Practical Verification of the Inductive Graph Library

1 Research Questions

- What preconditions does the library require?
- What are the properties and invariants of the library?
- Is it possible and feasible to state and proof the properties of the library?
- Does the library fall within the common subset identified by agda2hs?

2 Background

Practical Verification

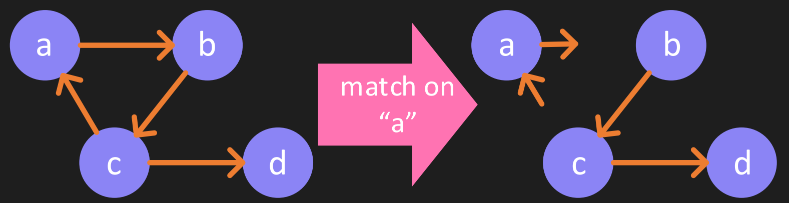
Verifying if a library works correctly is usually done with testing, however this approach is incomplete. To know for sure that a library works as intended, its properties should be proved.

Haskell and Agda

Haskell is a non-total functional programming language. Agda is very similar, but is a total and dependently typed language, so it can be used as a proof assistant.

Inductive Graphs

The Inductive graph is defined as an abstract class. Inductive graphs can be decomposed into parts as shown in the example below.



3 Haskell to Agda

```
context :: (Graph gr) => gr a b -> Node -> Context a b
context g v = fromMaybe (error (show v)) (fst (match v g))
```

```
context : ⦃ _ : Graph gr ⦄ -> (grab : gr a b) -> (n : Node) ->
          ⦃ IsTrue (isJust (fst (match n grab))) ⦄ -> Context a b
context g v = fromJust (fst (match v g))
```

4 Verification Process

Abstract Properties

How to proof properties of abstract functions that are required by preconditions?

- Make the property part of the abstract class.
- Provide an instance of the abstract class.
- Proof the property using the implemented instance.

```
prop : (grab : gr a b) -> isEmpty grab ≡ null (nodesList grab)
```

Abstract Properties Class

How to proof the properties of an abstract class?

- State all properties in an abstract properties class.
- Provide an instance of the abstract properties class.
  - This checks if the properties are correctly stated.
  - This proofs the properties and thus verifies the abstract class.

5 agda2hs

How to compile Agda code to Haskell with agda2hs?

- Code must fall within the common subset identified by agda2hs.
- Add compile pragma: {-# COMPILE AGDA2HS context #-}
- Run agda2hs.

6 Preconditions

- nonempty graph.
- specific node is contained in the graph.
- Nodes referenced by edge are contained in the graph.

Invariants

- A graph should not contain duplicate nodes.
- Edges in the graph should only reference nodes that are in the graph.

7 Conclusions

- An instance of the graph is required to verify the Inductive Graph Library.
- The use of an abstract properties class cannot guarantee that the stated properties are correct.
- Formal verification is not feasible if the only goal is to verify the abstract class.
- The properties that are proven did not produce issues, but the verification is not complete. So, no definitive conclusions can be made.
- With some changes, the library falls within the common subset of agda2hs.

Researcher: Rico van Buren
r.p.g.vanburen@student.tudelft.nl

TU Delft

Supervisors: Jesper Cockx
j.g.h.cockx@tudelft.nl

Lucas Escot
l.f.b.escot@tudelft.nl