Practical Verification of QuadTrees

Jonathan Brouwer (jtbrouwer@student.tudelft.nl) Delft University of Technology

1. Introduction

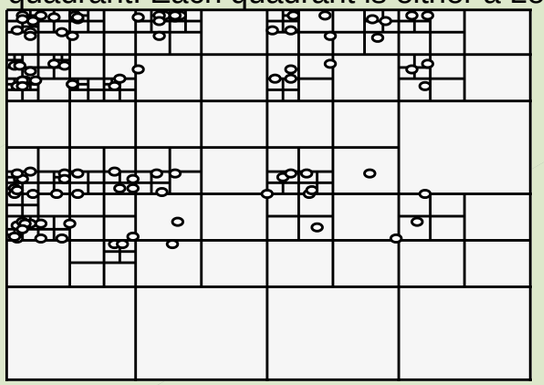
QuadTree is a Haskell library. This paper aims to rewrite this in Agda, so it can be formally verified using Curry Howard. It can then be compiled back to Haskell using Agda2hs.

```
takesGtFive : (n : Nat)
            -> IsTrue (n > 5) -> ?
```

Can agda2hs be used to produce a verified implementation of the QuadTree library?

2. QuadTrees

QuadTrees are used for storing two-dimensional information in a functional style. They consist of a size and a root quadrant. Each quadrant is either a Leaf, or a



3. Invariants

Invariants are proven by adding the proof as an implicit constructor argument. To verify that a quadrant is compressed (no identical leafs) and has a certain depth, we can use:

3. Invariants

Invariants are proven by adding the proof as an implicit constructor argument. To verify that a quadrant is compressed (no identical leafs) and has a certain depth, we can use:

```
data VQuadrant (t : Set) {dep : Nat} : Set where
  CVQuadrant : (qd : Quadrant t)
             -> {.(IsTrue (depth qd <= dep && isCompressed qd))}
             -> VQuadrant t {dep}
```

4. Preconditions

Preconditions are proven by adding the proofs as implicit arguments to the function. To verify that the location given to getLocation is in the QuadTree, one can use:

```
getLocation : (loc : Nat × Nat) -> {dep : Nat}
    -> (qt : QuadTree t)
    -> {.( IsTrue (isInsideQuadTree loc qt) )} -> t
```

Alternatively, we can pass in a datatype with an invariant. This verifies that the input of lensLeaf is depth 0 and that the depth of the input of lensA is greater than 0.

```
lensLeaf : Lens (VQuadrant t {0}) t
lensA : {dep : Nat}
    -> Lens (VQuadrant t {S dep}) (VQuadrant t {dep})
```

5. Postconditions

Postconditions a example, this is the proof that one of the lens laws holds for lensLeaf.

```
ValidLens-Leaf-ViewSet :
    -> (v : t) (s : VQuadrant t {0})
    -> view (lensLeaf {t}) (set (lensLeaf {t}) v s) ≡ v
ValidLens-Leaf-ViewSet v (CVQuadrant (Leaf x)) = refl
```