

Automatically and Efficiently Illustrating Polynomial Equalities in Agda

Donnacha Oisín Kidney

March 26, 2019

Agda is both a mathematical formalism and an executable programming language.

As a programming language, it is similar to Haskell.

```
reverse :  $\forall \{a\} \{A : \text{Set } a\} \rightarrow \text{List } A \rightarrow \text{List } A$   
reverse [] = []  
reverse (x :: xs) = reverse xs ++ x :: []
```

Ulf Norell and James Chapman. Dependently Typed Programming in Agda.

2008

Mathematical Formalisms

Classical

Zermelo-Fraenkel Set Theory

Constructivist

Martin-Löf's Intuitionistic
Type Theory

Agda, Idris

Calculus of Constructions

Coq

Axiom of Choice

$$\forall P. P \vee \neg P$$

Proof by Contradiction

$$\forall P. \neg \neg P \rightarrow P$$

Two Problems

Formalisms are too Verbose

A. N. Whitehead and B. Russell.

Principia Mathematica. Vol. I.

1910

$1 + 1 = 2$ proven on page 360.

Automation is Untrustworthy

Kenneth Appel and Wolfgang

Haken. The Solution of the
Four-Color-Map Problem.

Scientific American,

237(4):108–121, 1977

This Project

lemma : $\forall x y \rightarrow x + y * 1 + 3 \approx 2 + 1 + y + x$

lemma x y = begin

$x + y * 1 + 3 \approx \langle \text{refl } \langle +\text{-cong} \rangle * \text{-identity}^r y \langle +\text{-cong} \rangle \text{refl } \{3\} \rangle$

$x + y + 3 \approx \langle +\text{-comm } x y \langle +\text{-cong} \rangle \text{refl} \rangle$

$y + x + 3 \approx \langle +\text{-comm } (y + x) 3 \rangle$

$3 + (y + x) \approx \langle \text{sym } (+\text{-assoc } 3 y x) \rangle$

$2 + 1 + y + x$ ■

lemma = solve NatRing

Figure 1: A Tedious Proof

Figure 2: Our Solver

The Algorithm

Horner normal form

Reflection

Perfectly suited

Questions?