Interval computations and some computable analysis

Florian Steinberg (some is joint work with Laurent Théry)

June 21, 2018

Prerequisites

- GMP, MPFR, MPI, Sollya.

Prerequisites

- GMP, MPFR, MPI, Sollya.
- inside of Coq: Flocq, coq-Interval, (coq-approx).

Prerequisites

- GMP, MPFR, MPI, Sollya.
- inside of Coq: Flocq, coq-Interval, (coq-approx).
- iRRAM.

Prerequisites

- GMP, MPFR, MPI, Sollya.
- inside of Coq: Flocq, coq-Interval, (coq-approx).
- iRRAM.

Sollya: Chebyshev Approximation for fast interval computations.

Prerequisites

- GMP, MPFR, MPI, Sollya.
- inside of Coq: Flocq, coq-Interval, (coq-approx).
- iRRAM.

Sollya: Chebyshev Approximation for fast interval computations.

Documentation in Mioara Joldes PhD Thesis
"Approximations polynomiales rigoureuses et applications".

Prerequisites

- GMP, MPFR, MPI, Sollya.
- inside of Coq: Flocq, coq-Interval, (coq-approx).
- iRRAM.

Sollya: Chebyshev Approximation for fast interval computations.

Documentation in Mioara Joldes PhD Thesis
"Approximations polynomiales rigoureuses et applications".

Goal: Get some of these algorithms to run inside of Coq.

Prerequisites

- GMP, MPFR, MPI, Sollya.
- inside of Coq: Flocq, coq-Interval, (coq-approx).
- iRRAM.

Sollya: Chebyshev Approximation for fast interval computations.

Documentation in Mioara Joldes PhD Thesis
"Approximations polynomiales rigoureuses et applications".

Goal: Get some of these algorithms to run inside of Coq.

Starting point: Coq-Interval.

Prerequisites

- GMP, MPFR, MPI, Sollya.
- inside of Coq: Flocq, coq-Interval, (coq-approx).
- iRRAM.

Sollya: Chebyshev Approximation for fast interval computations.

Documentation in Mioara Joldes PhD Thesis
"Approximations polynomiales rigoureuses et applications".

Goal: Get some of these algorithms to run inside of Coq.

Starting point: Coq-Interval. Supports Taylor approximation.

Prerequisites

- GMP, MPFR, MPI, Sollya.
- inside of Coq: Flocq, coq-Interval, (coq-approx).
- iRRAM.

Sollya: Chebyshev Approximation for fast interval computations.

Documentation in Mioara Joldes PhD Thesis
"Approximations polynomiales rigoureuses et applications".

Goal: Get some of these algorithms to run inside of Coq.

Starting point: Coq-Interval. Supports Taylor approximation.

Returns reasonable results?

Prerequisites

- GMP, MPFR, MPI, Sollya.
- inside of Coq: Flocq, coq-Interval, (coq-approx).
- iRRAM.

Sollya: Chebyshev Approximation for fast interval computations.

Documentation in Mioara Joldes PhD Thesis
"Approximations polynomiales rigoureuses et applications".

Goal: Get some of these algorithms to run inside of Coq.

Starting point: Coq-Interval. Supports Taylor approximation.

Returns reasonable results? $\rightsquigarrow$ Try to prove it correct.

Prerequisites

- GMP, MPFR, MPI, Sollya.
- inside of Coq: Flocq, coq-Interval, (coq-approx).
- iRRAM.

Sollya: Chebyshev Approximation for fast interval computations.

Documentation in Mioara Joldes PhD Thesis
"Approximations polynomiales rigoureuses et applications".

Goal: Get some of these algorithms to run inside of Coq.

Starting point: Coq-Interval. Supports Taylor approximation.

Returns reasonable results? $\rightsquigarrow$ Try to prove it correct.

My background: Computational complexity in computable analysis.

Prerequisites

- GMP, MPFR, MPI, Sollya.
- inside of Coq: Flocq, coq-Interval, (coq-approx).
- iRRAM.

Sollya: Chebyshev Approximation for fast interval computations.

Documentation in Mioara Joldes PhD Thesis
"Approximations polynomiales rigoureuses et applications".

Goal: Get some of these algorithms to run inside of Coq.

Starting point: Coq-Interval. Supports Taylor approximation.

Returns reasonable results? $\rightsquigarrow$ Try to prove it correct.

My background: Computational complexity in computable analysis.

Formalize results from computable analysis.

Chebyshev Approximation

Interval computation

Some computable analysis

Chebyshev Polynomials

The Chebyshev polynomials are defined as follows:

$$T_0 := 1, \quad T_1 := X, \quad T_{n+1} := 2X \cdot T_n - T_{n-1}.$$

Chebyshev Polynomials

The Chebyshev polynomials are defined as follows:

$$T_0 := 1, \quad T_1 := X, \quad T_{n+1} := 2X \cdot T_n - T_{n-1}.$$

Many good properties:

Chebyshev Polynomials

The Chebyshev polynomials are defined as follows:

$$T_0 := 1, \quad T_1 := X, \quad T_{n+1} := 2X \cdot T_n - T_{n-1}.$$

Many good properties:

- Minimal supremum-norm for any degree n Polynomial.

Chebyshev Polynomials

The Chebyshev polynomials are defined as follows:

$$T_0 := 1, \quad T_1 := X, \quad T_{n+1} := 2X \cdot T_n - T_{n-1}.$$

Many good properties:

- Minimal supremum-norm for any degree n Polynomial.
- Interpolating in the roots of Chebyshev polynomials leads to near best approximation in supremum norm.

Chebyshev Polynomials

The Chebyshev polynomials are defined as follows:

$$T_0 := 1, \quad T_1 := X, \quad T_{n+1} := 2X \cdot T_n - T_{n-1}.$$

Many good properties:

- Minimal supremum-norm for any degree n Polynomial.
- Interpolating in the roots of Chebyshev polynomials leads to near best approximation in supremum norm.
- Simple formula for multiplication:

$$2T_m T_n = T_{m+n} + T_{|m-n|}$$

Chebyshev Polynomials

The Chebyshev polynomials are defined as follows:

$$T_0 := 1, \quad T_1 := X, \quad T_{n+1} := 2X \cdot T_n - T_{n-1}.$$

Many good properties:

- Minimal supremum-norm for any degree n Polynomial.
- Interpolating in the roots of Chebyshev polynomials leads to near best approximation in supremum norm.
- Simple formula for multiplication:

$$2T_m T_n = T_{m+n} + T_{|m-n|}$$

- T_m is a basis

Chebyshev Polynomials

The Chebyshev polynomials are defined as follows:

$$T_0 := 1, \quad T_1 := X, \quad T_{n+1} := 2X \cdot T_n - T_{n-1}.$$

Many good properties:

- Minimal supremum-norm for any degree n Polynomial.
- Interpolating in the roots of Chebyshev polynomials leads to near best approximation in supremum norm.
- Simple formula for multiplication:

$$2T_m T_n = T_{m+n} + T_{|m-n|}$$

- T_m is a basis $\rightsquigarrow$ exist uniqe a_i s.t. $p = \sum_{i \leq \deg(p)} a_i T_m$.

Analytic facts

Alternate trigonometrical Definition:

$$T_n(x) = \cos(n \operatorname{acos} x), \text{ if } x \leq 1.$$

Analytic facts

Alternate trigonometrical Definition:

$$T_n(x) = \cos(n \operatorname{acos} x), \text{ if } x \leq 1.$$

Orthonomal Basis in a weighted L^2 space over the unit interval.

$$\int_{-1}^{1} \frac{T_n(x)\, T_m(x)}{\sqrt{1-x^2}}\, dx = \begin{cases} 0 & \text{if } n \neq m \\ \pi & \text{if } n = 0 = m \\ \frac{\pi}{2} & \text{if } n = m \neq 0 \end{cases}$$

Analytic facts

Alternate trigonometrical Definition:

$$T_n(x) = \cos(n \operatorname{acos} x), \text{ if } x \leq 1.$$

Orthonomal Basis in a weighted L^2 space over the unit interval.

$$\int_{-1}^{1} \frac{T_n(x)\, T_m(x)}{\sqrt{1-x^2}}\, dx = \begin{cases} 0 & \text{if } n \neq m \\ \pi & \text{if } n = 0 = m \\ \frac{\pi}{2} & \text{if } n = m \neq 0 \end{cases}$$

$\rightsquigarrow$ formula for coefficients a_i:

Analytic facts

Alternate trigonometrical Definition:

$$T_n(x) = \cos(n \operatorname{acos} x), \text{ if } x \leq 1.$$

Orthonomal Basis in a weighted L^2 space over the unit interval.

$$\int_{-1}^{1} \frac{T_n(x)\, T_m(x)}{\sqrt{1-x^2}} dx = \begin{cases} 0 & \text{if } n \neq m \\ \pi & \text{if } n = 0 = m \\ \frac{\pi}{2} & \text{if } n = m \neq 0 \end{cases}$$

$\rightsquigarrow$ formula for coefficients a_i:

$$a_i \sim \langle p, T_i \rangle = \int_{-1}^{1} \frac{p T_i(x)}{\sqrt{1-x^2}} dx$$

Analytic facts

Alternate trigonometrical Definition:

$$T_n(x) = \cos(n \operatorname{acos} x), \text{ if } x \leq 1.$$

Orthonomal Basis in a weighted L^2 space over the unit interval.

$$\int_{-1}^{1} \frac{T_n(x)\, T_m(x)}{\sqrt{1-x^2}} dx = \begin{cases} 0 & \text{if } n \neq m \\ \pi & \text{if } n = 0 = m \\ \frac{\pi}{2} & \text{if } n = m \neq 0 \end{cases}$$

$\rightsquigarrow$ formula for coefficients a_i:

$$a_i \sim \langle p, T_i \rangle = \int_{-1}^{1} \frac{p T_i(x)}{\sqrt{1-x^2}} dx$$

$\rightsquigarrow$ closely related to Fourier-series.

Current state

Using mathcomp library for polynomials over a Ring.

Current state

Using mathcomp library for polynomials over a Ring.

- Defined the Chebyshev polynomials.

Current state

Using mathcomp library for polynomials over a Ring.

- Defined the Chebyshev polynomials.
- Proved some basic algebraic facts.

Current state

Using mathcomp library for polynomials over a Ring.

- Defined the Chebyshev polynomials.
- Proved some basic algebraic facts.

Using Coquelicot and mathcomp:

Current state

Using mathcomp library for polynomials over a Ring.

- Defined the Chebyshev polynomials.
- Proved some basic algebraic facts.

Using Coquelicot and mathcomp:

- Proved $T_n(x) = \cos(n \operatorname{acos}(x))$.

Current state

Using mathcomp library for polynomials over a Ring.

- Defined the Chebyshev polynomials.
- Proved some basic algebraic facts.

Using Coquelicot and mathcomp:

- Proved $T_n(x) = \cos(n \operatorname{acos}(x))$.
- Proved orthogonality.

Executability

The ssreflect library is not centered around executability.

Executability

The ssreflect library is not centered around executability.
Execute on lists: `ladd_poly: seq R -> seq R`.

Executability

The ssreflect library is not centered around executability.
Execute on lists: `ladd_poly: seq R -> seq R`.
Provide specification results.

Executability

The ssreflect library is not centered around executability.
Execute on lists: `ladd_poly: seq R -> seq R`.
Provide specification results.

Lemma (`ladd_poly_spec`)

```
∀(l k: seq R), Poly (ladd l k) = Poly l + Poly k.
```

Executability

The ssreflect library is not centered around executability.
Execute on lists: `ladd_poly: seq R -> seq R`.
Provide specification results.

Lemma (`ladd_poly_spec`)

$\forall$`(l k: seq R), Poly (ladd l k) = Poly l + Poly k.`

In the above: `Poly l = sum_(i < size l) l_i * X^i`.

Executability

The ssreflect library is not centered around executability.
Execute on lists: `ladd_poly: seq R -> seq R`.
Provide specification results.

Lemma (`ladd_poly_spec`)

$\forall$`(l k: seq R), Poly (ladd l k) = Poly l + Poly k.`

In the above: `Poly l = sum_(i < size l) l_i * X^i`.
To operate in Chebyshev basis replace `Poly` by the function
`CPoly l := sum_(i < size l) l_i * T_i`.

Executability

The ssreflect library is not centered around executability.
Execute on lists: `ladd_poly: seq R -> seq R`.
Provide specification results.

Lemma (`ladd_poly_spec`)

```
∀(l k: seq R), Poly (ladd l k) = Poly l + Poly k.
```

In the above: `Poly l = sum_(i < size l) l_i * X^i`.
To operate in Chebyshev basis replace `Poly` by the function
`CPoly l := sum_(i < size l) l_i * T_i`.
Provide algorithms and specification wrt `CPoly`.

Lemma (`Cshaw_spec`)

```
∀(l: seq R) (x: R), Cshaw l = (CPoly l).[x].
```

Executability

The ssreflect library is not centered around executability.
Execute on lists: `ladd_poly: seq R -> seq R`.
Provide specification results.

Lemma (`ladd_poly_spec`)

`∀(l k: seq R), Poly (ladd l k) = Poly l + Poly k.`

In the above: `Poly l = sum_(i < size l) l_i * X^i`.
To operate in Chebyshev basis replace `Poly` by the function
`CPoly l := sum_(i < size l) l_i * T_i`.
Provide algorithms and specification wrt `CPoly`.

Lemma (`Cshaw_spec`)

`∀(l: seq R) (x: R), Cshaw l = (CPoly l).[x].`

Execution works fine for computational R.

Executability

The ssreflect library is not centered around executability.
Execute on lists: `ladd_poly: seq R -> seq R`.
Provide specification results.

Lemma (`ladd_poly_spec`)

`∀(l k: seq R), Poly (ladd l k) = Poly l + Poly k.`

In the above: `Poly l = sum_(i < size l) l_i * X^i`.
To operate in Chebyshev basis replace `Poly` by the function
`CPoly l := sum_(i < size l) l_i * T_i`.
Provide algorithms and specification wrt `CPoly`.

Lemma (`Cshaw_spec`)

`∀(l: seq R) (x: R), Cshaw l = (CPoly l).[x]`.

Execution works fine for computational R. For instance $R = Q$.

Executability

The ssreflect library is not centered around executability.
Execute on lists: `ladd_poly: seq R -> seq R`.
Provide specification results.

Lemma (`ladd_poly_spec`)

`∀(l k: seq R), Poly (ladd l k) = Poly l + Poly k.`

In the above: `Poly l = sum_(i < size l) l_i * X^i`.
To operate in Chebyshev basis replace `Poly` by the function
`CPoly l := sum_(i < size l) l_i * T_i`.
Provide algorithms and specification wrt `CPoly`.

Lemma (`Cshaw_spec`)

`∀(l: seq R) (x: R), Cshaw l = (CPoly l).[x]`.

Execution works fine for computational R. For instance $R = Q$.

Intervals

The real numbers are not a computational Type.

Intervals

The real numbers are not a computational Type.
Floatingpoint types are not a ring.

Intervals

The real numbers are not a computational Type.
Floatingpoint types are not a ring.
Instead: Use Intervals, track containement.

Intervals

The real numbers are not a computational Type.
Floatingpoint types are not a ring.
Instead: Use Intervals, track containement.
I, J Intervals $\rightsquigarrow$

$$I + J := \{x + y \mid x \in I \text{ and } y \in J\} \qquad .$$

Intervals

The real numbers are not a computational Type.
Floatingpoint types are not a ring.
Instead: Use Intervals, track containement.
I, J Intervals $\rightsquigarrow$

$$I + J := \{x + y \mid x \in I \text{ and } y \in J\} \qquad .$$

If I, J have dyadic endpoints a, b, c, d.

Intervals

The real numbers are not a computational Type.
Floatingpoint types are not a ring.
Instead: Use Intervals, track containement.
I, J Intervals $\rightsquigarrow$

$$I + J := \{x + y \mid x \in I \text{ and } y \in J\} \subseteq [\lfloor a + c \rfloor, \lceil b + d \rceil].$$

If I, J have dyadic endpoints a, b, c, d.

Intervals

The real numbers are not a computational Type.
Floatingpoint types are not a ring.
Instead: Use Intervals, track containement.
I, J Intervals $\rightsquigarrow$

$$I + J := \{x + y \mid x \in I \text{ and } y \in J\} \subseteq [\lfloor a + c \rfloor, \lceil b + d \rceil].$$

If I, J have dyadic endpoints a, b, c, d.
An interval extension of $f : \mathbb{R} \to \mathbb{R}$ is a function $F : \mathbb{ID} \to \mathbb{ID}$ such that:

$$\forall x \in \mathbb{R}, I \in \mathbb{ID}, x \in I \Rightarrow f(x) \in F(I).$$

Intervals

The real numbers are not a computational Type.
Floatingpoint types are not a ring.
Instead: Use Intervals, track containement.
I, J Intervals $\rightsquigarrow$

$$I + J := \{x + y \mid x \in I \text{ and } y \in J\} \subseteq [\lfloor a + c \rfloor, \lceil b + d \rceil].$$

If I, J have dyadic endpoints a, b, c, d.
An interval extension of $f : \mathbb{R} \to \mathbb{R}$ is a function $F : \mathbb{ID} \to \mathbb{ID}$ such that:

$$\forall x \in \mathbb{R}, I \in \mathbb{ID}, x \in I \Rightarrow f(x) \in F(I).$$

Obtain Interval extension of f?

Intervals

The real numbers are not a computational Type.
Floatingpoint types are not a ring.
Instead: Use Intervals, track containement.
I, J Intervals $\rightsquigarrow$

$$I + J := \{x + y \mid x \in I \text{ and } y \in J\} \subseteq [\lfloor a + c \rfloor, \lceil b + d \rceil].$$

If I, J have dyadic endpoints a, b, c, d.
An interval extension of $f : \mathbb{R} \to \mathbb{R}$ is a function $F : \mathbb{ID} \to \mathbb{ID}$ such that:

$$\forall x \in \mathbb{R}, I \in \mathbb{ID}, x \in I \Rightarrow f(x) \in F(I).$$

Obtain Interval extension of f?
replace operations in the definition of f by Interval operations.

Definition (horner_rec l x)

```
match l with
| [::]  => 0
| a::k => x * horner_rec k x + a
end.
```

Definition (horner_rec l x)

```
match l with
| [::]  => 0
| a::k => x * horner_rec k x + a
end.
```

Replacements lead to:

Definition (hornerIA pI I)

```
match pI with
| [::]  => I0
| J::pJ => I.add (I.mul (hornerIA pJ I) I) J
end.
```

Definition (horner_rec l x)

```
match l with
| [::]  => 0
| a::k => x * horner_rec k x + a
end.
```

Replacements lead to:

Definition (hornerIA pI I)

```
match pI with
| [::]  => I0
| J::pJ => I.add (I.mul (hornerIA pJ I) I) J
end.
```

Prove correctness by means of interval enclosures:

Lemma (hornerIA_correct)

```
∀l x lI I:
(∀i, l_i contained_in lI_i) ->
x contained_in I ->
p.[x] contained_in (hornerIA pI I).
```

Current state

Clenshaw algorithm with a proof of correctness.

Current state

Clenshaw algorithm with a proof of correctness.
Working on sin function.

Computable analysis

Represented space (X, δ)

Computable analysis

Represented space (X, δ):
X: set (matematical structure).

Computable analysis

Represented space (X, δ):
X: set (matematical structure).
$\delta :\subseteq \mathcal{B} \to X$ surjective, partial.

Computable analysis

Represented space (X, δ):
X: set (matematical structure).
$\delta :\subseteq \mathcal{B} \to X$ surjective, partial.
φ is called **name** of x if $\delta(\varphi) = x$.

Computable analysis

Represented space (X, δ):
X: set (matematical structure).
$\delta :\subseteq \mathcal{B} \to X$ surjective, partial.
φ is called **name** of x if $\delta(\varphi) = x$.
Usually: $\mathcal{B} := \Sigma^* \to \Sigma^*$

Computable analysis

Represented space (X, δ):
X: set (matematical structure).
$\delta :\subseteq \mathcal{B} \to X$ surjective, partial.
φ is called **name** of x if $\delta(\varphi) = x$.
Usually: $\mathcal{B} := \Sigma^* \to \Sigma^*$ better: $\mathcal{B} := Q \to A$.

Computable analysis

Represented space (X, δ):
X: set (matematical structure).
$\delta :\subseteq \mathcal{B} \to X$ surjective, partial.
φ is called **name** of x if $\delta(\varphi) = x$.
Usually: $\mathcal{B} := \Sigma^* \to \Sigma^*$ better: $\mathcal{B} := Q \to A$. $\varphi\colon Q \to A$.

Computable analysis

Represented space (X, δ):
X: set (matematical structure).
$\delta :\subseteq \mathcal{B} \to X$ surjective, partial.
φ is called **name** of x if $\delta(\varphi) = x$.
Usually: $\mathcal{B} := \Sigma^* \to \Sigma^*$ better: $\mathcal{B} := Q \to A$. $\varphi\colon Q \to A$.
$\rightsquigarrow$ coqrep.

Computable analysis

Represented space (X, δ):
X: set (matematical structure).
$\delta :\subseteq \mathcal{B} \to X$ surjective, partial.
φ is called **name** of x if $\delta(\varphi) = x$.
Usually: $\mathcal{B} := \Sigma^* \to \Sigma^*$ better: $\mathcal{B} := Q \to A$. $\varphi\colon Q \to A$.
$\rightsquigarrow$ coqrep.
Example: only one question.

Computable analysis

Represented space (X, δ):
X: set (matematical structure).
$\delta :\subseteq \mathcal{B} \to X$ surjective, partial.
φ is called **name** of x if $\delta(\varphi) = x$.
Usually: $\mathcal{B} := \Sigma^* \to \Sigma^*$ better: $\mathcal{B} := Q \to A$. $\varphi\colon Q \to A$.
$\rightsquigarrow$ coqrep.
Example: only one question.
Answer is complete description of the object.

Computable analysis

Represented space (X, δ):
X: set (matematical structure).
$\delta :\subseteq \mathcal{B} \to X$ surjective, partial.
φ is called **name** of x if $\delta(\varphi) = x$.
Usually: $\mathcal{B} := \Sigma^* \to \Sigma^*$ better: $\mathcal{B} := Q \to A$. $\varphi\colon Q \to A$.
$\rightsquigarrow$ coqrep.
Example: only one question.
Answer is complete description of the object.

Example

List of rational numbers $l \rightsquigarrow$ polynomial `Poly l`.

Computable analysis

Represented space (X, δ):
X: set (matematical structure).
$\delta :\subseteq \mathcal{B} \to X$ surjective, partial.
φ is called **name** of x if $\delta(\varphi) = x$.
Usually: $\mathcal{B} := \Sigma^* \to \Sigma^*$ better: $\mathcal{B} := Q \to A$. $\varphi\colon Q \to A$.
$\rightsquigarrow$ coqrep.
Example: only one question.
Answer is complete description of the object.

Example

List of rational numbers $l \rightsquigarrow$ polynomial `Poly l`.

More interesting: $\mathbb{R}$.

Two representations of $\mathbb{R}$

Example (The Cauchy rep. δ_C)

Set $Q := \mathbb{Q}$ and $A := \mathbb{Q}$. $\varphi\colon \mathbb{Q} \to \mathbb{Q}$ is δ_C-name of $x \in \mathbb{R}$ iff

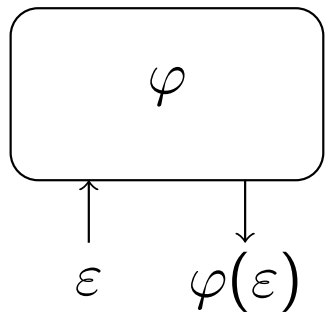
$$\forall \varepsilon > 0\colon |\varphi(\varepsilon) - x| \leq \varepsilon.$$

Two representations of $\mathbb{R}$

Example (The Cauchy rep. δ_C)

Set $Q := \mathbb{Q}$ and $A := \mathbb{Q}$. $\varphi\colon \mathbb{Q} \to \mathbb{Q}$ is δ_C-name of $x \in \mathbb{R}$ iff

$$\forall \varepsilon > 0\colon |\varphi(\varepsilon) - x| \leq \varepsilon.$$



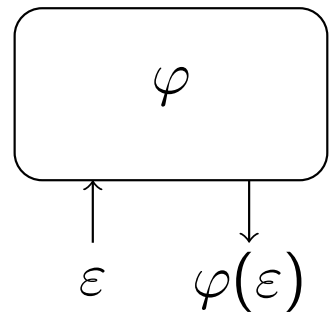
Two representations of $\mathbb{R}$

Example (The Cauchy rep. δ_C)

Set $Q := \mathbb{Q}$ and $A := \mathbb{Q}$. $\varphi\colon \mathbb{Q} \to \mathbb{Q}$ is δ_C-name of $x \in \mathbb{R}$ iff

$$\forall \varepsilon > 0\colon |\varphi(\varepsilon) - x| \leq \varepsilon.$$

x is computable if it has a computable name.



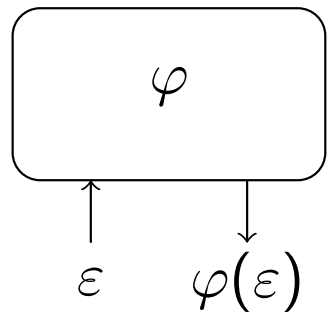
Two representations of $\mathbb{R}$

Example (The Cauchy rep. δ_C)

Set $Q := \mathbb{Q}$ and $A := \mathbb{Q}$. $\varphi\colon \mathbb{Q} \to \mathbb{Q}$ is δ_C-name of $x \in \mathbb{R}$ iff

$$\forall \varepsilon > 0\colon |\varphi(\varepsilon) - x| \leq \varepsilon.$$

x is computable if it has a computable name.



Example (The interval rep. δ_i)

Monotone sequence I_n of intervals is a δ_i-name of $x \in \mathbb{R}$ if

$$\{x\} = \bigcap_n I_n.$$

Two representations of $\mathbb{R}$

Example (The Cauchy rep. δ_C)

Set $Q := \mathbb{Q}$ and $A := \mathbb{Q}$. $\varphi\colon \mathbb{Q} \to \mathbb{Q}$ is δ_C-name of $x \in \mathbb{R}$ iff

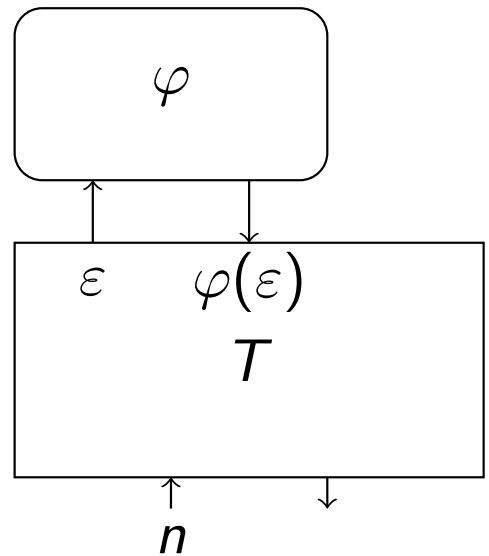
$$\forall \varepsilon > 0\colon |\varphi(\varepsilon) - x| \leq \varepsilon.$$

Example (The interval rep. δ_i)

Monotone sequence I_n of intervals is a δ_i-name of $x \in \mathbb{R}$ if

$$\{x\} = \bigcap_n I_n.$$

x is computable if it has a computable name.



Two representations of $\mathbb{R}$

Example (The Cauchy rep. δ_C)

Set $Q := \mathbb{Q}$ and $A := \mathbb{Q}$. $\varphi\colon \mathbb{Q} \to \mathbb{Q}$ is δ_C-name of $x \in \mathbb{R}$ iff

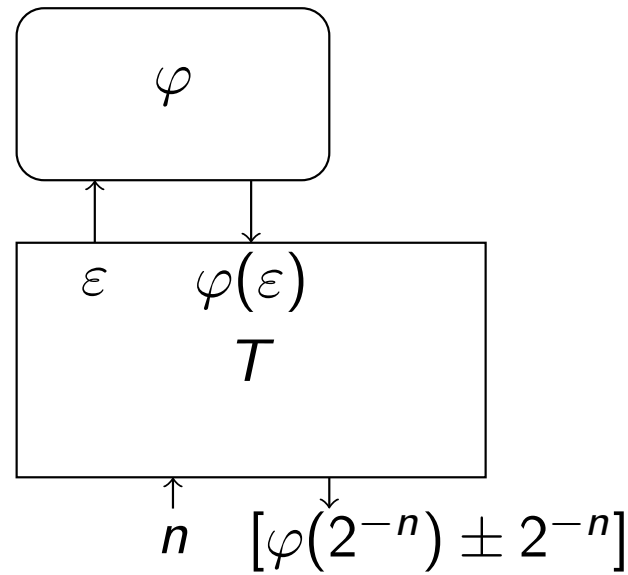
$$\forall \varepsilon > 0\colon |\varphi(\varepsilon) - x| \leq \varepsilon.$$

Example (The interval rep. δ_i)

Monotone sequence I_n of intervals is a δ_i-name of $x \in \mathbb{R}$ if

$$\{x\} = \bigcap_n I_n.$$

x is computable if it has a computable name.



Two representations of $\mathbb{R}$

Example (The Cauchy rep. δ_C)

Set $Q := \mathbb{Q}$ and $A := \mathbb{Q}$. $\varphi\colon \mathbb{Q} \to \mathbb{Q}$ is δ_C-name of $x \in \mathbb{R}$ iff

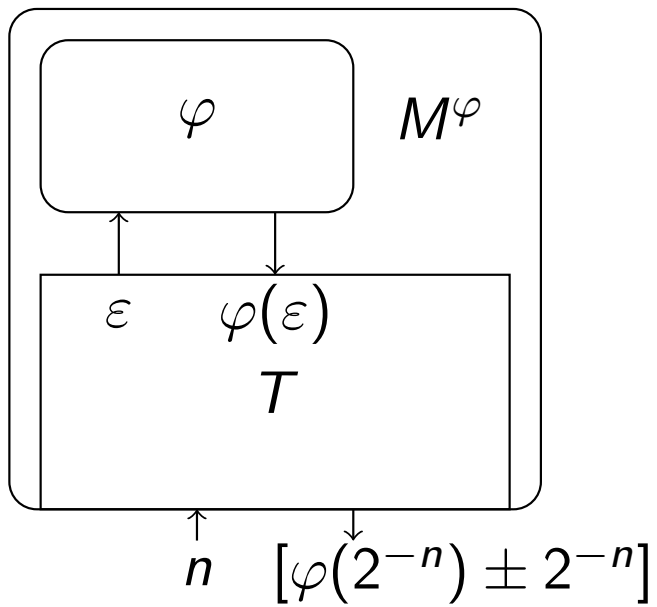
$$\forall \varepsilon > 0\colon |\varphi(\varepsilon) - x| \leq \varepsilon.$$

Example (The interval rep. δ_i)

Monotone sequence I_n of intervals is a δ_i-name of $x \in \mathbb{R}$ if

$$\{x\} = \bigcap_n I_n.$$

x is computable if it has a computable name.



Two representations of $\mathbb{R}$

Example (The Cauchy rep. δ_C)

Set $Q := \mathbb{Q}$ and $A := \mathbb{Q}$. $\varphi : \mathbb{Q} \to \mathbb{Q}$ is δ_C-name of $x \in \mathbb{R}$ iff

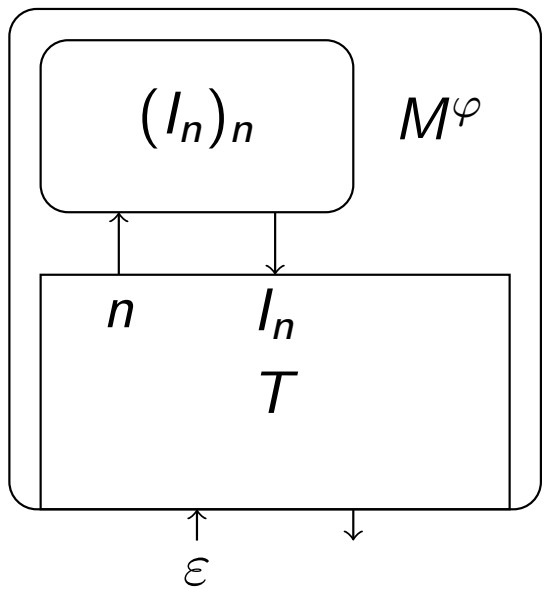
$$\forall \varepsilon > 0 : |\varphi(\varepsilon) - x| \leq \varepsilon.$$

Example (The interval rep. δ_i)

Monotone sequence I_n of intervals is a δ_i-name of $x \in \mathbb{R}$ if

$$\{x\} = \bigcap_n I_n.$$

x is computable if it has a computable name.



Two representations of $\mathbb{R}$

Example (The Cauchy rep. δ_C)

Set $Q := \mathbb{Q}$ and $A := \mathbb{Q}$. $\varphi\colon \mathbb{Q} \to \mathbb{Q}$ is δ_C-name of $x \in \mathbb{R}$ iff

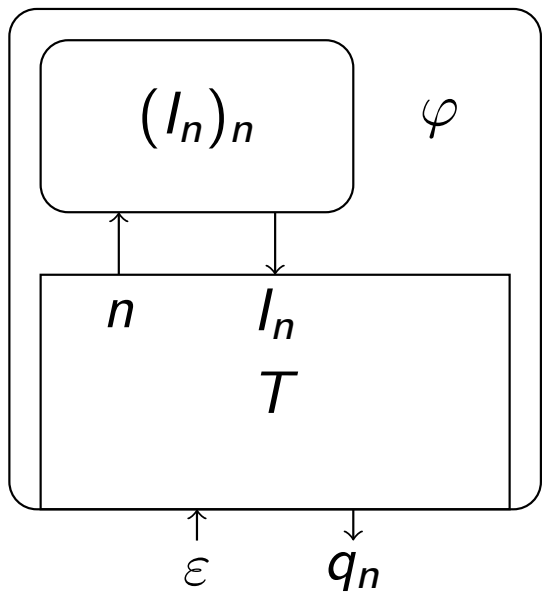
$$\forall \varepsilon > 0\colon |\varphi(\varepsilon) - x| \leq \varepsilon.$$

Example (The interval rep. δ_i)

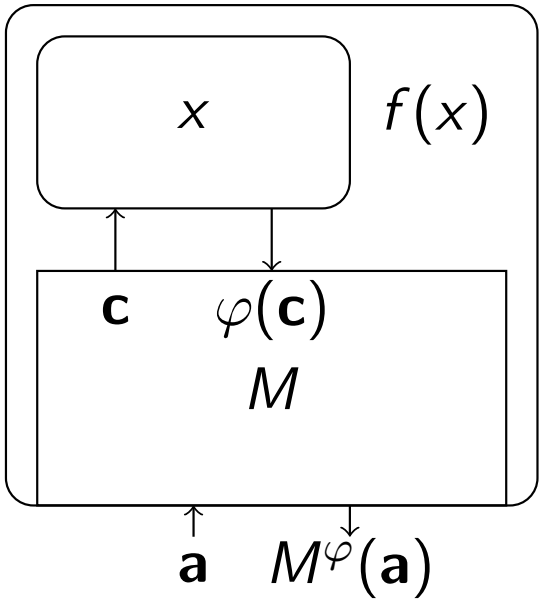
Monotone sequence I_n of intervals is a δ_i-name of $x \in \mathbb{R}$ if

$$\{x\} = \bigcap_n I_n.$$

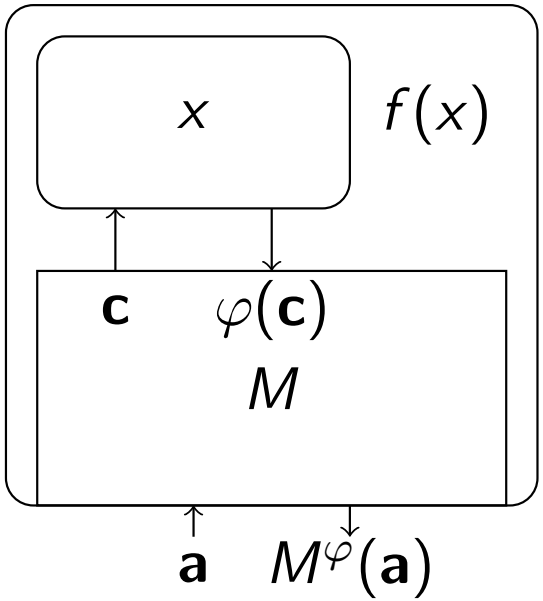
x is computable if it has a computable name.



Functions on the interval reals

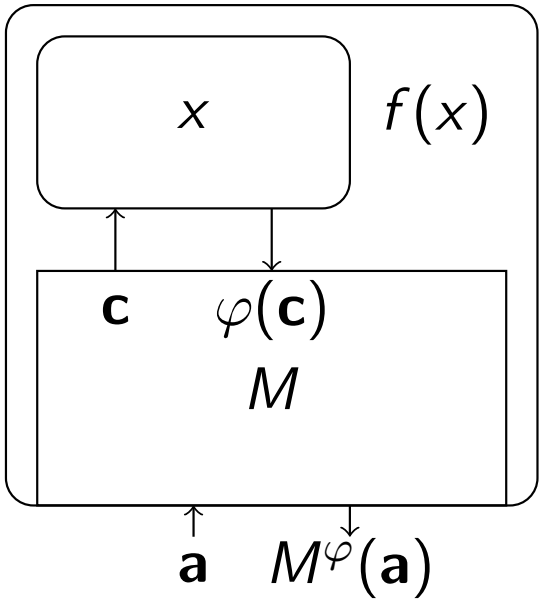


Functions on the interval reals



- M maps names of x to names of $f(x)$.

Functions on the interval reals



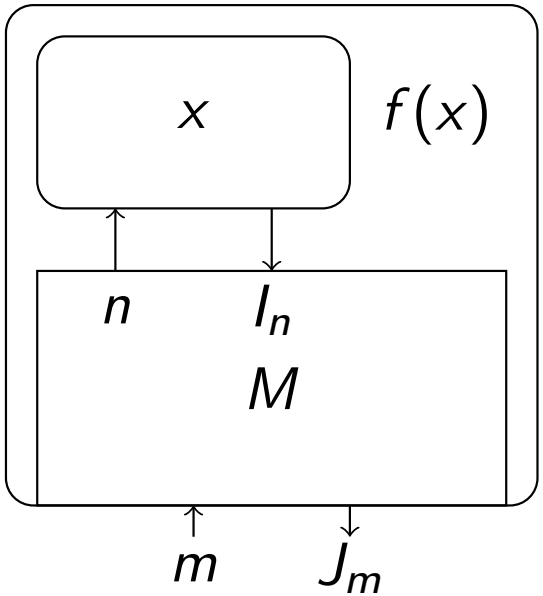
- M maps names of x to names of $f(x)$.

Recall:

Lemma (`ladd_poly_spec`)

```
∀(l k:  seq R), Poly (ladd l k) = Poly l + Poly k.
```

Functions on the interval reals



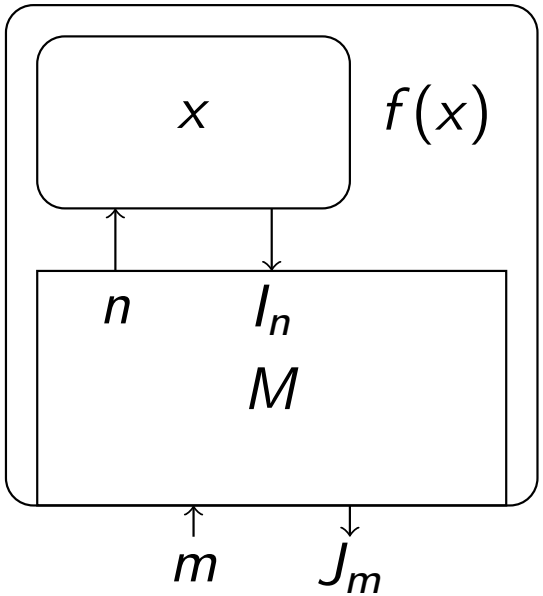
- M maps names of x to names of $f(x)$.

Recall:

Lemma (ladd_poly_spec)

```
∀(l k:  seq R), Poly (ladd l k) = Poly l + Poly k.
```

Functions on the interval reals



- M maps names of x to names of $f(x)$.
- F interval extension of f.

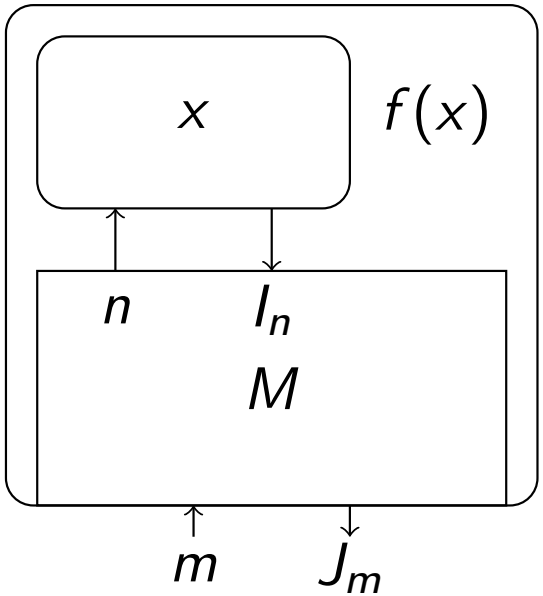
.

Recall:

Lemma (ladd_poly_spec)

```
∀(l k:  seq R), Poly (ladd l k) = Poly l + Poly k.
```

Functions on the interval reals



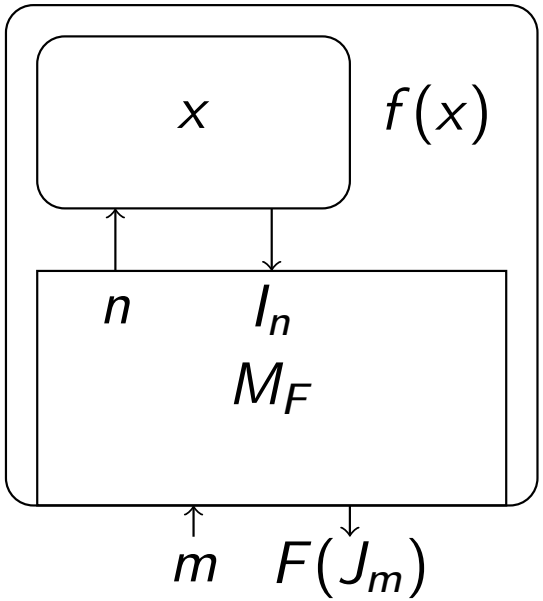
- M maps names of x to names of $f(x)$.
- F interval extension of f.
  Set: $M_F^{\varphi}(\mathbf{a}) := F(\varphi(\mathbf{a}))$.

Recall:

Lemma (ladd_poly_spec)

```
∀(l k:  seq R), Poly (ladd l k) = Poly l + Poly k.
```

Functions on the interval reals



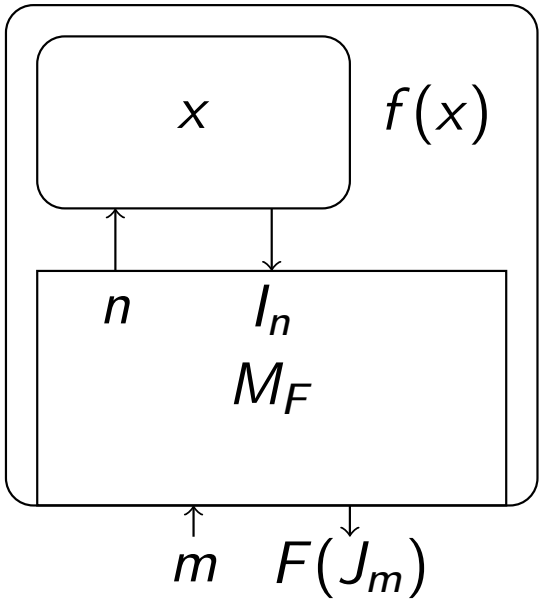
- M maps names of x to names of $f(x)$.
- F interval extension of f.
  Set: $M_F^{\varphi}(\mathbf{a}) := F(\varphi(\mathbf{a}))$.

Recall:

Lemma (ladd_poly_spec)

```
∀(l k:  seq R), Poly (ladd l k) = Poly l + Poly k.
```

Functions on the interval reals



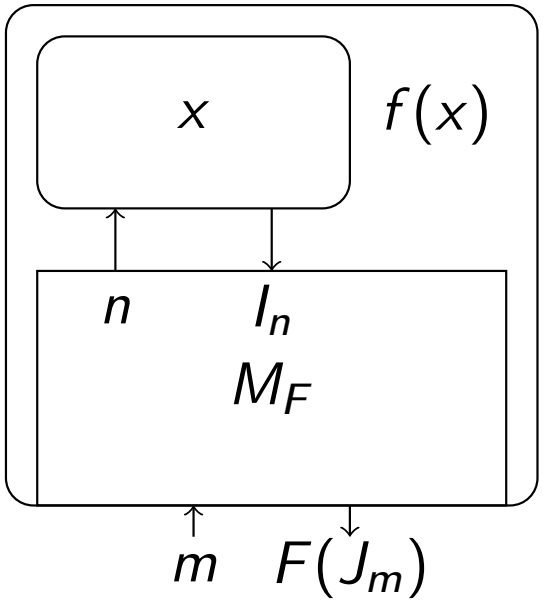
- M maps names of x to names of $f(x)$.
- F interval extension of f.
  Set: $M_F^{\varphi}(\mathbf{a}) := F(\varphi(\mathbf{a}))$.
- Need additional conditions.

Recall:

Lemma (`ladd_poly_spec`)

```
∀(l k:  seq R), Poly (ladd l k) = Poly l + Poly k.
```

Functions on the interval reals



- M maps names of x to names of $f(x)$.
- F interval extension of f.
  Set: $M_F^{\varphi}(\mathbf{a}) := F(\varphi(\mathbf{a}))$.
- Need additional conditions.

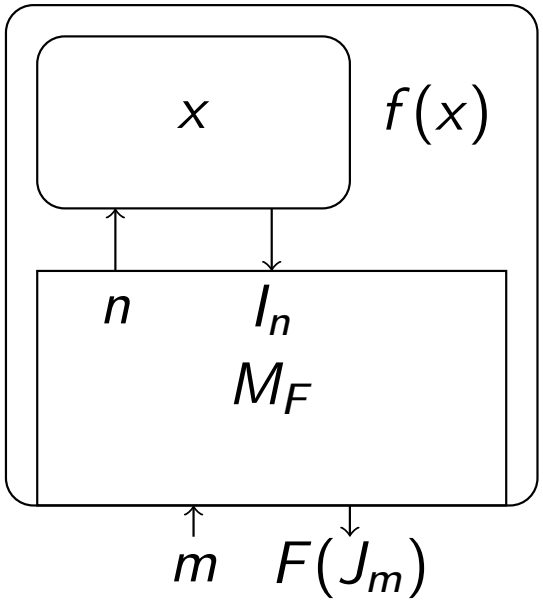
Recall:

Lemma (`ladd_poly_spec`)

```
∀(l k:  seq R), Poly (ladd l k) = Poly l + Poly k.
```

- Idea: use Coq-Interval to prove basic functions computable.

Functions on the interval reals



- M maps names of x to names of $f(x)$.
- F interval extension of f.
  Set: $M_F^{\varphi}(\mathbf{a}) := F(\varphi(\mathbf{a}))$.
- Need additional conditions.

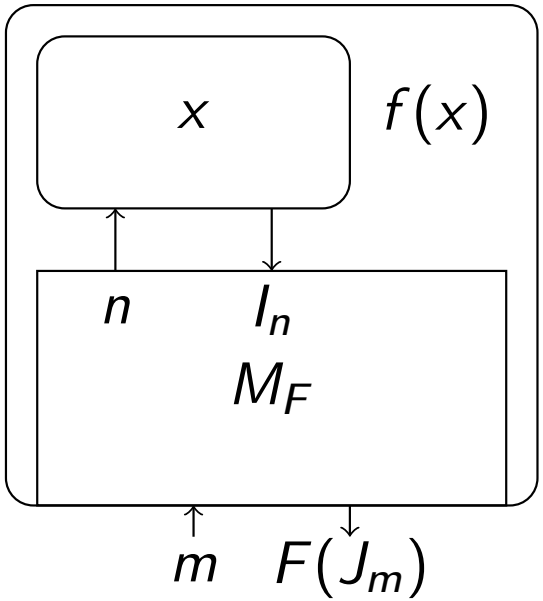
Recall:

Lemma (`ladd_poly_spec`)

```
∀(l k:  seq R), Poly (ladd l k) = Poly l + Poly k.
```

- Idea: use Coq-Interval to prove basic functions computable.
- Need error estimates for Interval computations.

Functions on the interval reals



- M maps names of x to names of $f(x)$.
- F interval extension of f.
  Set: $M_F^{\varphi}(\mathbf{a}) := F(\varphi(\mathbf{a}))$.
- Need additional conditions.

Recall:

Lemma (`ladd_poly_spec`)

```
∀(l k:  seq R), Poly (ladd l k) = Poly l + Poly k.
```

- Idea: use Coq-Interval to prove basic functions computable.
- Need error estimates for Interval computations.
- Computable analysis uses different concept of computability.

Thanks!

Thanks!

On the Github:
`https://github.com/FlorianSteinberg/Cheby`
`https://github.com/FlorianSteinberg/coqrep`