Formal verification of exact computations using Newton's method

Nicolas Julien and Ioana Paşca

INRIA Sophia Antipolis

TPHOLs, August 2009

Newton's method

Definition:

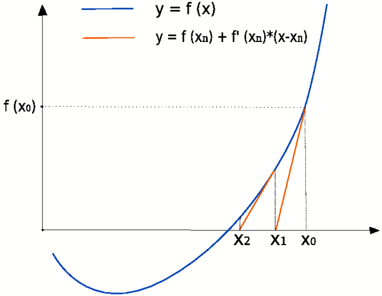
- $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$

Properties:

- convergence to the root of function *f*
- speed of convergence
- local unicity of the root
- local stability

In Coq:

- express the properties
- implement efficient computation



Outline

Real numbers in Coq
- Axiomatic reals
- Exact reals

Newton's method
- Implementation and verification
- Optimizations for efficient computation

Conclusion and future work

Real numbers in COQ

- high level proofs: *reals in* COQ *Standard Library*
  - defined by axioms
    e.g. $r_1 + (r_2 + r_3) = (r_1 + r_2) + r_3$
  - definitions and proofs from “paper mathematics”
    e.g. convergence, derivability, fundamental theorem of calculus etc.
  - but no computational power

Real numbers in COQ

- high level proofs: *reals in* COQ *Standard Library*
  - defined by axioms
    e.g. $r_1 + (r_2 + r_3) = (r_1 + r_2) + r_3$
  - definitions and proofs from "paper mathematics"
    e.g. convergence, derivability, fundamental theorem of calculus etc.
  - but no computational power
- efficient computation: *library on exact real arithmetic*

Exact real arithmetic with co-inductive streams

Representation

- compute a real number in $[-1, 1]$ with arbitrary precision
- real numbers represented as streams of signed digits in base β
  e.g. $\frac{1}{3} = 0.333\ldots = [\![3::3::3\ldots]\!]_{10} = [\![4::-7::4::-7\ldots]\!]_{10}$
  $$[\![s]\!]_\beta = [\![d_1::d_2::d_3::\ldots]\!]_\beta = \sum_{i=1}^{\infty} \frac{d_i}{\beta^i};\ -\beta < d_i < \beta$$
- notice $[\![d_1::\overline{s}]\!]_\beta = \frac{d_1 + [\![\overline{s}]\!]_\beta}{\beta}$

Exact real arithmetic with co-inductive streams

Representation

- compute a real number in $[-1, 1]$ with arbitrary precision
- real numbers represented as streams of signed digits in base β
  e.g. $\frac{1}{3} = 0.333\ldots = [\![3::3::3\ldots]\!]_{10} = [\![4::-7::4::-7\ldots]\!]_{10}$
  $$[\![s]\!]_\beta = [\![d_1::d_2::d_3::\ldots]\!]_\beta = \sum_{i=1}^{\infty} \frac{d_i}{\beta^i};\ -\beta < d_i < \beta$$
- notice $[\![d_1::\overline{s}]\!]_\beta = \frac{d_1 + [\![\overline{s}]\!]_\beta}{\beta}$
- redundant representation → useful for designing algorithms
  e.g. $[\![0::3::\ldots]\!]_{10} + [\![0::6::\ldots]\!]_{10} =?$
  $[\![0::3::3::\ldots]\!]_{10} + [\![0::6::5::\ldots]\!]_{10} = [\![1::-1::\ldots]\!]_{10}$
  $[\![0::3::3::\ldots]\!]_{10} + [\![0::6::7::\ldots]\!]_{10} = [\![1::0::\ldots]\!]_{10}$

Exact real arithmetic with co-inductive streams

Implementation

$$[\![s]\!]_\beta = [\![d_1 :: d_2 :: d_3 :: \ldots]\!]_\beta = [\![d_1 :: \overline{s}]\!]_\beta;\ -\beta < d_i < \beta$$

- in Coq: co-inductive definitions and co-recursive functions

```
CoInductive Stream (A: Type): Type:=
  | Cons: A → Stream A → Stream A.
Notation "x :: s" := Cons x s.

CoFixpoint Sopp (s: Stream digit): Stream digit:=
  match s with | d₁ :: s̄ ⇒ (−d₁) :: Sopp s̄ end.
```

Exact real arithmetic with co-inductive streams

Certification

$$[\![d_1 :: \overline{s}]\!]_\beta = \frac{d_1 + [\![\overline{s}]\!]_\beta}{\beta}$$

- link the exact reals with axiomatic reals

```
Variable β : ℕ.
CoInductive represents: Stream digit → R → Prop:=
  | rep: ∀ s r k, −β < k < β → −1 ≤ r ≤ 1 →
  represents s r → represents (k :: s) (k+r)/β .
Notation " s ≃ r " := represents s r.
```

- certify implementations via this relation

```
Theorem Sopp_correct: ∀ s r, s ≃ r → (Sopp s) ≃ (−r).
```

Newton's method

$$f, x_0, \; x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Properties

- $\lim_{n \to \infty} x_n = x^*$
- $f(x^*) = 0$
- speed of convergence
  $|x_n - x^*| \leq \Delta_n$
- local stability
  $\forall x'_0 \in U_{x_0}, x'_n \to x^*$

Proofs

concepts from real analysis:

- continuity
- derivability
- mean value theorem
- convergence of sequences
- completness of $\mathbb{R}$ etc.

formalize proofs on axiomatic reals of COQ

Implementation of Newton's method

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

on streams

$$Sx_0 := s_0$$
$$Sx_{n+1} := Sx_n \ominus g(Sx_n)$$

on axiomatic reals

$$Rx_0 := r_0$$
$$Rx_{n+1} := Rx_n - \frac{f(Rx_n)}{f'(Rx_n)}$$

```
Theorem Snewt_correct: (* ... *) (Sxn g s0 n) ≃ (Rxn f f' r0 n).
```

- we can express properties on elements of Newton's sequence
- but, we cannot reason about the root of the function
- we want to compute the root in arbitrary precision

Newton for streams

Goal define a co-recursive algorithm to compute the root x^* of the function f

- produce the first digit
- use a guarded co-recursive call

Newton for streams

Goal define a co-recursive algorithm to compute the root x^* of the function f

- produce the first digit
- use a guarded co-recursive call

Idea

- start with f and x_0
- speed of convergence $\Rightarrow n$ s.t. $x_n = \frac{d_1 + \overline{x_n}}{\beta} \Rightarrow x^* = \frac{d_1 + \overline{x^*}}{\beta}$

Newton for streams

Goal define a co-recursive algorithm to compute the root x^* of the function f

- produce the first digit
- use a guarded co-recursive call

Idea

- start with f and x_0
- speed of convergence $\Rightarrow n$ s.t. $x_n = \frac{d_1 + \overline{x_n}}{\beta} \Rightarrow x^* = \frac{d_1 + \overline{x^*}}{\beta}$
- $f(x^*) = 0 \Rightarrow f(\frac{d_1 + \overline{x^*}}{\beta}) = 0$
- define $f_1(x) := f(\frac{d_1 + x}{\beta}) \Rightarrow f_1(\overline{x^*}) = 0$
- repeat process to get the first digit of $\overline{x^*}$; start with f_1 and $\overline{x_n}$

Newton for streams

Goal define a co-recursive algorithm to compute the root x^* of the function f

- produce the first digit
- use a guarded co-recursive call

Idea

- start with f and x_0
- speed of convergence $\Rightarrow n$ s.t. $x_n = \frac{d_1 + \overline{x_n}}{\beta} \Rightarrow x^* = \frac{d_1 + \overline{x^*}}{\beta}$
- $f(x^*) = 0 \Rightarrow f(\frac{d_1 + \overline{x^*}}{\beta}) = 0$
- define $f_1(x) := f(\frac{d_1 + x}{\beta}) \Rightarrow f_1(\overline{x^*}) = 0$
- repeat process to get the first digit of $\overline{x^*}$; start with f_1 and $\overline{x_n}$
- $g = \frac{f}{f'} \Rightarrow g_1(x) := \frac{f_1(x)}{f_1'(x)} = \frac{f(\frac{d_1+x}{\beta})}{\frac{1}{\beta} f'(\frac{d_1+x}{\beta})} = \beta \times g(\frac{d_1+x}{\beta})$

Newton for streams

Idea

- to produce a first digit of x^* determine $x_n = \frac{d_1 + \overline{x_n}}{\beta}$ s.t. $x^* = \frac{d_1 + \overline{x^*}}{\beta}$
- do a co-recursive call with function $g_1(x) = \beta \times g(\frac{d_1 + x}{\beta})$ and $\overline{x_n}$

Algorithm

```
CoFixpoint exact_newton g s0 n:=
  match (make_digit (Sxn g s0 n)) with
    |d1 :: sn ⇒ d1 :: exact_newton (fun s ⇒ (β ⊙ g(d1 :: s))) sn n
  end.
```

Newton for streams

Idea

- to produce a first digit of x^* determine $x_n = \frac{d_1 + \overline{x_n}}{\beta}$ s.t. $x^* = \frac{d_1 + \overline{x^*}}{\beta}$
- do a co-recursive call with function $g_1(x) = \beta \times g(\frac{d_1 + x}{\beta})$ and $\overline{x_n}$

Algorithm

```
CoFixpoint exact_newton g s0 n:=
  match (make_digit (Sxn g s0 n)) with
    | d1 :: s̄n ⇒ d1 :: exact_newton (fun s ⇒ (β ⊙ g(d1 :: s))) s̄n n
  end.
```

```
Theorem exact_newton_correct: (* ... *)
  (exact_newton g s0 n) ≃ x*.
```

- ensure the same hypotheses for $\overline{x_n}$ and g_1 as for x_0 and g

Rounding for efficiency

$$Sx_{n+1} := Sx_n \ominus g(Sx_n)$$

- the internal precision is too high
  e.g. $\frac{\sqrt{2}}{2} = 0.7071067$
  $Sx_n = 0.709876\ldots$ $Sx_{n+1} = 0.70715\ldots$
  $Sx'_n = 0.700000\ldots$ $Sx_{n+1} = 0.70705\ldots$

Solution:

- use only the meaningfull digits for each iteration

Certified rounding

Newton's method with rounding:

$$t_0 = x_0 \qquad\qquad t_{n+1} = rnd_{n+1}(t_n - \frac{f(t_n)}{f'(t_n)})$$

To prove that $t_n \rightarrow x^*$
use local stability: $\forall x_0' \in U_{x_0}, x_n(x_0') \rightarrow x^*$

- $x_n(x_0)$: $x_0, x_1, x_2, x_3, \ldots \rightarrow x^*$
- $x_n(x_1)$: $x_1, x_2, x_3 \ldots \rightarrow x^*$
- $x_n(\widetilde{x_1})$: $\widetilde{x_1}, \widetilde{x_2}, \widetilde{x_3} \ldots \rightarrow x^*$
- $x_n(\widetilde{x_2})$: $\widetilde{x_2}, \widetilde{x_3} \ldots \rightarrow x^*$
- $x_n(\widetilde{\widetilde{x_2}})$: $\widetilde{\widetilde{x_2}}, \widetilde{\widetilde{x_3}} \ldots \rightarrow x^*$
- $\ldots$

Conclusion and future work

- we have a verified algorithm for computing the root of a function
  - goal: provide an efficient algorithm for the exact real library on streams
- we have a verified rounding process for Newton's method
  - goal: reuse the result in other contexts like floating point computations

Properties for Newton's method

Given the equation $f(x) = 0$, with $f : [a,b] \to \mathbb{R}$, $f(x) \in C^{(1)}([a,b])$ and $x^{(0)} \in]a,b[$ such that $\overline{U_\varepsilon}(x^{(0)}) = \{|x - x^{(0)}| \leq \varepsilon\} \subset]a,b[$.
If:

I. $f'(x^{(0)}) \neq 0$ and $|\frac{1}{f'(x^{(0)})}| \leq A_0$;

II. $|\frac{f(x^{(0)})}{f'(x^{(0)})}| \leq B_0 \leq \frac{\varepsilon}{2}$;

III. $\forall x, y \in [a,b], |f'(x) - f'(y)| \leq C|x - y|$

IV. $\mu_0 = 2A_0B_0C \leq 1$.

Then, Newton's method:
$x^{(n+1)} = x^{(n)} - \frac{f(x^{(n)})}{f'(x^{(n)})}$

1. converges, $\lim_{n\to\infty} x^{(n)} = x^*$ and $f(x^*) = 0$
2. the root x^* is unique in $\{|x - x^{(0)}| \leq 2B_0\}$
3. the speed of convergence is given by $|x^{(n)} - x^*| \leq \frac{1}{2^{n-1}} \mu_0^{2^n - 1} B_0$
4. if, additionally, $0 < \mu_0 < 1$ and $[x^{(0)} - \frac{2}{\mu_0}B_0, x^{(0)} + \frac{2}{\mu_0}B_0] \subset]a,b[$, then $\forall x'^{(0)}$ s.t. $|x'^{(0)} - x^{(0)}| \leq \frac{1-\mu_0}{2\mu_0}B_0$ the associated Newton's process converges to x^*

Newton with rounding

Theorem

We consider a function $f :]a, b[\to \mathbb{R}$ and an initial approximation $x^{(0)}$ satisfying the conditions in Theorem 1.
We also consider a function $rnd : \mathbb{N} \times \mathbb{R} \to \mathbb{R}$ that models the approximation we will make at each step in the perturbed Newton sequence:
$t^{(0)} = x^{(0)}$ and $t^{(n+1)} = rnd_{n+1}(t^{(n)} - f(t^{(n)})/f'(t^{(n)}))$
If

1. $\forall n \forall x, x \in]a, b[\Rightarrow rnd_n(x) \in]a, b[$
2. $\frac{1}{2} \leq \mu_0 < 1$
3. $[x^{(0)} - 3B_0, x^{(0)} + 3B_0] \subset]a, b[$
4. $\forall n \forall x, |x - rnd_n(x)| \leq \frac{1}{3^n} R_0$, where $R_0 = \frac{1-\mu_0^2}{8\mu_0} B_0$

then

a. the sequence $\{t^{(n)}\}_{n \in \mathbb{N}}$ converges and $\lim_{n \to \infty} t^{(n)} = x^*$ where x^* is the root of the function f given by Theorem 1
b. $\forall n, |x^* - t^{(n)}| \leq \frac{1}{2^{n-1}} B_0$