Implementation of Taylor Models in Coq

Érik Martin-Dorel & Ioana Pașca

École Normale Supérieure de Lyon, LIP
46 allée d'Italie, 69364 LYON CEDEX 07, France

erik.martin-dorel@ens-lyon.org
http://perso.ens-lyon.fr/erik.martin-dorel/

Journées CoqApprox
Les 14 & 15 juin 2011
Salle du Conseil du LIP

Outline

Outline

Motivation

- TaMaDi: certification of hardest-to-round points for elementary functions
- Certified polynomial approximation

Goals

- Formal validation
- Fast computations
- Genericity $\Rightarrow$ ease the implantation of new functions

Context: Taylor Model

Definition

A Taylor Model representing a univariate function f on an interval $\boldsymbol{I}$ is a pair $(T, \boldsymbol{\Delta})$ where T is a polynomial and $\boldsymbol{\Delta}$ is an interval bounding the error between f and T on $\boldsymbol{I}$. Roughly speaking, we have:

$$\forall x \in \boldsymbol{I}, \quad f(x) - T(x) \in \boldsymbol{\Delta}.$$

- A typical example of Taylor Model is a degree-n Taylor expansion along with the Taylor-Lagrange remainder
- Which type for the coefficients of T?
  - Exact real numbers?
  - Floating-point numbers?
  - Small floating-point intervals $\Rightarrow$ ensures the true coefficients of the Taylor polynomial lie inside the corresponding intervals

Context: Taylor Model

Definition

A Taylor Model representing a univariate function f on an interval $\boldsymbol{I}$ is a pair $(T, \boldsymbol{\Delta})$ where T is a polynomial and $\boldsymbol{\Delta}$ is an interval bounding the error between f and T on $\boldsymbol{I}$. Roughly speaking, we have:

$$\forall x \in \boldsymbol{I}, \quad f(x) - T(x) \in \boldsymbol{\Delta}.$$

- A typical example of Taylor Model is a degree-n Taylor expansion along with the Taylor-Lagrange remainder
- Which type for the coefficients of T?
  - Exact real numbers?
  - Floating-point numbers?
  - Small floating-point intervals $\Rightarrow$ ensures the true coefficients of the Taylor polynomial lie inside the corresponding intervals

Context: Taylor Model

Definition

A Taylor Model representing a univariate function f on an interval $\boldsymbol{I}$ is a pair $(T, \boldsymbol{\Delta})$ where T is a polynomial and $\boldsymbol{\Delta}$ is an interval bounding the error between f and T on $\boldsymbol{I}$. Roughly speaking, we have:

$$\forall x \in \boldsymbol{I}, \quad f(x) - T(x) \in \boldsymbol{\Delta}.$$

- A typical example of Taylor Model is a degree-n Taylor expansion along with the Taylor-Lagrange remainder
- Which type for the coefficients of T?
  - Exact real numbers?
  - Floating-point numbers?
  - Small floating-point intervals $\Rightarrow$ ensures the true coefficients of the Taylor polynomial lie inside the corresponding intervals

Context: Taylor Model

Definition

A Taylor Model representing a univariate function f on an interval $\boldsymbol{I}$ is a pair $(T, \boldsymbol{\Delta})$ where T is a polynomial and $\boldsymbol{\Delta}$ is an interval bounding the error between f and T on $\boldsymbol{I}$. Roughly speaking, we have:

$$\forall x \in \boldsymbol{I}, \quad f(x) - T(x) \in \boldsymbol{\Delta}.$$

- A typical example of Taylor Model is a degree-n Taylor expansion along with the Taylor-Lagrange remainder
- Which type for the coefficients of T?
  - Exact real numbers?
  - Floating-point numbers?
  - Small floating-point intervals $\Rightarrow$ ensures the true coefficients of the Taylor polynomial lie inside the corresponding intervals

Context: Taylor Model

Definition

A Taylor Model representing a univariate function f on an interval $\boldsymbol{I}$ is a pair $(T, \boldsymbol{\Delta})$ where T is a polynomial and $\boldsymbol{\Delta}$ is an interval bounding the error between f and T on $\boldsymbol{I}$. Roughly speaking, we have:

$$\forall x \in \boldsymbol{I}, \quad f(x) - T(x) \in \boldsymbol{\Delta}.$$

- A typical example of Taylor Model is a degree-n Taylor expansion along with the Taylor-Lagrange remainder
- Which type for the coefficients of T?
    - Exact real numbers?
    - Floating-point numbers?
    - Small floating-point intervals $\Rightarrow$ ensures the true coefficients of the Taylor polynomial lie inside the corresponding intervals

Context: Taylor Model

Definition

A Taylor Model representing a univariate function f on an interval $\boldsymbol{I}$ is a pair $(T, \boldsymbol{\Delta})$ where T is a polynomial and $\boldsymbol{\Delta}$ is an interval bounding the error between f and T on $\boldsymbol{I}$. Roughly speaking, we have:

$$\forall x \in \boldsymbol{I}, \quad f(x) - T(x) \in \boldsymbol{\Delta}.$$

- A typical example of Taylor Model is a degree-n Taylor expansion along with the Taylor-Lagrange remainder
- Which type for the coefficients of T?
  - Exact real numbers?
  - Floating-point numbers?
  - Small floating-point intervals $\Rightarrow$ ensures the true coefficients of the Taylor polynomial lie inside the corresponding intervals

Context: Taylor Model

Definition

A Taylor Model representing a univariate function f on an interval $\boldsymbol{I}$ is a pair $(T, \boldsymbol{\Delta})$ where T is a polynomial and $\boldsymbol{\Delta}$ is an interval bounding the error between f and T on $\boldsymbol{I}$. Roughly speaking, we have:

$$\forall x \in \boldsymbol{I}, \quad f(x) - T(x) \in \boldsymbol{\Delta}.$$

- A typical example of Taylor Model is a degree-n Taylor expansion along with the Taylor-Lagrange remainder
- Which type for the coefficients of T?
  - Exact real numbers?
  - Floating-point numbers?
  - Small floating-point intervals $\Rightarrow$ ensures the true coefficients of the Taylor polynomial lie inside the corresponding intervals

Context: Taylor Models for D-finite functions

- A D-finite (also called holonomic) function is a function that satisfies a *linear ordinary differential equation with polynomial coefficients* (cf. presentation by Nicolas)
- Most of elementary functions are D-finite
- Consequently, the coefficients of their Taylor expansion satisfy a recurrence relation
- The Taylor-Lagrange remainder has a factor that is very similar to a bare Taylor coefficient, except that x_0 is replaced with the working interval I
- Consequently, computation by recurrence combined with Interval Arithmetic (IA) appears to be an attractive way to provide certified polynomial approximation

Context: Taylor Models for D-finite functions

- A D-finite (also called holonomic) function is a function that satisfies a *linear ordinary differential equation with polynomial coefficients* (cf. presentation by Nicolas)
- Most of elementary functions are D-finite
- Consequently, the coefficients of their Taylor expansion satisfy a recurrence relation
- The Taylor-Lagrange remainder has a factor that is very similar to a bare Taylor coefficient, except that x_0 is replaced with the working interval I
- Consequently, computation by recurrence combined with Interval Arithmetic (IA) appears to be an attractive way to provide certified polynomial approximation

Context: Taylor Models for D-finite functions

- A D-finite (also called holonomic) function is a function that satisfies a *linear ordinary differential equation with polynomial coefficients* (cf. presentation by Nicolas)
- Most of elementary functions are D-finite
- Consequently, the coefficients of their Taylor expansion satisfy a recurrence relation
- The Taylor-Lagrange remainder has a factor that is very similar to a bare Taylor coefficient, except that x_0 is replaced with the working interval I
- Consequently, computation by recurrence combined with Interval Arithmetic (IA) appears to be an attractive way to provide certified polynomial approximation

Context: Taylor Models for D-finite functions

- A D-finite (also called holonomic) function is a function that satisfies a *linear ordinary differential equation with polynomial coefficients* (cf. presentation by Nicolas)
- Most of elementary functions are D-finite
- Consequently, the coefficients of their Taylor expansion satisfy a recurrence relation
- The Taylor-Lagrange remainder has a factor that is very similar to a bare Taylor coefficient, except that $\boldsymbol{x_0}$ is replaced with the working interval $\boldsymbol{I}$
- Consequently, computation by recurrence combined with Interval Arithmetic (IA) appears to be an attractive way to provide certified polynomial approximation

Context: Taylor Models for D-finite functions

- A D-finite (also called holonomic) function is a function that satisfies a *linear ordinary differential equation with polynomial coefficients* (cf. presentation by Nicolas)
- Most of elementary functions are D-finite
- Consequently, the coefficients of their Taylor expansion satisfy a recurrence relation
- The Taylor-Lagrange remainder has a factor that is very similar to a bare Taylor coefficient, except that $\boldsymbol{x_0}$ is replaced with the working interval $\boldsymbol{I}$
- Consequently, computation by recurrence combined with Interval Arithmetic (IA) appears to be an attractive way to provide certified polynomial approximation

Need for computation within the proof assistant

- Given the characteristics of the problem, it would be irrelevant to perform the calculations at stake with an external oracle before verifying them in Coq
- There is actually no "characteristic property" or any such formula that could "summarize" the series of IA calculations that generate the Taylor coefficients and the remainder

$\Rightarrow$ We need to compute all these quantities within the proof assistant

Need for computation within the proof assistant

- Given the characteristics of the problem, it would be irrelevant to perform the calculations at stake with an external oracle before verifying them in Coq
- There is actually no "characteristic property" or any such formula that could "summarize" the series of IA calculations that generate the Taylor coefficients and the remainder

$\Rightarrow$ We need to compute all these quantities within the proof assistant

Need for computation within the proof assistant

- Given the characteristics of the problem, it would be irrelevant to perform the calculations at stake with an external oracle before verifying them in Coq
- There is actually no "characteristic property" or any such formula that could "summarize" the series of IA calculations that generate the Taylor coefficients and the remainder

$\Rightarrow$ We need to compute all these quantities within the proof assistant

Taylor Model Structure

Definition

We will call a degree-n Taylor Model Structure a pair $(\boldsymbol{T}, \boldsymbol{\Delta})$ where $\boldsymbol{T} = (\boldsymbol{a_0}, \dots, \boldsymbol{a_n})$ is a list of $n+1$ interval coefficients, and $\boldsymbol{\Delta}$ an interval.

```
Structure tms (deg : nat) : Type := TMS {
  approx : seq I.type ;
  approx_len : size approx = deg.+1 ;
  error : I.type
}.
```

Outline

A generic approach based on recurrences

- We focus on functions whose Taylor coefficients $(u_n)_{n\in\mathbb{N}}$ satisfy a recurrence of the form:

$$\forall n \geqslant N, \quad u_n = U\left(u_{n-N}, u_{n-N+1}, \ldots, u_{n-1}\right),$$

  for a given N-ary function $U : T^N \to T$,
  with the first terms $(u_0, u_1, \ldots, u_{N-1})$ given by a list $L_0 \in T^N$

- We want to define and compute such recurrences in Coq, in an efficient and convenient way

A generic approach based on recurrences

- We focus on functions whose Taylor coefficients $(u_n)_{n\in\mathbb{N}}$ satisfy a recurrence of the form:

$$\forall n \geqslant N, \quad u_n = U\left(u_{n-N}, u_{n-N+1}, \ldots, u_{n-1}\right),$$

for a given N-ary function $U : T^N \to T$,
with the first terms $(u_0, u_1, \ldots, u_{N-1})$ given by a list $L_0 \in T^N$

- We want to define and compute such recurrences in Coq, in an efficient and convenient way

Standard Library Coq.Numbers.NaryFunctions (1/2)

```
Fixpoint nfun A n B :=
 match n with
  | O => B
  | S n => A -> (nfun A n B)
 end.

Notation "A ^^ n --> B" := (nfun A n B) : type_scope.

Fixpoint nprod A n : Type :=
 match n with
  | O => unit
  | S n => (A * nprod A n)%type
 end.

Notation "A ^ n" := (nprod A n) : type_scope.
```

Standard Library Coq.Numbers.NaryFunctions (2/2)

```
Fixpoint ncurry (A B:Type) n : (A^n -> B) -> (A^^n-->B).

Fixpoint nuncurry (A B:Type) n : (A^^n-->B) -> (A^n -> B).

Fixpoint nprod_to_list (A:Type) n : A^n -> list A :=
 match n with
  | O => fun _ => nil
  | S n => fun p => let (x,p):=p in x::(nprod_to_list _ n p)
 end.

Fixpoint nprod_of_list (A:Type) (l:list A) : A^(length l) :=
 match l return A^(length l) with
  | nil => tt
  | x::l => (x, nprod_of_list _ l)
 end.
```

Standard Library Coq.Numbers.NaryFunctions (2/2)

```
Fixpoint ncurry (A B:Type) n : (A^n -> B) -> (A^^n-->B).

Fixpoint nuncurry (A B:Type) n : (A^^n-->B) -> (A^n -> B).

Fixpoint nprod_to_seq (A:Type) n : A^n -> seq A :=
 match n with
  | O => fun _ => nil
  | S n => fun p => let (x,p):=p in x::(nprod_to_seq _ n p)
 end.

Fixpoint nprod_of_seq (A:Type) (l:seq A) : A^(size l) :=
 match l return A^(size l) with
  | nil => tt
  | x::l => (x, nprod_of_seq _ l)
 end.
```

ssrNaryRec : A generic theory for N-ary recurrences

```
Section GenericNaryRec.
Variable T : Type.
Variable L0 : seq T.
Local Notation N := (size L0).
Variable U : T^^N --> (nat -> T).
Definition Uprod := nuncurry T (nat -> T) N U.
Definition URec (k : nat) : seq T.

Lemma URec_nth_indep :
  forall (d : T) m n, (m < n) ->
  nth d (URec L0 U n) m = nth d (URec L0 U m.+1) m.
Lemma URec_correct :
  forall d e m, (N <= m) ->
  nth d (URec L0 U m.+1) m =
  Uprod (nprod_seq_dflt N (m-N) (URec L0 U m) e) m.
End GenericNaryRec.
```

Example of use: the Fibonacci sequence

Thanks to the definitions presented *supra*, we can define this typical recurrence in a very simple way:

```
Definition L0_fib := (1 :: 1 :: nil)%Z.
Definition U_fib := fun p q (_ : nat) => (p + q)%Z.
Definition fib := URec L0_fib U_fib.
```

Note that a naive definition such as the following is not particularly easier to write:

```
Fixpoint fib_naif (n : nat) : Z :=
  match n with
  | O => 1%Z
  | S O => 1%Z
  | S (S p as q) => (fib_naif p + fib_naif q)%Z
  end.
```

and above all, it would lead to a much worse complexity, due to the intrinsic redundancy of the recursion in this definition

TMExp

```
Section TMExp.
Variable prec : F.precision.

Definition Uexp u n :=
  I.I.div_mixed_r prec u (F.fromZ (Z_of_nat n)).
Definition L0exp J := (I.exp prec J) :: nil.
Definition TCexp J n := URec (L0exp J) Uexp (S n).
Lemma TCexp_len : forall J n, size (TCexp J n) = S n.
Definition TMExp (n : nat) X X0 : tms n :=
  TMS (TCexp X0 n) (TCexp_len X0 n) (Trem prec TCexp n X X0).

End TMExp.
```

TMInv

```
Section TMInv.
Variable prec: F.precision.

Definition Uinv u (J : I.type) (n : nat) :=
  I.I.div prec u J.
Definition L0inv J := (I.inv prec J) :: nil.
Definition TCinv J n := URec (L0inv J) Uinv (I.neg J) (S n).
Lemma TCinv_len : forall J n, size (TCinv J n) = S n.
Definition TMInv (n : nat) X0 X : tms n :=
  TMS (TCinv X0 n) (TCinv_len X0 n) (Trem prec TCinv n X X0).

End TMInv.
```

TMSin

```
Section TMsin.
Variable prec : F.precision.

Definition Usin (u v : I.type) n :=
  I.neg (I.I.div_mixed_r prec
    u (F.fromZ (Z_of_nat (predn n) * Z_of_nat n))).
Definition L0sin J := (I.sin prec J :: I.cos prec J :: nil).
Definition TCsin J n := URec (L0sin J) Usin (S n).
Lemma TCsin_len : forall J n, size (TCsin J n) = S n.
Definition TMSin (n : nat) X X0 : tms n :=
  TMS (TCsin X0 n) (TCsin_len X0 n) (Trem prec TCsin n X X0).

End TMsin.
```

TMArctan

Definition

$$\begin{cases} \arctan(0) = 0 \\ \dfrac{\partial \arctan}{\partial x}(x) = \dfrac{1}{1+x^2} \quad \forall x \in \mathbb{R} \end{cases}$$

[Demonstration using Gfun, Coq-Interval & ssrNaryRec]

TMArctan

Definition

$$\begin{cases} \arctan(0) = 0 \\ \dfrac{\partial \arctan}{\partial x}(x) = \dfrac{1}{1+x^2} \quad \forall x \in \mathbb{R} \end{cases}$$

[Demonstration using Gfun, Coq-Interval & ssrNaryRec]

Computation of a polynomial bound

Horner-like evaluation of a polynomial with small interval coefficients:

```
Fixpoint ipoly_eval p x :=
  match p with
  | nil => I.fromZ 0
  | c :: p' => I.add prec (I.mul prec (ipoly_eval p' x) x) c
  end.
```

Other handy functions

```
Definition Irnd prec xi :=
  match xi with
  | Inan => Inan
  | Ibnd xl xu => Ibnd (F.round rnd_DN prec xl) (F.round rnd_UP prec xu)
  end.
```

⇒ Useful to have an idea of the magnitude of a result, with `(Irnd 0 result)`

```
Definition Idiam c := (I.sub prec c c).
Fixpoint map_poly_diam p :=
  match p with
  | nil => nil
  | c :: p' => Idiam c :: map_poly_diam p'
  end.
Definition error_on_poly :=
  ipoly_eval prec (map_poly_diam prec (approx tm)) (I.sub prec X X0).
Definition total_error := (I.add prec error_on_poly (error tm)).
```

⇒ Useful to calculate the error that takes into account the errors on the coefficients

Outline

TMAdd

```
Section Map2.
Variables (A : Type) (B : Type) (C : Type).
Variable f : A -> B -> C.
Fixpoint map2 (l1 : seq A) (l2 : seq B) : seq C :=
  match l1, l2 with
  | a :: l3, b :: l4 => f a b :: map2 l3 l4
  | _, _ => nil
  end.
Lemma map2_len :
  forall l1 l2, size (map2 l1 l2) = minn (size l1) (size l2).
End Map2.

Section TMAddition.
Lemma TMAdd_aux : forall n (Mf Mg : tms n),
  size (map2 (I.add prec) (approx Mf) (approx Mg)) = n.+1.
Definition TMAdd (n : nat) (Mf Mg : tms n) : tms n :=
  TMS (map2 (I.add prec) (approx Mf) (approx Mg)) (TMAdd_aux Mf Mg)
      (I.add prec (error Mf) (error Mg)).
End TMAddition.
```

TMMul

```
Section TMMultiplication.
Variable prec : F.precision.
Local Notation Izero := (I.fromZ 0). Local Notation inth p j := (nth Izero p j).
Definition mul_coeff (p q : seq (I.type)) (n : nat) : I.type :=
  foldr (fun i x => I.add prec (I.mul prec (inth p i) (inth q (n - i))) x)
  Izero (iota 0 n.+1).
Definition mul_seq p q n := mkseq (mul_coeff p q) n.+1.
Lemma size_TMMul :
  forall n (Mf Mg : tms n), size (mul_seq (approx Mf) (approx Mg) n) = n.+1.
Definition seq_of_ds pf pg n :=
  nseq n.+1 Izero ++ map (mul_coeff pf pg) (iota (n.+1) n).
Definition mul_error n (f g : tms n) X0 I :=
  let pf := (approx f) in  let pg := (approx g) in
  let B  := ipoly_eval prec (seq_of_ds pf pg n) (I.sub prec I X0) in
  let Bf := ipoly_eval prec pf (I.sub prec I X0) in
  let Bg := ipoly_eval prec pf (I.sub prec I X0) in
  I.add prec B (I.add prec (I.mul prec (error f) Bg)
  (I.add prec (I.mul prec (error g) Bf) (I.mul prec (error f) (error g)))).
Definition TMMul n (Mf Mg : tms n) X0 I : tms n :=
  TMS (mul_seq (approx Mf) (approx Mg) n) (size_TMMul Mf Mg)
      (mul_error Mf Mg X0 I).
End TMMultiplication.
```

TMComp

```
Section TMComposition. Variable prec : F.precision.
Fixpoint poly_eval_tm_aux n rl (Mf M : tms n) X0 I : tms n :=
  match rl with | [::] => M
                | t :: rl' => let M1 := TMMul prec M Mf X0 I in
                              let M2 := TMAdd prec M1 (TMConst n t) in
                              poly_eval_tm_aux rl' Mf M2 X0 I  end.
Definition poly_eval_tm n rl Mf X0 I :=
  poly_eval_tm_aux rl Mf (TMConst n Izero) X0 I.
Inductive fBase := fSin | fCos | fExp | fAtan | fInv.
Definition switchBase (f : fBase) (n : nat) (X0 I : I.type) : tms n.
Definition replace1 T (p : seq T) t :=
  match p with [::] => [::] | _ :: l => t ::l end.
Lemma size_tm_replace1 :
  forall n (Mf : tms n) t, size (replace1 (approx Mf) t) = n.+1.
Definition TMreplace1 n (Mf :tms n) t :=
  TMS (replace1 (approx Mf) t) (size_tm_replace1 Mf t) (error Mf).
Definition TMComp n (Mf : tms n) g X0 I :=
  let Bf := ipoly_eval prec (approx Mf) (I.sub prec I X0) in
  let Mg := switchBase g n (inth (approx Mf) 0) (I.add prec Bf (error Mf)) in
  let M1 := TMreplace1 Mf Izero in
  let rMg := (rev (approx Mg)) in  let M0 := poly_eval_tm rMg M1 X0 I in
  TMS (approx M0) (approx_len M0) (I.add prec (error M0) (error Mg)).
End TMComposition.
```

TMBaseDiv

We use the following fact:

$$\frac{f}{g} = f \times \left[\left(x \mapsto \frac{1}{x}\right) \circ g\right]$$

```
Section TMBaseDivision.
Variable prec : F.precision.
Definition TMBaseDiv n (Mf Mg : tms n) X0 I :=
  TMMul prec Mf (TMComp prec Mg fInv X0 I) X0 I.
End TMBaseDivision.
```

Summary of the algebraic operations on Taylor Models

TMOp	Complexity in terms of the degree n
TMAdd	$O(n)$
TMMul	$O(n^2)$
TMComp	$O(n^3)$
TMBaseDiv	$O(n^3)$

Outline

Validity of a Taylor Model

Definition

$M = (\boldsymbol{a_0}, \ldots, \boldsymbol{a_n}, \boldsymbol{\Delta})$ is a valid Taylor Model of $f : \mathbb{R} \to \mathbb{R}$ in $\boldsymbol{x_0}$ over $\boldsymbol{I}$ if $\boldsymbol{x_0} \subset \boldsymbol{I}$, $0 \in \boldsymbol{\Delta}$, and

$$\forall \xi_0 \in \boldsymbol{x_0},\ \exists \alpha_0 \in \boldsymbol{a_0}, \ldots, \alpha_n \in \boldsymbol{a_n},\ \forall x \in \boldsymbol{I},\ f(x) - \sum_{i=0}^{n} \alpha_i \left(x - \xi_0\right)^i \in \boldsymbol{\Delta}.$$

```
Definition validTM n (X X0 : I.type)
    (tm : tms n) (f : ExtendedR -> ExtendedR) :=
  forall fi0, contains (I.convert X0) fi0 ->
  exists alf, size alf = S n /\ ( forall k, (k <= n) ->
    contains (I.convert (inth (approx tm) k)) (xnth alf k) ) /\
  forall x, contains (I.convert X) x ->
  contains (I.convert (error tm))
           (Xsub (f x) (xpoly_eval alf (Xsub x fi0))).
```

Proofs of correctness

Two main types of proofs of correctness:

- Correctness of building blocks:

```
Lemma TMFun_valid : forall n (X X0 : I.type),
  validTM X X0 (TMFun prec n X X0) Xfun.
```

  for a Taylor Model TMFun representing a base function Xfun

- Correctness of algebraic operations:

```
Lemma TMOp_valid :
  forall n (X X0 : I.type) (TMf TMg : tms n) f g,
  validTM X X0 TMf f ->
  validTM X X0 TMg g ->
  validTM X X0 (TMOp prec TMf TMg)
                (fun xr => Xop (f xr) (g xr)).
```

  for an algebraic rule TMOp related to the considered operation Xop, allowing to build composite Taylor Models in a recursive way

Summary of the Proofs

TMFun/TMOp	implemented	proved
TMExp	☒	☐[1]
TMSin	☒[2]	☐
TMCos	☒[2]	☐
TMTan	☐	☐
TMArctan	☒	☐
TMConst	☒	☒
TMVar	☒	☐
TMInv	☒	☐
TMAdd	☒	☒
TMMul	☒	☐
TMComp	☒	☐
TMBaseDiv	☒	☐

[1] A key subgoal related to the Taylor-Lagrange theorem remains to prove

[2] Coq-Interval currently provides `I.sin` and `I.cos` on a limited range

Outline

Conclusion and Perspectives

- Goal: Certified Polynomial Approximation
- A recurrence-based generic approach

- Add more functions
- Prove all fonctions

- Implement more tight remainders
- Consider Chebyshev Models

End of the talk

- Many thanks for your attention
- Any question?