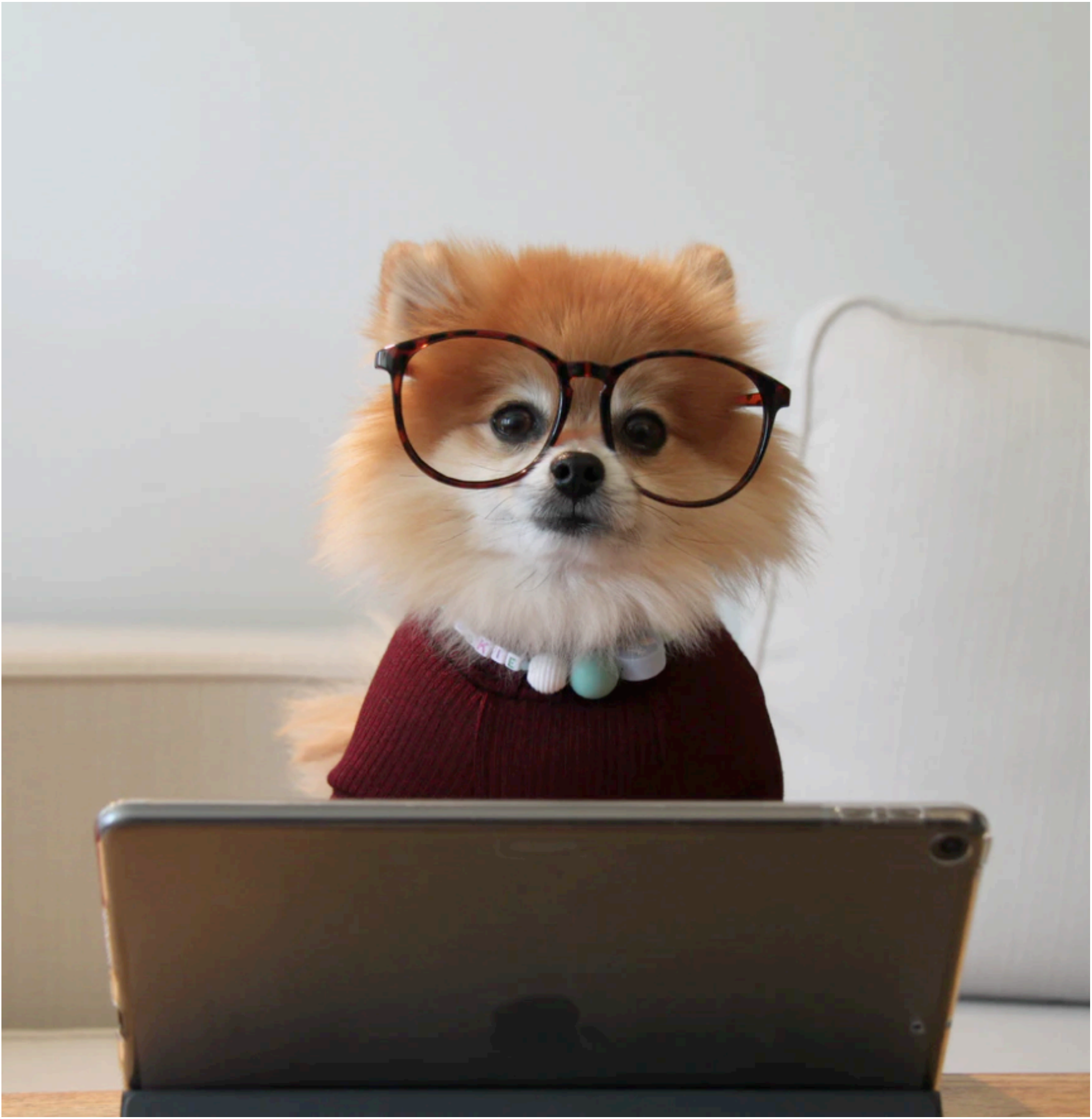
- If I have a way of generating arbitrary members of $\mathbb{S}$ …

- A merge(a, b) function

- And a law, for instance commutativity, defined as

$$\forall x \in S, y \in S : merge(x, y) = merge(y, x)$$

- Then I could generate a large number of inputs, and make sure the law holds of every one of them



- If I have a way of generating arbitrary members of S …
- A $\mathrm{merge(a,\ b)}$ function
- And a law, for instance commutativity, defined as
  $\forall x \in S, y \in S : merge(x, y) = merge(y, x)$
- Then I could generate a large number of inputs, and make sure the law holds of every one of them

Introducing QuickCheck

```haskell
-- In QuickCheck, the Arbitrary class associates a random value generator to a type
class Arbitrary t where
  arbitrary :: Gen t -- Gen t is a way of generating random values of type t


-- Let's define Arbitrary instances of our CRDTs
instance arbitraryGSet :: (Arbitrary t, Hashable t) ⇒ Arbitrary (GSet t) where
  arbitrary = GSet <<< Set.fromArray <$> arbitrary

instance gCounterArbitrary :: Arbitrary GCounter where
  arbitrary = GCounter <<< Map.fromArray <$> arrayOf kvGen
    where
    kvGen :: Gen (Tuple ReplicaId Int)
    kvGen = Tuple <$> arbitrary <*> suchThat arbitrary (\x → greaterThanOrEq x 0
```