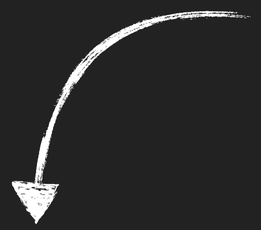
Since the laws are the same for every state-based CRDT, I can reuse my tests

```
spec =
  describe "State-based CRDTs" do
    stateBasedCRDTLaws "GSet of integers" (Proxy :: Proxy (GSet Int))
    stateBasedCRDTLaws "GSet of strings" (Proxy :: Proxy (GSet String))
    stateBasedCRDTLaws "GCounter" (Proxy :: Proxy GCounter)
    stateBasedCRDTLaws "2PSet of integers" (Proxy :: Proxy (TwoPhaseSet Int))
    stateBasedCRDTLaws "2PSet of strings" (Proxy :: Proxy (TwoPhaseSet String))
    -- and so on
```

```
❯ spago test
GSet - Grow-only set » Int laws
  √ should be associative
  √ should be commutative
  √ should be idempotent
  √ should have a neutral element
GSet - Grow-only set » String laws
  √ should be associative
  √ should be commutative
  √ should be idempotent
  √ should have a neutral element
2PSet - Two-phase Set » Int laws
  √ should be associative
  √ should be commutative
  √ should be idempotent
  √ should have a neutral element
2PSet - Two-phase Set » String laws
  √ should be associative
  √ should be commutative
  √ should be idempotent
  √ should have a neutral element
GCounter - Grow-only counter » laws
  √ should be associative
  √ should be commutative
  √ should be idempotent
  √ should have a neutral element
OPCounter - Operation-based counter » laws
  √ should be commutative
  √ should have complementary generateOperations / applyOperation functions
```

A single test run generates hundreds of test cases, giving us a reasonable level of confidence that our implementation is correct



Operation-based CRDTs can be tested the same way