algebraica de las expresiones es, entonces, una simplificación de la construcción; lo que producirá un abaratamiento de los costos.

Ejemplos 4.4. 4

4.4.1 Se verifica que $xy \oplus xz = x(y \oplus z)$, lo que nos convierte una expresión con dos AND y un OR en una expresión con único AND y un OR.

4.4.2 Se verifica que $x \oplus x = x$, lo que nos permite eliminar una ficha.

4.4.3 Se verifica que $x \oplus xy = x$, lo que reduce de dos fichas a cero fichas.

4.4.4 Finalmente, observemos que $x'y + xy' = x$ XOR y, lo que reduce cinco piezas a una sola.

5. Los Sumadores

Una de las aplicaciones más conocidas de las computadoras es como calculadoras. En esta sección, veremos como combinar l circuitos lógicos para construir una pequeña sumadora.

5.1. Los números binarios

¿Cómo representar los números dentro de las máquinas? Nosotros, los humanos, usamos una representación decimal para los números; es decir, diez dígitos para representar los números. Pero, las máquinas lógicas no disponen de tantos dedos; tan sólo de 0 y 1. Por lo tanto, usaremos una representación binaria para los números. Una representación binaria utiliza exactamente dos dígitos para representar los números. A continuación, mostramos una tabla de la conversión de decimales a binarios, para los primeros números.

Decimal	Binario
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
⋮	⋮

5.2. Sumador de un Bit

El primer sumador que diseñaremos nos permitirá sumar dos números binarios de un dígito cada uno. Usualmente, llamamos BIT a un dígito binario; por lo tanto, lo que queremos diseñar es una sumadora de un BIT. No será la mayor calculadora del mundo, pero ciertamente es un comienzo.
En este contexto, las combinaciones elementales están dadas por:

$$\begin{array}{r} 0 \\ +0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0 \\ +1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +1 \\ \hline 10 \end{array}$$

El resultado de $1 + 1$, requiere dos dígitos para ser expresado. Por lo tanto nuestro sumador, para ser completo, deberá tener dos señales de entrada (una para cada dígito) y dos señales de salida. Reescribiendo las sumas con dos dígitos de salida, tendremos que:

$$\begin{array}{r} 0 \\ +0 \\ \hline 00 \end{array} \qquad \begin{array}{r} 0 \\ +1 \\ \hline 01 \end{array} \qquad \begin{array}{r} 1 \\ +0 \\ \hline 01 \end{array} \qquad \begin{array}{r} 1 \\ +1 \\ \hline 10 \end{array}$$

Llamamos al dígito de la izquierda de la suma, "acarreo" (en inglés, "carry") y al otro, "resultado". Con esta notación, nuestra máquina sumadora luce esquemáticamente como en la figura 5.

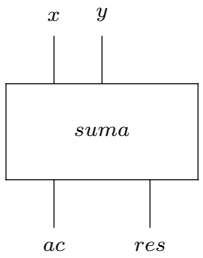

Figura 5: Esquema de una sumadora

Construyamos la tabla de valores para las señales de salida deseadas:

x	y	acarreo	resultado
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Observando las tablas, vemos que para el acarreo se cumple que $c = xy$ lo que puede ser producido por una AND. Para el resultado, r, tenemos que $r = x'y \oplus x'y = xXORy$.

5.3. Sumadores de más de un Bit

Procederemos ahora al diseño de sumadores de números de más de un dígito. Sumar números con más de un dígito, agrega una dificultad no encontrada en el ejemplo anterior, la posible presencia de "acarreo" desde la columna anterior. Por ejemplo,

$$\begin{array}{cccccc} & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & & \\ & 1 & 0 & 1 & 1 \\ + & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 0 & 1 & 0 \end{array}$$

Diseñaremos, pues, un nuevo sumador de un dígito, que tendrá tres señales de entrada: los dos dígitos a sumar y un acarreo anterior. Las señales de salida serán: acarreo (nuevo) y resultado. Usando varios de estos sumadores (llamados sumadores totales) podremos sumar números de varios dígitos. Veamos un ejemplo, con números tres dígitos.
Veamos ahora la tabla de valores para el sumador total:

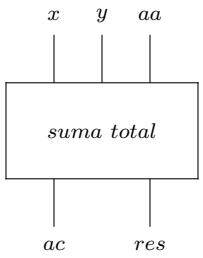

Figura 6: Sumador total

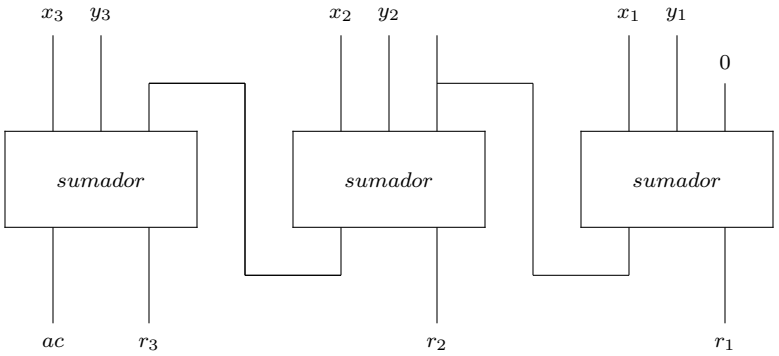

Figura 7: Sumadora de tres dígitos

acarreo anterior	x	y	acarreo nuevo	resultado
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Tenemos, usando nuestro método para ir de una tabla a una función, que:

$$\text{Acarreo nuevo} = c'xy \oplus cx'y \oplus cxy' \oplus cxy = xy + c(x+y)$$
$$\text{Resultado} = c'x'y \oplus c'xy' \oplus cx'y' \oplus cxy = c + x + y$$

donde c es el acarreo de la ficha anterior. El primer acarreo es igual a 0. Dejaremos al cuidado del lector la construcción del correspondiente circuito.