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 siganture as a proposition and attempt to automatically find a constructive proof

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash}{a_1 : a,\ \Gamma \vdash}\ \boxed{\texttt{clear}\langle a_1\rangle}}{a_1 : a,\ \Gamma \vdash a}\ \boxed{\texttt{trivial}\langle a_1\rangle}}{\Gamma \vdash a \to a}\ \boxed{\texttt{intro}\langle a_1\rangle}$$

- Translate the proof into any programming language

id =	intro< a_1 >	trivial< a_1 >	clear< a_1 >
id =	$\lambda a_1 \to$	a_1	

id = $\lambda a1 \to a1$ -- Valid Haskell