

Lógica Digital

Organización de computadoras 2018

Universidad Nacional de Quilmes

Introducción: Cálculo Proposicional

El cálculo proposicional es el estudio de las relaciones lógicas entre objetos llamados proposiciones, que generalmente pueden interpretarse como afirmaciones que tienen algún significado en contextos de la vida real. Para nosotros, una proposición será cualquier frase que pueda ser verdadera o falsa.

- *Está lloviendo* es una proposición
- *¿Cómo te sentís hoy?* no es una proposición
- *El mar y el sonido de las olas* no es una proposición
- *Yo estudio organización de computadoras* es una proposición

En el cálculo proposicional se utilizan letras minúsculas (ej: p, q, r) para simbolizar proposiciones, que se pueden combinar utilizando conectivos lógicos:

1. \neg para “no” o negación
2. \bullet Para “y”
3. $+$ para “o”
4. \Rightarrow para implicación condicional

Veamos el siguiente ejemplo con las proposiciones p, q y r definidas de la siguiente manera:

- p = “está lloviendo”
- q = “el sol está brillando”
- r = “hay nubes en el cielo”

Simbolizamos las siguientes frases:

1. Está lloviendo y el sol está brillando: $p \bullet q$

2. Si está lloviendo, entonces hay nubes en el cielo
 $p \Rightarrow r$

3. Si no está lloviendo, entonces el sol no está brillando y hay nubes en el cielo $\bar{p} \Rightarrow (\bar{q} \bullet r)$

La suposición fundamental del cálculo proposicional consiste en que los valores de verdad de una proposición construida a partir de otras proposiciones quedan completamente determinados por los valores de verdad de las proposiciones originales. Para ello se establecen los valores de verdad según las posibles combinaciones de valores de verdad de las proposiciones originales, basándonos en las siguientes tablas, considerar 0 como Falso y 1 como verdadero:

A	B	$A \bullet B$
0	0	0
0	1	0
1	0	0
1	1	1

La tabla indica que, el conectivo lógico “y” sólo será verdadero cuando ambas proposiciones p y q sean verdaderas.

A	B	$A + B$
0	0	0
0	1	1
1	0	1
1	1	1

La tabla indica que, si al menos una de las proposiciones es verdadera, la proposición formada por el conectivo “o” será verdadera.

A	\bar{A}
0	1
1	0

Evidentemente, si la proposición p es verdadera, su negación será falsa y viceversa.

Operadores elementales

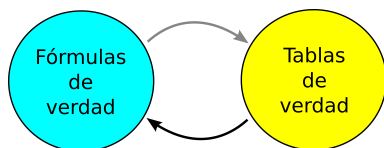
El álgebra de Boole, como cualquier álgebra, usa variables y operaciones. En particular, las variables y operaciones son lógicas, por lo cual las variables pueden tomar el valor 1 (que representa al valor verdadero) y el 0 (que representa al valor falso). A continuación un repaso de las tablas de verdad de los operadores clásicos.

	p	q	$p \bullet q$
Conjunción	0	0	0
	0	1	0
	1	0	0
	1	1	1

	p	q	$p + q$
Disyunción	0	0	0
	0	1	1
	1	0	1
	1	1	1

	p	\bar{p}
Negación	0	1
	1	0

De fórmula lógica a tabla de verdad



Una fórmula lógica se puede traducir a una tabla de verdad para obtener como resultado su valor de verdad. A continuación se mostrara como se realiza la traducción.

Consideremos la siguiente formula:

a) $(p \bullet q) + p$

y tomemos en cuenta lo siguiente:

- la cantidad de variables dentro de la fórmula nos va a indicar la cantidad de filas que va a tener nuestra tabla. En este caso tenemos 2 variables p y q , por lo tanto vamos a tener 4 filas: $2^{cant.variables} = 2^2$.

- las operaciones lógicas que componen la fórmula. En este caso tenemos las operaciones **and** y **or**. Es importante tener en mente la tabla de verdad de cada operación, para poder completar la tabla.

Para armar la tabla hay que seguir un orden, primero se empieza por los valores de verdad de la variables lógicas y luego se calculan el valor de verdad de cada termino de la formula:

Tabla de verdad :

p	q	$p \bullet q$	$(p \bullet q) + p$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

El valor de verdad de dicha fórmula se refleja en la última columna. Si no se dieron cuenta, en la tabla se hace diferencia entre los valores de las variables (que son fijos) y los valores de cada termino de la fórmula (que se calculan).

Formas normales

A menudo es útil contar con un mecanismo estructurado para deducir la fórmula de verdad a partir de una tabla de verdad de una relativa complejidad. Para eso presentamos la **suma de productos (SoP)** y el **producto de sumas (PoS)**.

SoP La expresión SOP (*Sum of product*) es una norma de escritura para las fórmula de verdad. Como lo dice su nombre, es una suma (disyunción) de términos que son productos (conjunciones) entre literales. ¿Y que es un literal? Bueno, es una variable o su negación. Para construirla, se debe extraer el término que describe cada caso de la tabla de verdad que verifica la fórmula. Es decir, cada fila donde la salida vale 1. Con cada uno de esos casos se describe con un término con todas las variables, según aparezcan afirmadas o negadas. Por ejemplo

p	q	s
0	0	0
0	1	1 $\rightarrow \bar{p} \bullet q$
1	0	1 $\rightarrow p \bullet \bar{q}$
1	1	1 $\rightarrow p \bullet q$

Finalmente, los términos se componen con una disyunción: $(\bar{p} \bullet q) + (p \bullet \bar{q}) + (p \bullet q)$

PoS La expresión PoS (*Products of sums*) es una conjunción de disyunciones. A diferencia de la anterior, describe la fórmula a partir de los casos donde no se cumple, algo así como **”f vale cuando no ocurre el caso ... ni el caso...”** El primer paso es rescatar los casos donde la fórmula vale **0**, por ejemplo:

p	q	s
0	0	0 $\rightarrow \bar{p} \bullet \bar{q}$
0	1	1
1	0	0 $\rightarrow p \bullet \bar{q}$
1	1	1

Luego, esos casos se niegan y se unen con conjunción, ya que no debe cumplirse ninguno de ellos: $(\bar{p} \bullet \bar{q}) \bullet (p \bullet \bar{q})$ Finalmente, aplicando la propiedad de De Morgan, los terminos negados se convierten en disyunciones: $(p + q) \bullet (\bar{p} + \bar{q})$

Relacionemos este tema con la Informática:

Desde hace mucho tiempo, el hombre en su vida diaria se expresa, comunica, almacena información, la manipula, etc. mediante letras y números. Para la representación numérica utiliza el sistema de representación decimal, en tanto que, dependiendo del idioma, dispone de un alfabeto que representa estas letras. Siguiendo el mismo principio que guía al hombre, las computadoras tienen su propio sistema de representación. Debido a su construcción basada fundamentalmente en circuitos electrónicos digitales, utiliza un sistema binario. Esto obliga a transformar la representación de nuestra información, tanto numérica como alfanumérica, a una representación binaria para que la máquina sea capaz de procesarlos.

Por cuestiones de índole técnica, los circuitos electrónicos que conforman una computadora suelen estar capacitados para reconocer señales eléctricas de tipo digital; por lo tanto, se hace necesario que los métodos de codificación internos tengan su origen en el sistema binario, y con ellos se pueda representar todo tipo de informaciones y órdenes que sean manejadas por la computadora. En los circuitos electrónicos suele representarse la presencia de tensión (electricidad) en un punto de un circuito

por medio de un 1, en tanto que 0 representa la ausencia de dicha tensión.

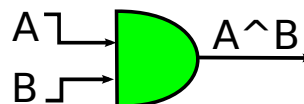
En consecuencia utilizaremos lógica proposicional para resolver problemas y lógica digital para automatizar soluciones de problemas.

La electrónica digital está fundamentada en la base matemática formada por el álgebra de Boole (George Boole, matemático inglés, 1815 -1864). Este método de análisis considera que todos los elementos poseen únicamente dos estados (biestables) o dos valores, verdadero o falso (1 ó 0) que son opuestos entre sí, no permitiéndose nunca la adopción de estados intermedios. Estudiando las distintas asociaciones entre ellos se obtienen las leyes generales sobre los procesos lógicos. Fue Claude Shannon (matemático e ingeniero norteamericano, 1916 -2001) quien aplicó estas técnicas de estudio, a los circuitos compuestos de elementos que solo pueden adoptar dos estados estables, apareciendo entonces los llamados circuitos lógicos. Puede decirse entonces que el álgebra de Boole es el sistema matemático empleado en el diseño de circuitos lógicos, que nos permite identificar mediante símbolos el objeto de un circuito lógico de modo que su estado sea equivalente a un circuito real.

Un circuito lógico es una composición de compuertas que traduce un conjunto de entradas en una salida, de acuerdo a una función lógica, que se actualiza inmediatamente luego de proveerse las entradas.

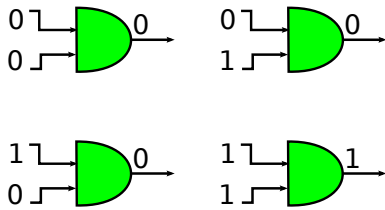
Existe un convenio gráfico para representar dispositivos (electrónicos, hidráulicos, mecánicos, etc.) que lleven a cabo funciones booleanas elementales y que, en función de la combinación o combinaciones diseñadas, se obtendrán funciones más complejas. Las compuertas lógicas son dispositivos electrónicos que desarrollan las funciones booleanas

Conjunción - AND

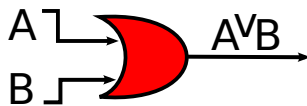


Una compuerta AND tiene dos entradas como mínimo y su operación lógica es un producto entre ambas, no es un producto aritmético, aunque en este caso coincidan. *Observa que su salida será 1 si sus dos entradas están a nivel alto (1)*

Casos:

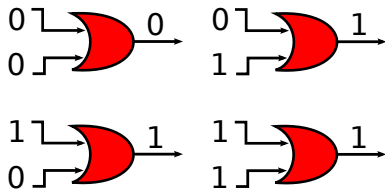


Disyunción: OR

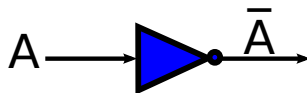


Al igual que la anterior posee dos entradas como mínimo y la operación lógica, será una suma entre ambas. Bueno, todo va bien hasta que $1 + 1 = 1$, el tema es que se trata de una compuerta O Inclusiva es como a y/o b. *Es decir, basta que una de ellas sea 1 para que su salida sea también 1*

Casos:

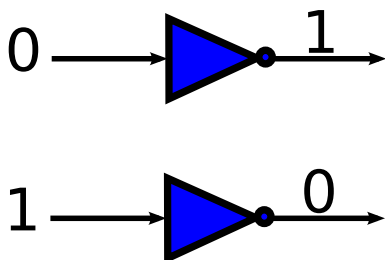


Negación: NOT



Se trata de un inversor, es decir, invierte el dato de entrada, por ejemplo; si se pone su entrada a 1 (nivel alto) se obtiene en su salida un 0 (o nivel bajo), y viceversa. Esta compuerta dispone de una sola entrada. *Su operación lógica es igual a a invertida*

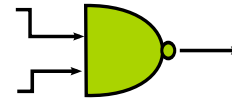
Casos:



Operaciones lógicas adicionales

1. Compuerta NAND

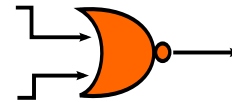
$$a \uparrow b = \overline{a \cdot b}$$



Responde a la inversión del producto lógico de sus entradas, en su representación simbólica se reemplaza la compuerta NOT por un círculo a la salida de la compuerta AND.

2. Compuerta NOR

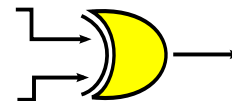
$$a \downarrow b = \overline{a + b}$$



El resultado que se obtiene a la salida de esta compuerta resulta de la inversión de la operación lógica o inclusiva es como un no a y/o b. Igual que antes, solo agregas un círculo a la compuerta OR y ya tienes una NOR.

3. Compuerta XOR

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

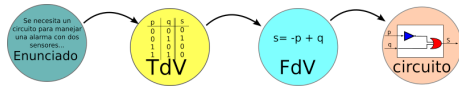


Es OR EXclusiva en este caso con dos entradas (puede tener más) y lo que hará con ellas será una suma lógica entre a por b invertida y a invertida por b.*Al ser O Exclusiva su salida será 1 si una y sólo una de sus entradas es 1*

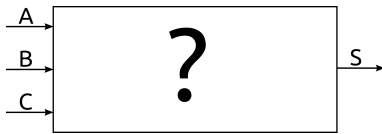
Diseño de circuitos

¿Cómo se construyen los circuitos? A partir de su fórmula de verdad ¿Y como se llega a la fórmula?

En general se cuenta con una descripción del problema a través de un enunciado en lenguaje natural (informal), que debe ser abstraído (formalizado) en una tabla de verdad, para luego ser convertida en una fórmula de verdad y luego en un circuito.



Por ejemplo, se necesita un circuito de 3 entradas y una salida que compute la función "mayoría". Es decir, que si dos o más entradas valen 1, la salida debe valer 1, y un 0 en caso contrario. Para entender lo que se pide, es útil dibujarlo como una caja negra:



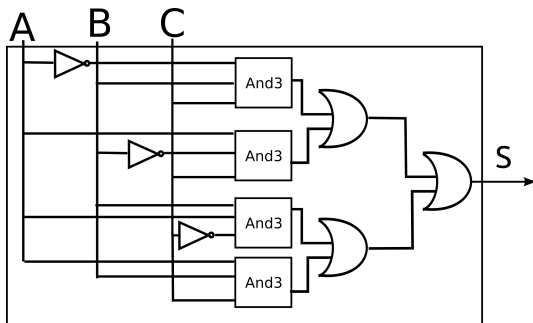
Y luego plantear la tabla de verdad:

A	B	C	Mayoría
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Usando el método SoP, se obtiene la fórmula:

$$(\bar{A} \bullet B \bullet C) + (A \bullet \bar{B} \bullet C) + (A \bullet B \bullet \bar{C}) + (A \bullet B \bullet C)$$

Finalmente, dibujar el circuito:



Actividad

1. Dibuja el circuito lógico para la siguiente fórmula $(A \bullet B)$

2. Dibuja el circuito lógico para la siguiente fórmula $((A \bullet B) + (\bar{A} \bullet C))$

3. ¿Podrías armar la tabla de verdad correspondiente al item anterior?