1) instance Arbitrary Type where ...

trabalhar com 3 vars de tipo ("α", "β", "γ") *
3 construtores (vars de programa) "A", "B", "C"
* talvez 8 com 2 para cada kind

devem ter de
gerar rapidamente um tipo de sessão arbitrário
para compor com ∘ ; etc

2)

```
data EquivPair = EquivPair Type Type

instance Arbitrary EquivPair where
  arbitrary = do
    n <- choose (1, 9) :: Gen Int
    case n of
      1 -> do        -- assoc
        (t, u, v) <- arbitrary
        return $ Pair ((t ; u); v)  (t; (u ; v))
      2 -> do        -- distrib.
        t <- arbitrary :: Type
        m <- arbitrary :: FieldMap
        p <- arbitrary :: Polarity
        return $ Pair (t; Choice defaultPos p m)
          ( Choice defaultPos p (map .. m t ) }

      3 -> do        -- rec unfold
        t <- arbitrary :: Type
        x <- arbitrary :: TypeVariable
        return $ Pair (μ x. t) ( t [μx.t/x] )
```

Scanned with CamScanner

```
4 -> do
    t <- arbitrary
    (x,y)<-arbitrary
    return $ Pair (rec x. rec y. t) (rec x . t[x/y])

5 ->  do            -- δ not free ∈ t
    t <- arbitrary         esta não pode ser gerado
                           por arbitrary
    return $ ( μ"δ".t, t)

6 ->  do            -- skip neutral left
      t <- arbitrary
      return $ Pair  t ( skip;t)

7 -> do            -- skip neutral right
      t <- arbitrary
      return $ Pair  t ( ~~skip~~t;skip)

8 ->  do
      (Pair t u) <- arbitrary :: EquivPair
              v <- arbitrary
              x <- arbitrary
      return $ Pair (v[t/x], v[u/x])

9 -> do
      t <- arbitrary
      return $ Pair t t
```