Program Extraction from Large Proof Developments

Luís Cruz-Filipe[1,2]
Bas Spitters[1]

[1] University of Nijmegen, Netherlands
[2] Center for Logic and Computation, IST (Lisbon), Portugal

Why?

- Large constructive library
- Coq has extraction mechanism
- It doesn’t work

Contents

Extraction

- BHK-interpretation: connectives
- Kleene's realizability: a more formal approach
- Curry–Howard isomorphism: proofs $\longleftrightarrow$ programs
- In practice: algorithm vs. properties; types as "markers"

FTA-logic

- No elimination of **Prop** terms over **Set** $\rightsquigarrow$ no function definition by cases

- All logic in **Set**

- Extracted program too big

A solution?

Identify *computationally meaningful* propositions; put everything else in $\mathbf{Prop}$.

$\leadsto$ most proof terms can be put back in $\mathbf{Prop}$

$\leadsto$ significant amount of "dead code" is eliminated

(for more details see paper in Procs. TPHOLS)

$$\neg \quad : \quad s \to \mathbf{Prop}$$

$$\to \quad : \quad s_1 \to s_2 \to s_2$$

$$\vee \quad : \quad s_1 \to s_2 \to \mathbf{Set}$$

$$\wedge \quad : \quad s_1 \to s_2 \to \begin{cases} \mathbf{Prop} & s_1 = s_2 = \mathbf{Prop} \\ \mathbf{Set} & \text{otherwise} \end{cases}$$

$$\forall \quad : \quad \sqcap(A : s_1).(A \to s_2) \to s_2$$

$$\exists \quad : \quad \sqcap(A : \mathbf{Set}).(A \to s) \to \mathbf{Set}$$

Results

- FTA: extracts, compiles, runs... but does not terminate
- Rational numbers: everything is (almost) instantaneous
- Somewhere in between: e, π and $\sqrt{2}$

Computing e

$$e \stackrel{\text{def}}{=} \sum_{n=0}^{+\infty} \frac{1}{k!}$$

$\rightsquigarrow$ each term is a rational (constant sequence)

$\rightsquigarrow$ but much is going on. . .

Immediate Problems...

- Unary natural numbers

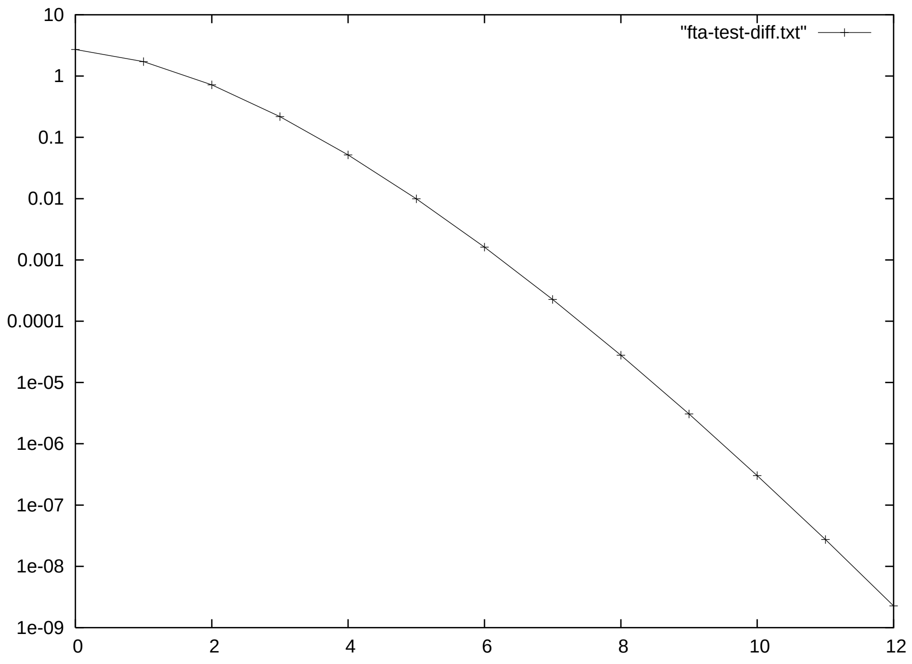
- A direct proof of $k! \neq 0$ requires computing $k!$ in unary notation

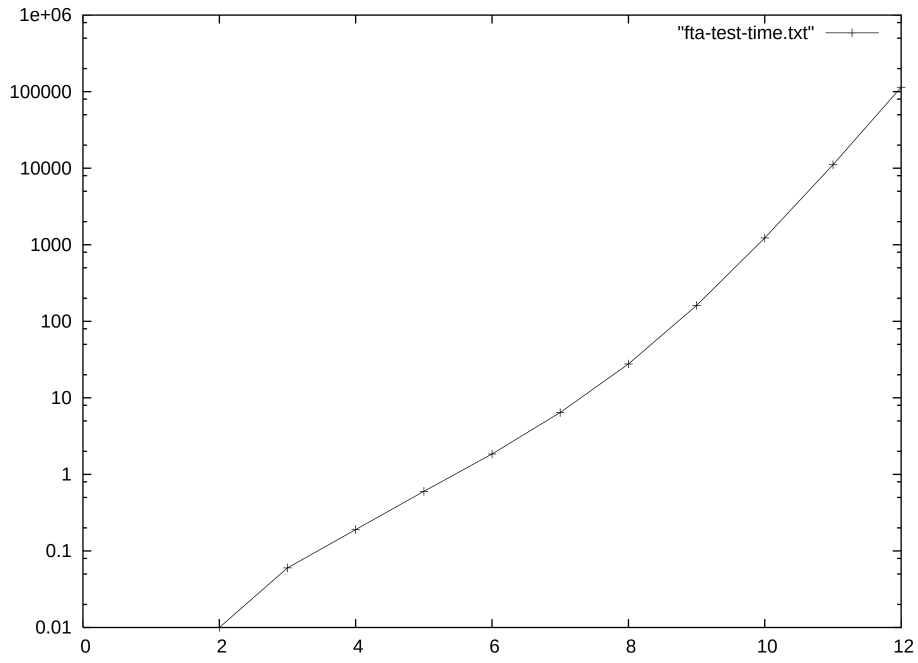
... & Solutions

- Directly inject $\mathbb{Z}^+$ into $\mathbb{R}$

- Prove $k! \neq 0$ by induction on k

Some statistics. . .





Still better

Optimize performance by working directly in the model:

- More efficient definition of factorial

- Simpler proofs and smaller proof terms

$\rightsquigarrow$ 100^{th} approximation in 77 seconds (with 157 correct digits)

Conclusions

- The more abstract the formalization, the less efficient the extracted program

- Obtaining a *working* program is far from straightforward

- Small, carefully thought, modifications in the formalization can make huge differences in the extracted program

- Future improvements in Coq may also make huge differences...

Future Work

- A similar analysis of $\sqrt{2}$

- Improving the extraction mechanism: pruning, modules

- Eventually: the FTA