# Practical Verification of the Reader Monad

*By Alex Harsani*
*Under supervision of Jesper Cockx & Lucas Escot*
TU Delft

## 1. Background Information

- **Haskell** is a pure functional language.
- **Agda** is a dependently typed language and proof assistant.
- **Agda2hs** is a tool that provides verified translation from Agda to Haskell.
- **Reader** is a monad that is used to model a global state.
- **ReaderT** is a monad transformer that can be used to combine Reader with other monads.

```
circleArea :: ReaderT Double IO ()
circleArea = do
  r <- lift $ getLine
  pi <- ask
  lift $ putStrLn (show (pi * (read r) * (read r)))
```

## 2. Research Questions

*Can agda2hs be used to produce verified implementations of Reader and ReaderT?*

- Can we implement Reader and ReaderT in Agda using the language subset defined by agda2hs?
- What are the properties that need to be satisfied by Reader and ReaderT and how do we prove these properties?
- Does agda2hs provide correct and useful translation of Reader and ReaderT to Haskell?

## 3. Implementation

- *Reader* and *ReaderT* are defined as record types:

```
record Reader (r a : Set) : Set where
    constructor MkReader
    field
        readerComputation : (r → a)
```

```
record ReaderT (r : Set) (m : Set -> Set) (a : Set) : Set where
    constructor MkReaderT
    field
        readerTComputation : (r → m a)
```

- We also define functions ask, asks, local and runReader.
- MonadTrans type class:

```
record MonadTrans (t : (Set → Set) → Set → Set) {{ @0 iT : ∀ {m}
    -> {{Monad m}} -> Monad (t m)}} : Set₁ where
    field
        lift : {{Monad m}} -> {@0 a : Set} →  m a → t m a
```

## 4. Verification

- *Functor*, *Applicative* and *Monad* laws
- *MonadTrans* laws
- To verify these laws we create instance of the verified type class with proof functions.

```
record VerifiedFunctor (f : Set → Set) {{@0 iF : Functor f}} : Set₁ where
    field
        @0 f-id-law : {a : Set} (x : f a) → fmap id x ≡ x
        @0 f-composition-law : {A B C : Set} (g : B → C) → (h : A → B)
            → (x : f A) → fmap (g ∘ h) x ≡ (fmap g ∘ fmap h) x
```

## 5. Results

- Verified implementation of Reader and ReaderT was successfully produced by agda2hs.

```
record Reader (r a : Set) : Set where
    constructor MkReader
    field
        readerComputation : (r → a)
```

↓

```
data Reader r a = MkReader{readerComputation :: r -> a}
```

- Complete implementation with proofs, as well as the demo can be found at:

https://github.com/AlexHarsani/monad-verification/releases/tag/paper

## 6. Limitations

- *newtype* - agda2hs cannot translate newtype definitions
- *Quantified constraint* - translating to Haskell

## 7. Conclusions and Future Work

- Reader and ReaderT were successfully implemented and verified.
- *Future Work - MonadReader class, Identity monad*

Contact: A.Harsani@student.tudelft.nl
Supervisors Contact: J.G.H.Cockx@tudelft.nl, L.F.B.Escot@tudelft.nl