Execution of Extracted Programs

Luís Cruz-Filipe[1] Pierre Letouzey[2]

[1]Center for Logic and Computation
Lisbon, Portugal

[2]
Mathematisches Institut
Ludwig-Maximilians-Universitt Mnchen
Mnchen, Germany

CALCULEMUS workshop
July 18th 2005

Motivation

- Coq extraction
- C-CoRN library
- work on extracting programs from C-CoRN (with Bas Spitters)

Motivation

- Coq extraction
- C-CoRN library
- work on extracting programs from C-CoRN (with Bas Spitters)

Motivation

- Coq extraction
- C-CoRN library
- work on extracting programs from C-CoRN (with Bas Spitters)

Practical problems

- extracted code unreadable
- compilation requires manual editing
- compiled code does not terminate

Practical problems

- extracted code unreadable
- compilation requires manual editing
- compiled code does not terminate

Practical problems

- extracted code unreadable
- compilation requires manual editing
- compiled code does not terminate

- Coq type system too powerful:
  - types may depend on terms
  - records may include types
- possibility of using unsafe coercions
- correctness still guaranteed

- Coq type system too powerful:
  - types may depend on terms
  - records may include types
- possibility of using unsafe coercions
- correctness still guaranteed

- Coq type system too powerful:
  - types may depend on terms
  - records may include types
- possibility of using unsafe coercions
- correctness still guaranteed

- Coq type system too powerful:
  - types may depend on terms
  - records may include types
- possibility of using unsafe coercions
- correctness still guaranteed

- Coq type system too powerful:
  - types may depend on terms
  - records may include types
- possibility of using unsafe coercions
- correctness still guaranteed

Definitions

e is defined as the sum of the series

$$e = \sum_{n=0}^{\infty} \frac{1}{n!}$$

- first non-trivial example of a "real" real number
- precise representation depends on proof terms

Definitions

e is defined as the sum of the series

$$e = \sum_{n=0}^{\infty} \frac{1}{n!}$$

- first non-trivial example of a "real" real number
- precise representation depends on proof terms

Definitions

e is defined as the sum of the series

$$e = \sum_{n=0}^{\infty} \frac{1}{n!}$$

- first non-trivial example of a "real" real number
- precise representation depends on proof terms

Immediate problems & solutions

- natural numbers in unary notation changed to binary notation
- proofs by induction changed to more structured proofs

Immediate problems & solutions

- natural numbers in unary notation changed to binary notation
- proofs by induction changed to more structured proofs

Immediate problems & solutions

- natural numbers in unary notation changed to binary notation
- proofs by induction changed to more structured proofs

Immediate problems & solutions

- natural numbers in unary notation changed to binary notation
- proofs by induction changed to more structured proofs

Immediate problems & solutions

- natural numbers in unary notation changed to binary notation
- proofs by induction changed to more structured proofs

Less trivial problems

- no advantage is taken of the concrete model
- solution: parameterize on proof terms
- efficient program!

Less trivial problems

- no advantage is taken of the concrete model
- solution: parameterize on proof terms
- efficient program!

Less trivial problems

- no advantage is taken of the concrete model
- solution: parameterize on proof terms
- efficient program!

Main strategy

Try to apply the same techniques that worked so well for e:

- identify and attack potential bottlenecks
- factor common proof steps
- change proofs in an intelligent way

Main strategy

Try to apply the same techniques that worked so well for e:

- identify and attack potential bottlenecks
- factor common proof steps
- change proofs in an intelligent way

Main strategy

Try to apply the same techniques that worked so well for e:

- identify and attack potential bottlenecks
- factor common proof steps
- change proofs in an intelligent way

Main strategy

Try to apply the same techniques that worked so well for e:

- identify and attack potential bottlenecks
- factor common proof steps
- change proofs in an intelligent way

Classical IVT

Theorem

Let f be a function defined on $[a, b]$ with $f(a) < y < f(b)$. Then there is $x \in [a, b]$ such that $f(x) = y$.

A constructive variant

Theorem

Let f be a locally non-constant function...

Definition

A function f is locally non-constant if, on every interval $[a, b]$, it satisfies

$$\forall y \exists x. f(x) \neq y.$$

Lemma

Polynomials are locally non-constant.

A constructive variant

Theorem

Let f be a locally non-constant function...

Definition

A function f is locally non-constant if, on every interval $[a, b]$, it satisfies

$$\forall y \exists x. f(x) \neq y.$$

Lemma

Polynomials are locally non-constant.

A constructive variant

Theorem

Let f be a locally non-constant function...

Definition

A function f is locally non-constant if, on every interval $[a, b]$, it satisfies

$$\forall y \exists x. f(x) \neq y.$$

Lemma

Polynomials are locally non-constant.

Another constructive variant

Theorem

Let f be a strictly increasing function...

Lemma

$\lambda x.x^n - c$ *is strictly increasing on* $[0, c+1]$.

Another constructive variant

Theorem

Let f be a strictly increasing function...

Lemma

$\lambda x.x^n - c$ is strictly increasing on $[0, c+1]$.

The bad news

After several rounds of optimizations. . . the extracted program is still VERY slow.

- exponential complexity (factor ≈ 3)
- no obvious improvements left. . .

The bad news

After several rounds of optimizations... the extracted program is still VERY slow.

- exponential complexity (factor ≈ 3)
- no obvious improvements left...

The bad news

After several rounds of optimizations... the extracted program is still VERY slow.

- exponential complexity (factor ≈ 3)
- no obvious improvements left...

The bad news

After several rounds of optimizations... the extracted program is still VERY slow.

- exponential complexity (factor ≈ 3)
- no obvious improvements left...

An alternative approach

Formalization of the FTA proof in the concrete model, with extraction in mind.

- reduced fractions
- explicit bounds for Cauchy sequences
- functions as limit of rational-valued functions

Much better results!

Unfortunately, not portable to C-CoRN

An alternative approach

Formalization of the FTA proof in the concrete model, with extraction in mind.

- reduced fractions
- explicit bounds for Cauchy sequences
- functions as limit of rational-valued functions

Much better results!

Unfortunately, not portable to C-CoRN

An alternative approach

Formalization of the FTA proof in the concrete model, with extraction in mind.

- reduced fractions
- explicit bounds for Cauchy sequences
- functions as limit of rational-valued functions

Much better results!

Unfortunately, not portable to C-CoRN

An alternative approach

Formalization of the FTA proof in the concrete model, with extraction in mind.

- reduced fractions
- explicit bounds for Cauchy sequences
- functions as limit of rational-valued functions

Much better results!

Unfortunately, not portable to C-CoRN

An alternative approach

Formalization of the FTA proof in the concrete model, with extraction in mind.

- reduced fractions
- explicit bounds for Cauchy sequences
- functions as limit of rational-valued functions

Much better results!

Unfortunately, not portable to C-CoRN

An alternative approach

Formalization of the FTA proof in the concrete model, with extraction in mind.

- reduced fractions
- explicit bounds for Cauchy sequences
- functions as limit of rational-valued functions

Much better results!

Unfortunately, not portable to C-CoRN

- extraction is not a magic button
- extraction probably will never be a magic button
- proving and computing seem to be essentially different things
- no notion of "good" proof
- will "good" proofs yield good programs?

- extraction is not a magic button
- extraction probably will never be a magic button
- proving and computing seem to be essentially different things
- no notion of "good" proof
- will "good" proofs yield good programs?

- extraction is not a magic button
- extraction probably will never be a magic button
- proving and computing seem to be essentially different things
- no notion of "good" proof
- will "good" proofs yield good programs?

- extraction is not a magic button
- extraction probably will never be a magic button
- proving and computing seem to be essentially different things
- no notion of "good" proof
- will "good" proofs yield good programs?

- extraction is not a magic button
- extraction probably will never be a magic button
- proving and computing seem to be essentially different things
- no notion of "good" proof
- will "good" proofs yield good programs?