Proving with the Help of Property-Based Testing

Exposing the QuickCheck Interface in Agda for Agda2hs

TU Delft Delft University of Technology

1. Background

QuickCheck [1] is a Property-Based Testing (PBT) framework written in Haskell

- PBT: test expected properties of code with randomly generated inputs

Agda [2] is a total language used as a proof assistant

- All functions *must* return a value in their codomain:
  No infinite loops, no exceptions/errors
- Can be used to prove propositions using the Curry-Howard correspondence:
  Encode a proposition as a function type, the function implementation acts as a proof

Agda2hs [3] is used to compile Agda code to Haskell.

- Write general purpose Haskell code with the safety of Agda
- Currently does not have a testing framework

2. Problem

Proving can take a lot of time
Proving a false proposition is impossible, but it can still take a lot of time to realise a proposition is false

Test a proposition before proving, using QuickCheck!

3. Research Question

How can the interface of the Haskell QuickCheck library and its implicit properties be exposed in Agda for use with Agda2hs?

4. Porting QuickCheck

- Postulate QuickCheck's functions, such as `===`, in Agda:
  Declaring their existence instead of reimplementing them
  Possible since code is transpiled to the original QuickCheck library
- Implement QuickCheck's data types, such as `Gen`:
  Now users can instantiate and use them in their code

```
record Arbitrary (a : Type) : Type where
  -- (...)
  field
    arbitrary : Gen a
    shrink : a → List a

record Circle : Type where
  no-eta-equality; pattern
  constructor MkCircle
  field radius : Nat

instance
  iArbitraryCircle : Arbitrary Circle
  iArbitraryCircle .arbitrary = MkCircle <$> arbitrary
  iArbitraryCircle .shrink (MkCircle r) = MkCircle <$> (shrink r)

prop_circle_area : Circle → Property
prop_circle_area c = area (rescale 2 c) === 4 * (area c)
```

```
record Gen (a : Type) : Type where
  no-eta-equality; pattern
  constructor MkGen
  field unGen : QCGen → ∃ Int IsNonNegativeInt → a

record GenN (a : Type) : Type where
  -- (...)
  field unGenN : QCGen → Nat → a
```

Figure 1: Implementing the `Arbitrary` class on a user-defined `Circle` type. Includes the two definitions of a generator.

- Implement QuickCheck's classes, such as `Arbitrary`:
  Allow users to create new instances
- Make **implicit properties** of functions **explicit**:
  Make sure function calls resulting in thrown exceptions cannot be made
  Allow for reasoning with postconditions of functions
- Add **correspondence proofs** to the port:
  Let users prove that a test asserts the same concept as a proposition

5. Evaluation

- **Most important functionality** has been made **available** in Agda2hs:
  Half of QuickCheck's API has been ported, prioritised by importance
- Missing: Surjectivity proofs of generators
  A generator with blind spots weakens the guarantees of a test
  This is especially relevant for correspondences proofs

```
@0 _corresponds-to_ : (a → Type) → (a → Bool) → Type
_corresponds-to_ P t = ∀ x → Reflects (P x) (t x)

module _ {ty : Type} {{_ : Eq ty}} {{_ : IsLawfulEq ty}}
                                                  where
  ==-to-≡ : ∀ {x y : ty} → IsTrue (x == y) → x ≡ y
  ≡-to-== : ∀ {x y : ty} → x ≡ y → IsTrue (x == y)
```

Figure 2: Definitions of some of the functions used for correspondence proofs.

```
proposition : List a → Type
proposition xs = xs ++ [] ≡ xs

postulate
  proof : ∀ (xs : List a) → proposition xs

prop_append_empty : List Int → Bool
prop_append_empty xs = xs ++ [] == xs

corresponds : proposition corresponds-to prop_append_empty
corresponds xs = mapReflects ==-to-≡ ≡-to-==
                             (reflectsIsTrue (prop_append xs))
```

Figure 3: A correspondence proof.

6. Conclusion

- **QuickCheck is successfully ported** by postulating functions and implementing types and classes
- Important implicit properties of QuickCheck can be made explicit in Agda
- Tests can be shown to correctly correspond to propositions

Future work

- Port remaining QuickCheck functionality
- Add surjectivity proofs of generators for stronger correspondence proofs
- Add support for `Property` in correspondence proofs

[1] K. Claessen and J. Hughes, "Quickcheck: a lightweight tool for random testing of haskell programs," in Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming, ser. ICFP'00. New York, NY, USA: Association for Computing Machinery, 2000, p. 268–279. [Online]. Available: https://doi.org/10.1145/351240.351266
[2] U. Norell, "Towards a practical programming language based on dependent type theory," Ph.D. dissertation, Chalmers University of Technology and Goteborg University, 2007.
[3] J. Cockx, O. Melkonian, L. Escot, J. Chapman, and U. Norell, "Reasonable agda is correct haskell: writing verified haskell using agda2hs," in Proceedings of the 15th ACM SIGPLAN International Haskell Symposium, ser. Haskell 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 108–122. [Online].Available: https://doi.org/10.1145/3546189.3549920

Maurits Sloof – M.L.C.Sloof@student.tudelft.nl CSE3000 – Research Project (2025/26Q4) Supervisor: Nathaniel Burke; Resp. Prof.: Jesper Cockx