Practical Verification of Ranged-sets

CSE3000 Research Project by Ioana Savu *under the supervision of Jesper Cockx, Lucas Escot*

a.i.savu-1@student.tudelft.nl, J.G.H.Cockx@tudelft.nl, L.F.B.Escot@tudelft.nl

TUDelft

01 Background

- **Haskell** is a pure, but partial functional language
- **Agda** is a total and dependently-typed language and can be used to prove properties of programs, we call this **verifying**
- **agda2hs** aims to convert Agda to Haskell code
- **Ranged-sets** library allows programming with sets of values described as lists of ranges

02 Problem

Can we reproduce a verified implementation of the **Ranged-sets Haskell library** in **Agda** using **agda2hs**?

03 Method

- Add to agda2hs the missing **types** needed by the library
- Port Ranged-sets to Agda & check if the **partial** functions can become **total**
- **Prove** the **properties** of the library

Proving properties

Preconditions

- Embedded as **instance** arguments

```
unsafeRangedSet : ⦃ o : Ord a ⦄ → ⦃ dio : DiscreteOrdered a ⦄
   → (rg : List (Range a))
   → ⦃ IsTrue (validRangeList rg) ⦄ → RSet a
```

Invariants

- Embedded in the constructor as **implicit** argument

```
RS : (rg : List (Range a))
    → {IsTrue (validRangeList rg)} → RSet a
```

Property based testing (QuickCheck)

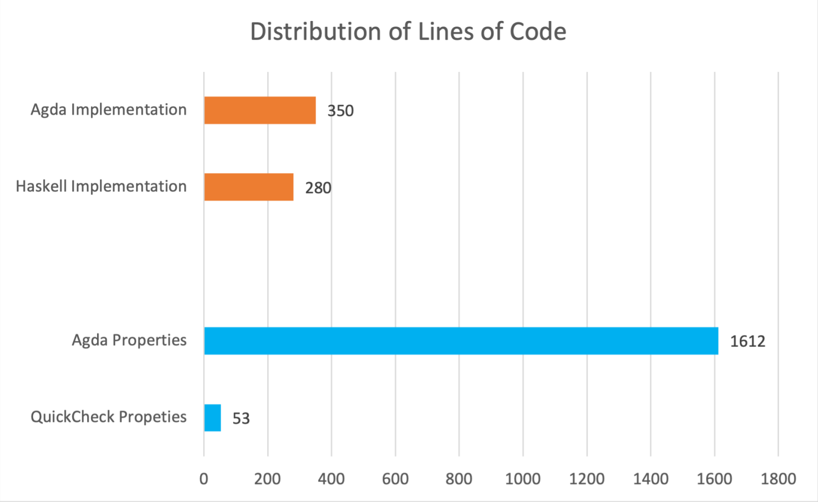
```
prop_union :: (DiscreteOrdered a ) => RSet a -> RSet a -> a -> Bool
prop_union rs1 rs2 v =
   (rs1 -?- v || rs2 -?- v) == ((rs1 -\/- rs2) -?- v)
```

↓ Agda translation of a QuickCheck property

```
prop_union : ⦃ o : Ord a ⦄ → ⦃ dio : DiscreteOrdered a ⦄
   → (rs1 rs2 : RSet a) → (v : a)
   → ((rs1 -\/- rs2) -?- v) ≡ (rs1 -?- v || rs2 -?- v)
```

04 Results

Preconditions & **invariants** are specified in the documentation of the library, but not **verified.** By verifying them, we ensure that the functions behave as expected.



05 Conclusion

- The **Ranged-sets** library can be translated and verified in Agda using **agda2hs**
- Further research is needed in order to simplify the **verification** process i.e., identifying tactics that work in similar situations