```haskell
stateBasedCRDTLaws ::
  forall t. StateBasedCRDT t ⇒ Arbitrary t ⇒ String → Eq t ⇒ Show t ⇒ Proxy t → Spec Unit
stateBasedCRDTLaws name _ =
  describe name do
    it "should be associative"
      $ quickCheck associativity
    it "should be commutative" do
      quickCheck commutativity
    it "should be idempotent" do
      quickCheck idempotence
    it "should have a neutral element" do
      quickCheck identity'
  where
  associativity a b c = (merge (merge a b) c) ≡≡ (merge a (merge b c))

  commutativity a b = merge a b ≡≡ merge b a

  idempotence a = merge a a ≡≡ a

  identity' a = (merge a mempty) ≡≡ a
```

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
```