

From Proofs of Formal Propositions to Executable Implementations



Author: František Silváši

Supervisor: doc. Ing. Martin Tomášek, PhD.

Technical University of Košice

Faculty of Electrical Engineering and Informatics, Department of Computers and Informatics

- Parse a type signature and additional definitions in any programming language as long as parametric polymorphism is expressible

Any `a => total id : a -> a`

- Interpret the type signature as a proposition and attempt to automatically find a constructive proof

$$\frac{\Gamma \vdash \text{clear}\langle a_1 \rangle}{a_1 : a, \Gamma \vdash \text{trivial}\langle a_1 \rangle} \quad \frac{a_1 : a, \Gamma \vdash a}{\Gamma \vdash a \rightarrow a} \text{intro}\langle a_1 \rangle$$

- Translate the proof into any programming language

```
id = intro<a1> trivial<a1> clear<a1>
id = λa1 → a1
id = λa1 → a1 -- Valid Haskell
```