From Monoids chapter of Functional Programming in Scala

"If you have your law-discovering cap on while reading this chapter, you may notice that there's a law that holds for some functions *between* monoids"

Monoid: (false, ||)

Zero: false
Associative operation: ||
Identity: (false || p) == (p || false) == p
Associativity: (p || q) || r == p || (q || r)

Monoid: (true, &&)

Zero: true
Associative operation: &&
Identity: (true && p) == (p && true) == p
Associativity: ((p && q) && r) == (p && (q && r))

A **monoid homomorphism** f between monoids M and N obeys the following general law for all values x and y:

```
M.op( f( x ), f( y ) ) == f( N.op( x, y ) )
```

If we choose M = (false, ||); N = (true, &&); f = !

then we have these **monoid homomorphisms**:

```
M.op( f( x ), f( y ) ) == f( N.op( x, y ) )
      ! p || ! q == ! ( p && q )

N.op( f( x ), f( y ) ) == f( M.op( x, y ) )
      ! p && ! q == ! ( p || q )
```

De Morgan's laws

In formal language, the rules are written as

$$\neg(P \wedge Q) \iff (\neg P) \vee (\neg Q),$$

and

$$\neg(P \vee Q) \iff (\neg P) \wedge (\neg Q),$$

where

- *P* and *Q are propositions,*
- ¬ is the negation logic operator (NOT),
- ∧ is the conjunction logic operator (AND),
- ∨ is the disjunction logic operator (OR),
- ⟺ is a metalogical symbol meaning "can be replaced in a logical proof with".

Augustus De Morgan

Augustus De Morgan (1806-1871)