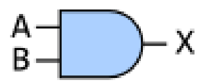
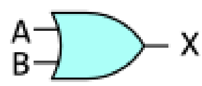
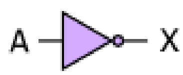
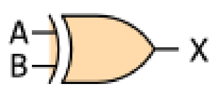
Boolean logic

In the 1840s, English mathematician George Boole developed an algebra (a set of operators and laws) for variables that can have just two states – **true** and **false.** Thus, a Boolean value is equivalent to one bit:

```
False = 0 = off
True  = 1 = on
```

The operators defined by Boole are pervasive throughout all of computing. You may have encountered them in doing library or other database searches. The ones we'll consider are:

AND	X=A·B	
OR	X=A+B	
NOT	X=A'	
XOR	X=A⊕B	

The table illustrates both the algebraic notation and the **circuit diagram** notation. The elements of circuit diagrams are called **gates,** as in "AND gate" or "XOR gate."

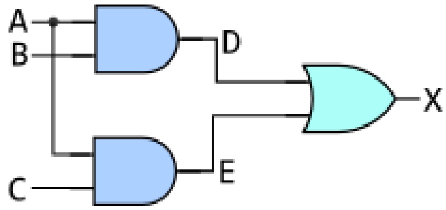
The behavior of these operators can be defined by **truth tables:**

A	B	A·B	A+B	A'	A⊕B
0	0	0	0	1	0
0	1	0	1	1	1
1	0	0	1	0	1
1	1	1	1	0	0

We combine the gates into **combinational circuits** to achieve various effects. For example, the algebraic expression `X = A·B + A·C` corresponds precisely to the following circuit diagram:

and we can discover its effect by completing the truth table:

A	B	C	D=A·B	E=A·C	X=D+E
0	0	0	0	0	0
0	0	1	0	0	0



0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

Exercise: Try drawing the circuits and the truth tables for `X=(A·B)'` and for `X=A'+B'`. They should produce the same result for the inputs `A` and `B`. This is one of **DeMorgan's Laws.**

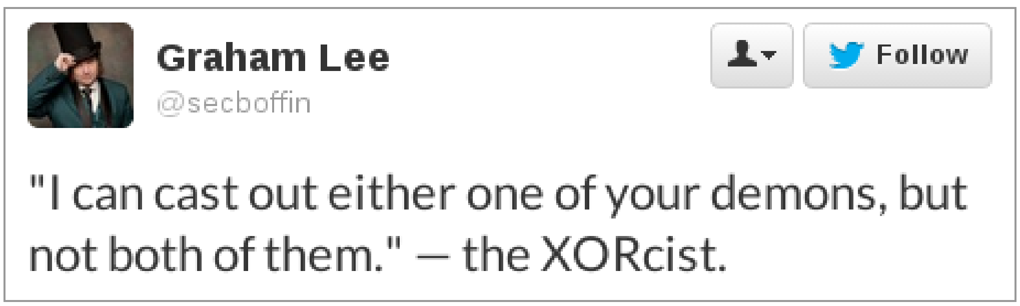


Figure 1: @secboffin on Twitter