Formal Proofs for Taylor Models in COQ

Ioana Paşca

AriC, LIP - ENS Lyon

CaCos, 26 July 2012

A big team

Érik Martin-Dorel, Micaela Mayero, Ioana Pașca, Laurence Rideau, Laurent Théry

Nicolas Brisebarre, Mioara Joldeș, Jean-Michel Muller

ANR project TaMaDi - Table Maker's Dilemma

Goal: certified polynomial approximation of real functions

Consider a function f, a polynomial P, an error δ and an interval I
Show: $\forall x \in I, |f(x) - P(x)| < \delta$

Goal: certified polynomial approximation of real functions

Consider a function f, a polynomial P, an error δ and an interval I
Show: $\forall x \in I, |f(x) - P(x)| < \delta$

- dedicated data structure: rigorous polynomial approximation
  - a pair (P, Δ) of a polynomial and an interval that contains the approximation error

Goal: certified polynomial approximation of real functions

Consider a function f, a polynomial P, an error δ and an interval I
Show: $\forall x \in I, |f(x) - P(x)| < \delta$

- dedicated data structure: rigorous polynomial approximation
  - a pair (P, Δ) of a polynomial and an interval that contains the approximation error
- Taylor models (TM)

Goal: certified polynomial approximation of real functions

Consider a function f, a polynomial P, an error δ and an interval I
Show: $\forall x \in I, |f(x) - P(x)| < \delta$

- dedicated data structure: rigorous polynomial approximation
  - a pair (P, Δ) of a polynomial and an interval that contains the approximation error
- Taylor models (TM)
- Formal verification
  - ensure correctness of the TM algorithms
  - ensure correct computation of TMs
  - by using a proof assistant

Interval Arithmetic

- interval = pair of representable numbers
- e.g., $\pi \in [3.14, 3.15]$
- operations and functions on intervals
  $[2,4] - [0,1] = [1,4]$ $\qquad$ $\mathbf{Exp}([0,1]) = [1, 2.72]$
- satisfy the enclosure property
  $\forall x \in [0,1], \exp(x) \in \mathbf{Exp}([0,1]) = [1, 2.72]$

Interval Arithmetic

- interval = pair of representable numbers
- e.g., $\pi \in [3.14, 3.15]$
- operations and functions on intervals
  $[2,4] - [0,1] = [1,4]$ $\qquad$ $\mathbf{Exp}([0,1]) = [1, 2.72]$
- satisfy the enclosure property
  $\forall x \in [0,1], \exp(x) \in \mathbf{Exp}([0,1]) = [1, 2.72]$
- tool for bounding the range of functions
- dependency problem:
  e.g. $F(x) := x - x$
  in interval arithmetic $F([1,4]) = [-4,4]$ while we expect $[0,0]$

Interval Arithmetic

- interval = pair of representable numbers
- e.g., $\pi \in [3.14, 3.15]$
- operations and functions on intervals
  $[2,4] - [0,1] = [1,4]$ $\quad$ $\mathbf{Exp}([0,1]) = [1, 2.72]$
- satisfy the enclosure property
  $\forall x \in [0,1], \exp(x) \in \mathbf{Exp}([0,1]) = [1, 2.72]$
- tool for bounding the range of functions
- dependency problem:
  e.g. $F(x) := x - x$
  in interval arithmetic $F([1,4]) = [-4,4]$ while we expect $[0,0]$
- interval arithmetic is not directly applicable to bound the approximation error $e := P - f$ as the values of f and P are very near

Outline

1. Algorithms for Taylor Models
2. Formalization of Taylor Models in COQ
3. Current Results and Future Developments

Taylor Models

Definition

An order-n Taylor Model (TM) for a function $f : D \subset \mathbb{R} \to \mathbb{R}$ over $\boldsymbol{I}$ is a pair $(T, \boldsymbol{\Delta})$ where T is a degree-n polynomial and $\boldsymbol{\Delta}$ is an interval, such that

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

Taylor Models

Definition

An order-n Taylor Model (TM) for a function $f : D \subset \mathbb{R} \to \mathbb{R}$ over $\boldsymbol{I}$ is a pair $(T, \boldsymbol{\Delta})$ where T is a degree-n polynomial and $\boldsymbol{\Delta}$ is an interval, such that

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

But what type for T?

Taylor Models

Definition

An order-n Taylor Model (TM) for a function $f : D \subset \mathbb{R} \to \mathbb{R}$ over $\boldsymbol{I}$ is a pair $(T, \boldsymbol{\Delta})$ where T is a degree-n polynomial and $\boldsymbol{\Delta}$ is an interval, such that

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

But what type for T?

Polynomial T with interval coefficients

- rounding errors are directly handled by the interval arithmetic

Theorem (Taylor-Lagrange)

If f is $n+1$ times derivable on $\boldsymbol{I}$, then $\forall x \in \boldsymbol{I}$, $\exists c$ between x_0 and x s.t.:

$$f(x) = \underbrace{\left(\sum_{i=0}^{n} \frac{f^{(i)}(x_0)}{i!} (x - x_0)^i \right)}_{\textit{Taylor expansion}} + \underbrace{\frac{f^{(n+1)}(c)}{(n+1)!} (x - x_0)^{n+1}}_{\Delta(x,c)}.$$

Computation

- for T: compute interval enclosures of $\frac{f^{(i)}(x_0)}{i!}$, $i = 0, \ldots, n$
- for $\boldsymbol{\Delta}$: compute in interval arithmetic $\frac{f^{(n+1)}(\boldsymbol{I})}{(n+1)!}(\boldsymbol{I} - x_0)^{n+1}$

Theorem (Taylor-Lagrange)

If f is $n+1$ times derivable on $\boldsymbol{I}$, then $\forall x \in \boldsymbol{I}$, $\exists c$ between x_0 and x s.t.:

$$f(x) = \underbrace{\left(\sum_{i=0}^{n} \frac{f^{(i)}(x_0)}{i!} (x - x_0)^i \right)}_{\textit{Taylor expansion}} + \underbrace{\frac{f^{(n+1)}(c)}{(n+1)!} (x - x_0)^{n+1}}_{\Delta(x,c)}.$$

Computation

- for T: compute interval enclosures of $\frac{f^{(i)}(x_0)}{i!}$, $i = 0, \ldots, n$
- for $\boldsymbol{\Delta}$: compute in interval arithmetic $\frac{f^{(n+1)}(\boldsymbol{I})}{(n+1)!}(\boldsymbol{I} - x_0)^{n+1}$

Issue

- for composite functions $\boldsymbol{\Delta}$ can be largely overestimated

Methodology for Taylor Models

Define arithmetic operations on Taylor Models:

- $\texttt{TM}_{\texttt{add}}$, $\texttt{TM}_{\texttt{mul}}$, $\texttt{TM}_{\texttt{comp}}$, and $\texttt{TM}_{\texttt{div}}$
- E.g., $\texttt{TM}_{\texttt{add}} : \Big((P_1, \boldsymbol{\Delta_1}),\ (P_2, \boldsymbol{\Delta_2})\Big) \mapsto (P_1 + P_2, \boldsymbol{\Delta_1} + \boldsymbol{\Delta_2})$.

A two-fold approach:

- apply these operations recursively on the structure of the function
- use Taylor-Lagrange remainder for atoms (i.e., for base functions)

Methodology for Taylor Models

Define arithmetic operations on Taylor Models:

- $\texttt{TM}_{\texttt{add}}$, $\texttt{TM}_{\texttt{mul}}$, $\texttt{TM}_{\texttt{comp}}$, and $\texttt{TM}_{\texttt{div}}$
- E.g., $\texttt{TM}_{\texttt{add}} : \Big((P_1, \boldsymbol{\Delta_1}),\ (P_2, \boldsymbol{\Delta_2})\Big) \mapsto (P_1 + P_2, \boldsymbol{\Delta_1} + \boldsymbol{\Delta_2})$.

A two-fold approach:

- apply these operations recursively on the structure of the function
- use Taylor-Lagrange remainder for atoms (i.e., for base functions)

We need to consider a relevant class for base functions, so that:

- we can easily compute their successive derivatives
- the interval remainder computed for these atoms is thin enough

D-finite functions

Definition

A D-finite function is a solution of a homogeneous linear ordinary differential equation with polynomial coefficients:

$$a_r(x)y^{(r)}(x) + \cdots + a_1(x)y'(x) + a_0(x)y(x) = 0, \qquad \text{for } a_k \in \mathbb{K}[X]$$

Example (exp)

The function $y = \exp$ is fully determined by $\{y' - y = 0,\ y(0) = 1\}$

- most common functions are D-finite

  (sin, cos, arcsin, arccos, sinh, cosh, arcsinh, arccosh, Si, Ci, Shi, Chi, arctan, exp, ln, Ei, erf, Ai, Bi, ...).

- tan is not D-finite

Taylor series of D-finite functions

Theorem

A function represented by a Taylor series $f(x) = \sum_{n=0}^{\infty} u_n \, (x - x_0)^n$ *is* D*-finite if and only if the sequence* (u_n) *of its Taylor coefficients satisfies a linear recurrence with polynomial coefficients.*

recurrence relation
initial conditions $\Big\}$ $\Rightarrow$ **fast numerical computation** of Taylor coefficients

Example ($\exp$)

Taylor series: $\exp(x) = \sum_{n=0}^{\infty} \frac{\exp(x_0)}{n!} \, (x - x_0)^n$

Recurrence: $\forall n \in \mathbb{N}, \; u_{n+1} = \frac{u_n}{n+1}$ Initial condition: $u_0 = \exp(x_0)$

Outline

1. Algorithms for Taylor Models
2. Formalization of Taylor Models in COQ
3. Current Results and Future Developments

How we use COQ

For Taylor Models

- implement TM algorithms in Coq
- formally prove these algorithms
- compute in Coq the TMs

Levels of trust

method	trust	speed
compute (kernel)	+++	+
vm_compute (byte code)	++	++
native_compute (native code)	+	+++

Coq and real numbers

The "Reals" library

- designed for high level proofs
- available in the COQ Standard Library
- defined by axioms
  e.g. $r_1 + (r_2 + r_3) = (r_1 + r_2) + r_3$
- use classical reasoning $\forall r, r = 0 \vee r \neq 0$
- definitions and proofs from "paper mathematics"
  e.g. convergence, derivability, fundamental theorem of calculus etc.
- but no computational power

Computing with real numbers

- libraries for computation in arbitrary precision (e.g. by O'Connor)
- the Flocq library for multiple-precision floating-point arithmetic
- the CoqInterval library for interval arithmetic

- formal verification of these libraries with respect to (some) standard implementation of real numbers

Formally verified computation: CoqInterval

- abstract interface for intervals
- instantiation to intervals with floating point bounds
- formal verification with respect to the "Reals" library

$x, y : \mathsf{R} \cup \{\texttt{NaN}\}$
$\mathbf{X}, \mathbf{Y} : \mathsf{IR}$

$$x \in \mathbf{X}, y \in \mathbf{Y} \Rightarrow x + y \in \mathbf{X} + \mathbf{Y}$$

$$x \in \mathbf{X} \Rightarrow \exp(x) \in \mathbf{Exp}(\mathbf{X})$$

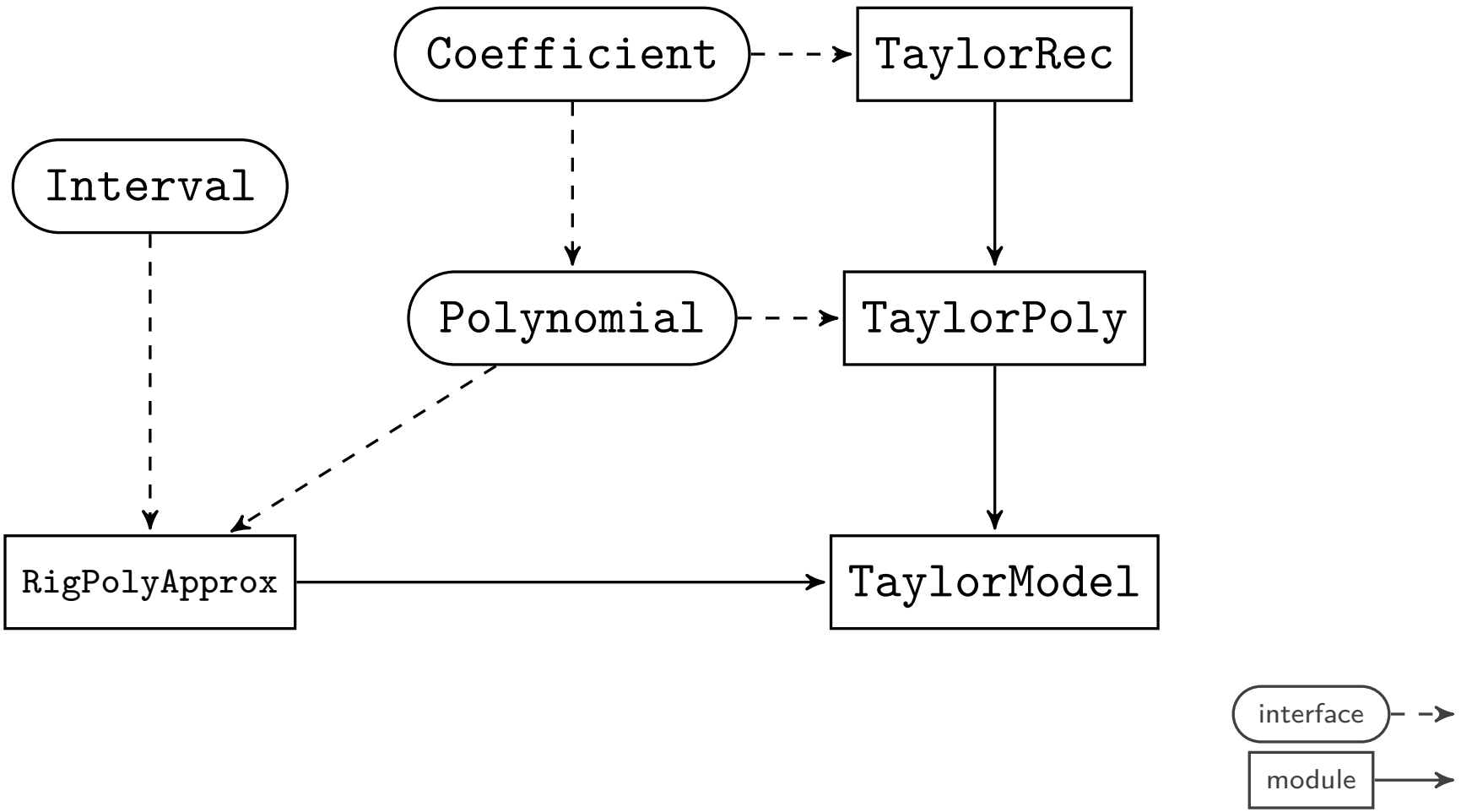
Implementation of Taylor models in COQ

Focus on being generic

- Taylor models are an instance of a rigorous polynomial approximation (i.e. a pair (P, Δ))
- generic with respect to the type of coefficients of polynomial P, to its implementation, as well as the type of interval $\boldsymbol{\Delta}$

Prove correctness with respect to the standard "Reals" library

A generic implementation of TMs: modular hierarchy



`Coefficient, Polynomial, Interval, RigPolyApprox`

`Coefficient:`

tzero, tone, tadd, tmul, tdiv, tnat, texp, tsin, ...

`Polynomial:`

`tadd, tmul, tmul_trunc, teval, tnth, tsize, trec1, trec2, tfold, ...`

`Interval:`

- reuse CoqInterval library
- abstract interval operations: `I.add, I.exp, ...`

`RigPolyApprox:`

- the `RPA` structure: a pair (polynomial, interval)

TaylorRec, TaylorPoly, TaylorModel

TaylorRec

```
Definition exp_rec n u := tdiv u (tnat n).
```

TaylorPoly

```
Definition T_exp n u := trec1 exp_rec (texp u) n.
```

TaylorModel

```
Definition TM_exp n I x0 :=
    RPA (T_exp n x0) (Trem T_exp n I x0).
```

Example instance of the hierarchy

`Coefficient:` intervals with multiple precision floating point bounds from CoqInterval

`Polynomial:` lists

`Interval:` intervals with multiple precision floating point bounds from CoqInterval

A comparison

Sollya

- written in C
- based on the MPFI library (Multiple-Precision FP IA)
- contains an implementation of Taylor Models
- in an imperative-programming framework
- polynomials as arrays of coefficients

CoqApprox

- formalized in Coq
- based on the CoqInterval library
- implements Taylor Models using a similar algorithm
- in a functional-programming framework
- polynomials as lists of coefficients (linear access time)

COQ is less than 10 times slower than Sollya! It's very good!

Some benchmarks for base functions

	Timing		Approximation error	
	Coq	Sollya	Coq	Sollya
arctan prec=120, deg=8 $\boldsymbol{I}=[1,2]$ split in 256	11.45s	1.03s	7.43×10^{-29}	2.93×10^{-29}
exp prec=600, deg=40 $\boldsymbol{I}=[\ln 2, 1]$ split in 256	38.10s	16.39s	6.23×10^{-182}	6.22×10^{-182}

Some benchmarks for composite functions

	Timing		Approximation error	
	Coq	Sollya	Coq	Sollya
exp $\times$ sin prec=200, deg=10 $\boldsymbol{I}$=$[1/2, 1]$ split in 2048	1m22s	12.05s	6.92×10^{-50}	6.10×10^{-50}
exp $\circ$ sin prec=200, deg=10 $\boldsymbol{I}$=$[1/2, 1]$ split in 2048	3m24s	12.19s	4.90×10^{-47}	4.92×10^{-47}

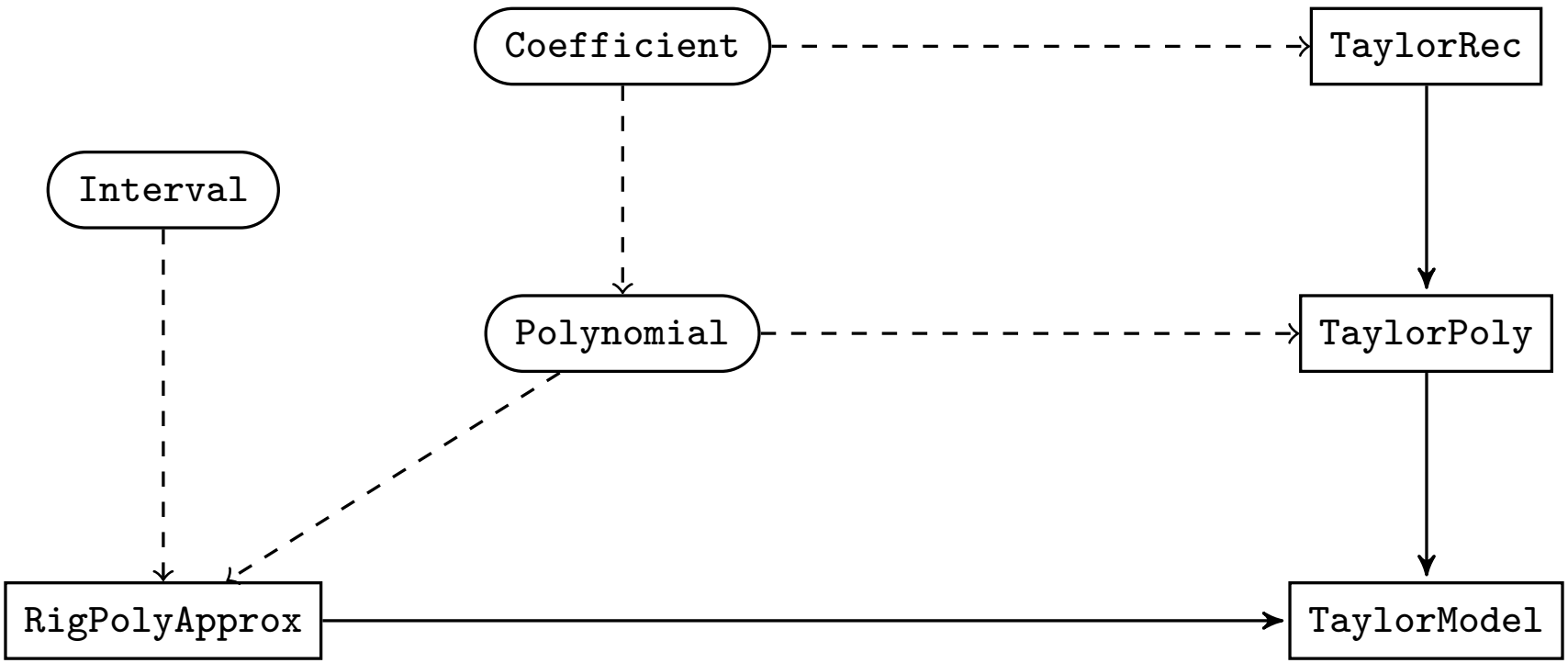
Proving Taylor models in COQ

Definition (validTM)

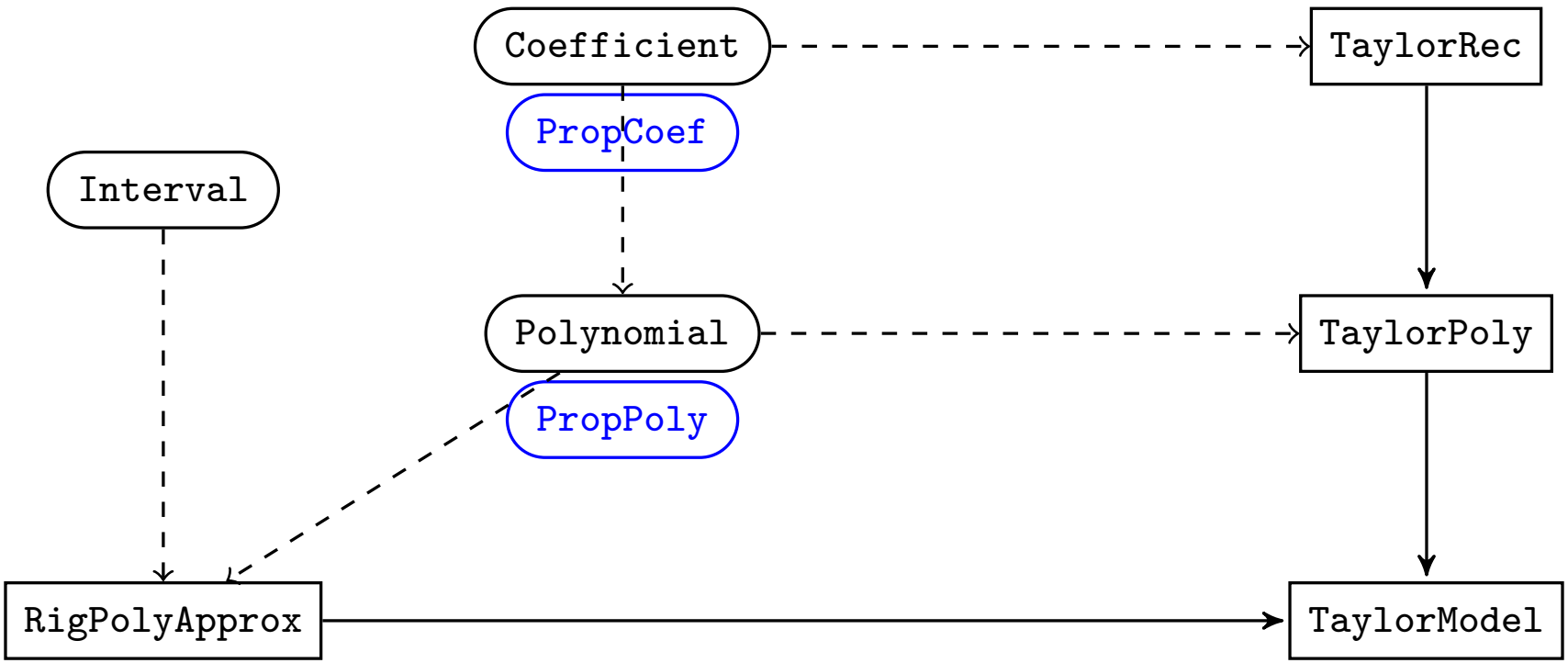
Let $f : \boldsymbol{I} \to \mathbb{R}$ be a function, $\boldsymbol{x_0}$ be a small interval around an expansion point x_0. Let T be a polynomial with interval coefficients $\boldsymbol{a_0}, \ldots, \boldsymbol{a_n}$ and $\boldsymbol{\Delta}$ an interval. We say that $(T, \boldsymbol{\Delta})$ is a Taylor model of f at $\boldsymbol{x_0}$ on $\boldsymbol{I}$ when

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

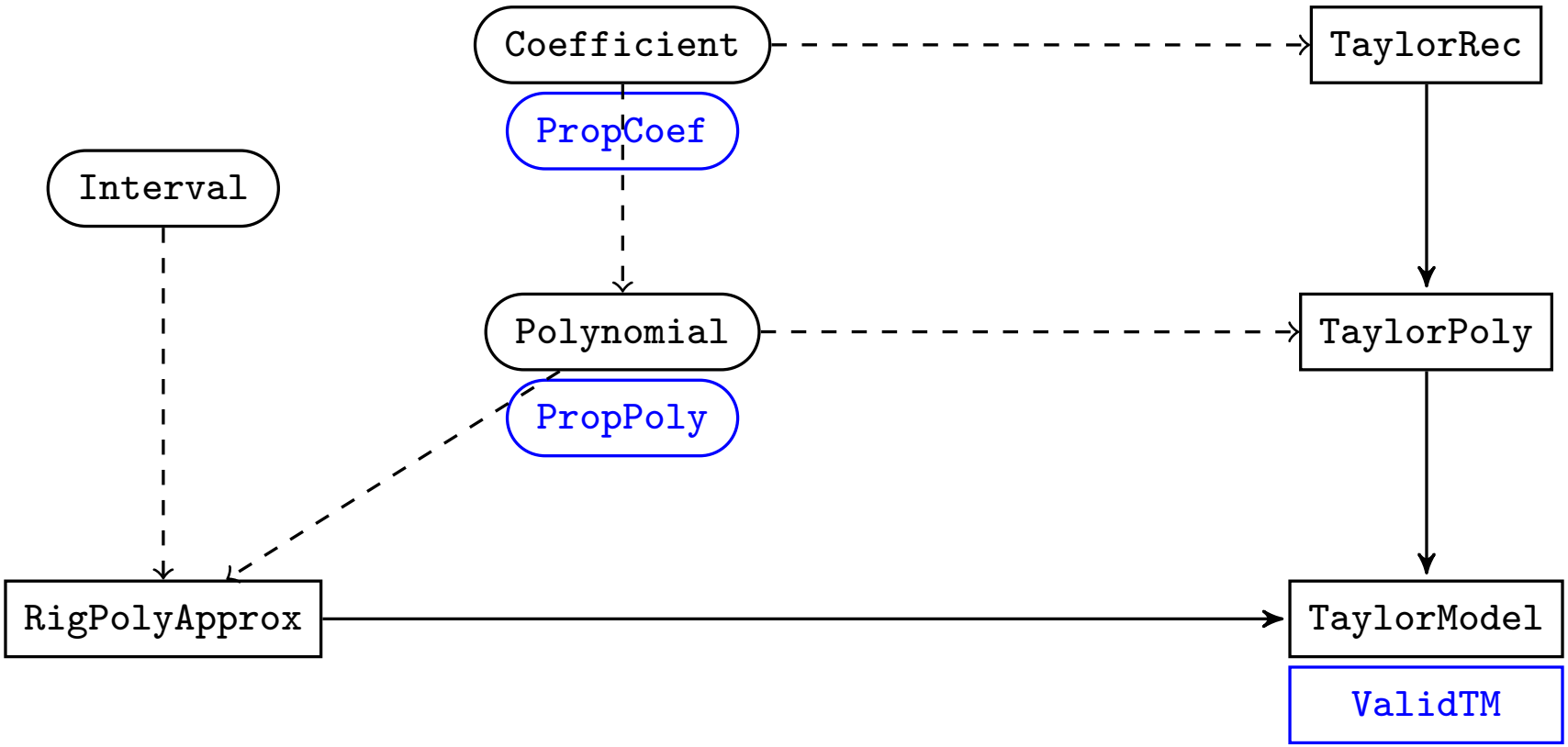
Adapting the hierarchy for proofs



Adapting the hierarchy for proofs



Adapting the hierarchy for proofs



Problems with the specification

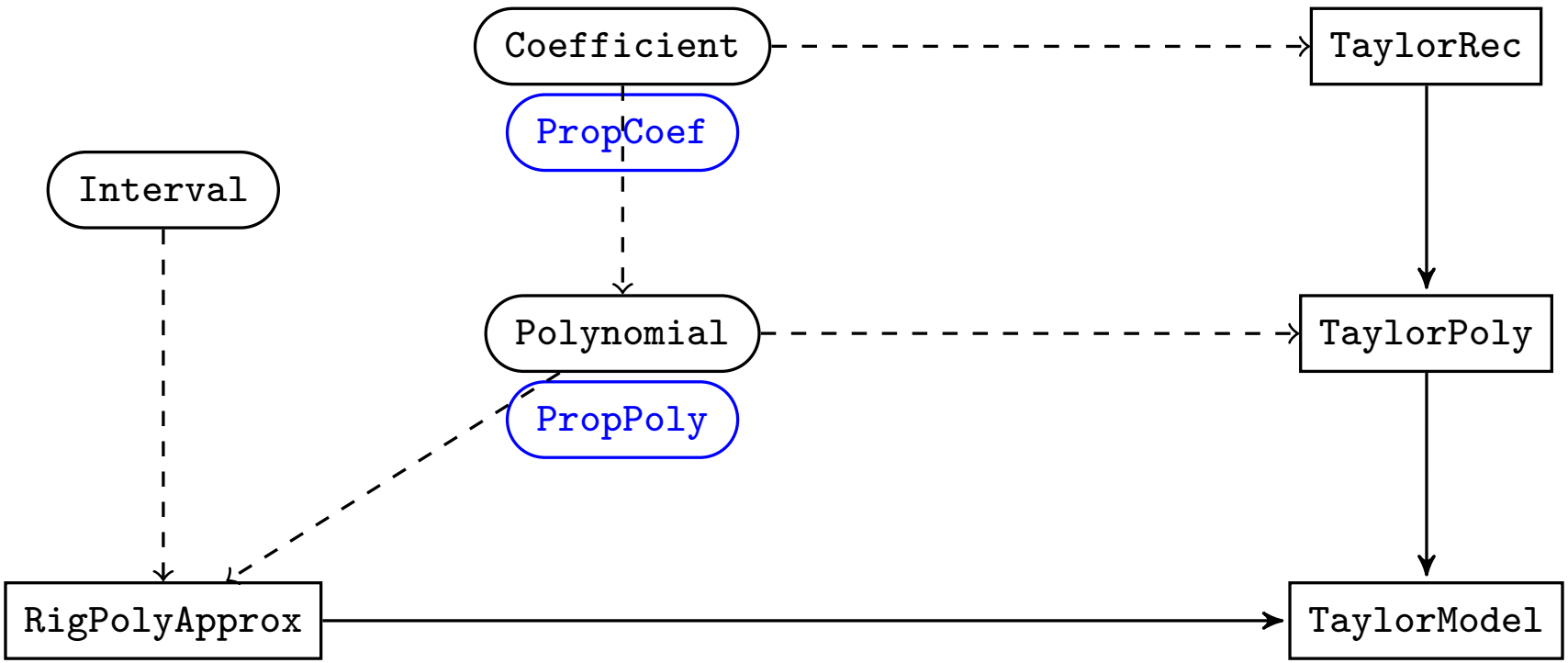
At the coefficient level
e.g associativity of addition

- holds for real numbers
- does not hold for floating point numbers or intervals

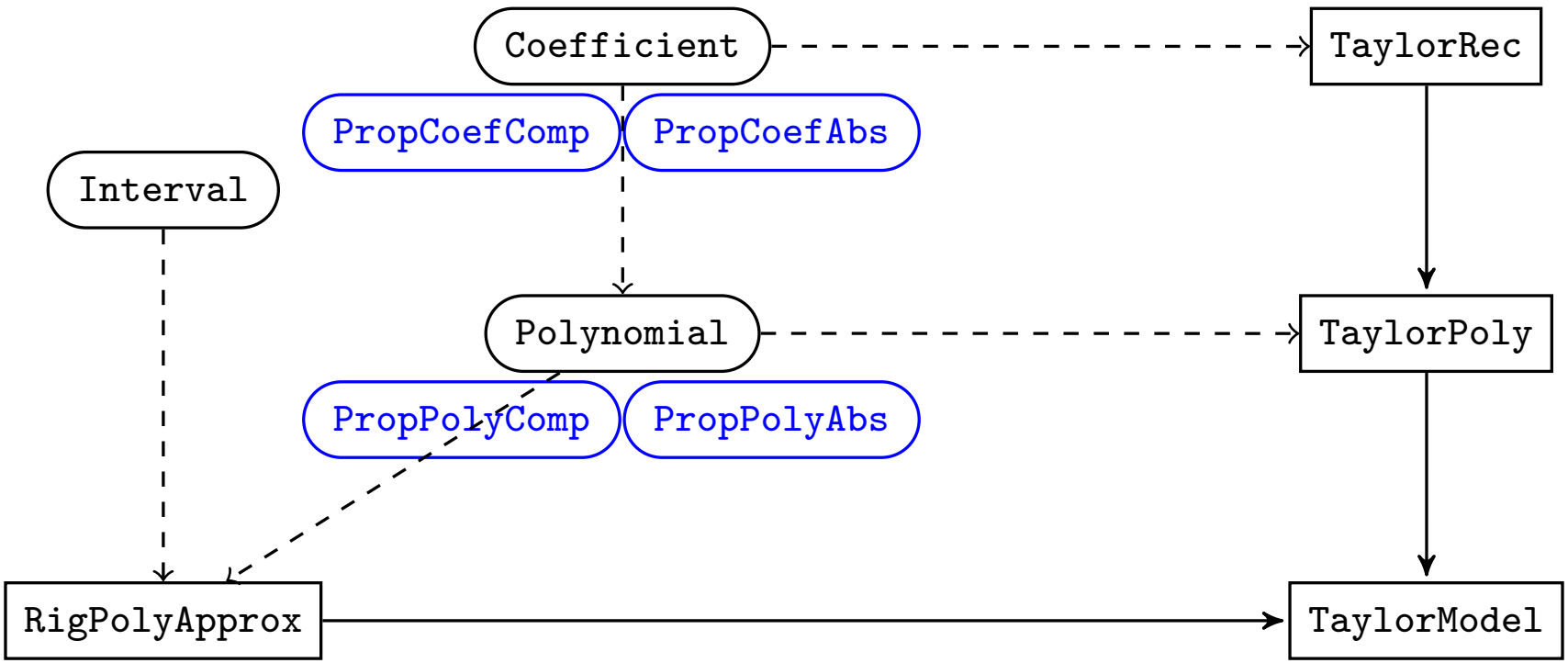
At the polynomial level
e.g. eval $(P + Q) =$ eval $P +$ eval Q

- holds for polynomials with real number coefficients
- does not hold for polynomials with interval coefficients

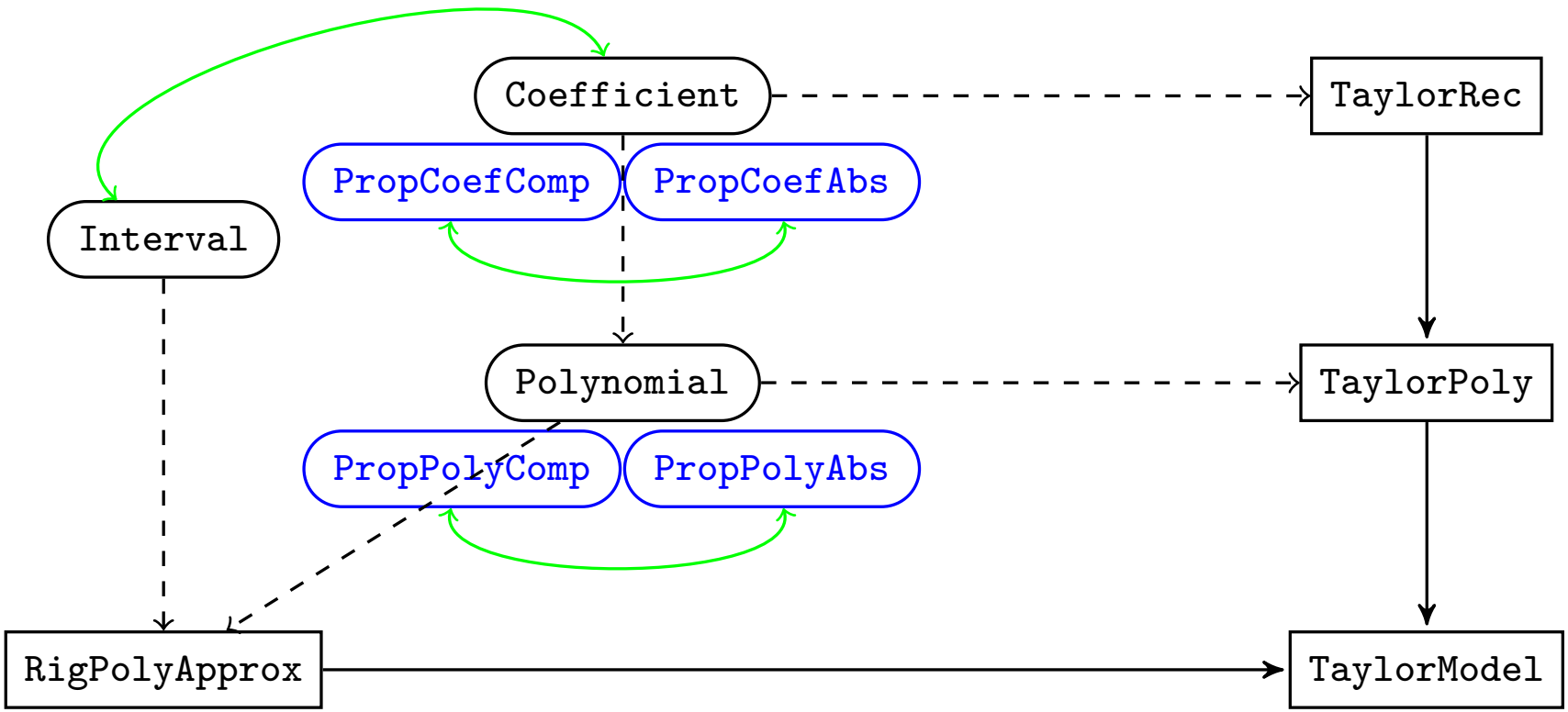
Adding specifications to the hierarchy



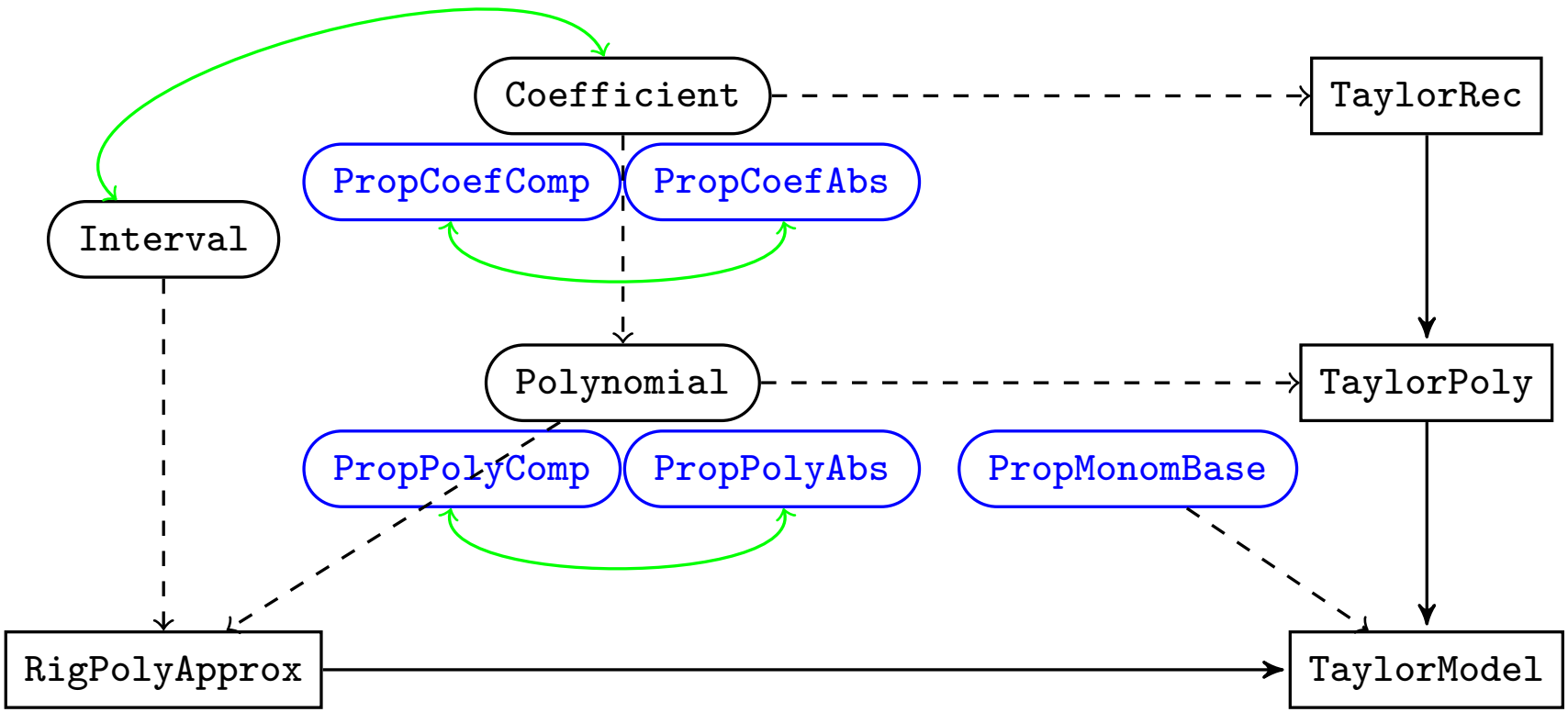
Adding specifications to the hierarchy



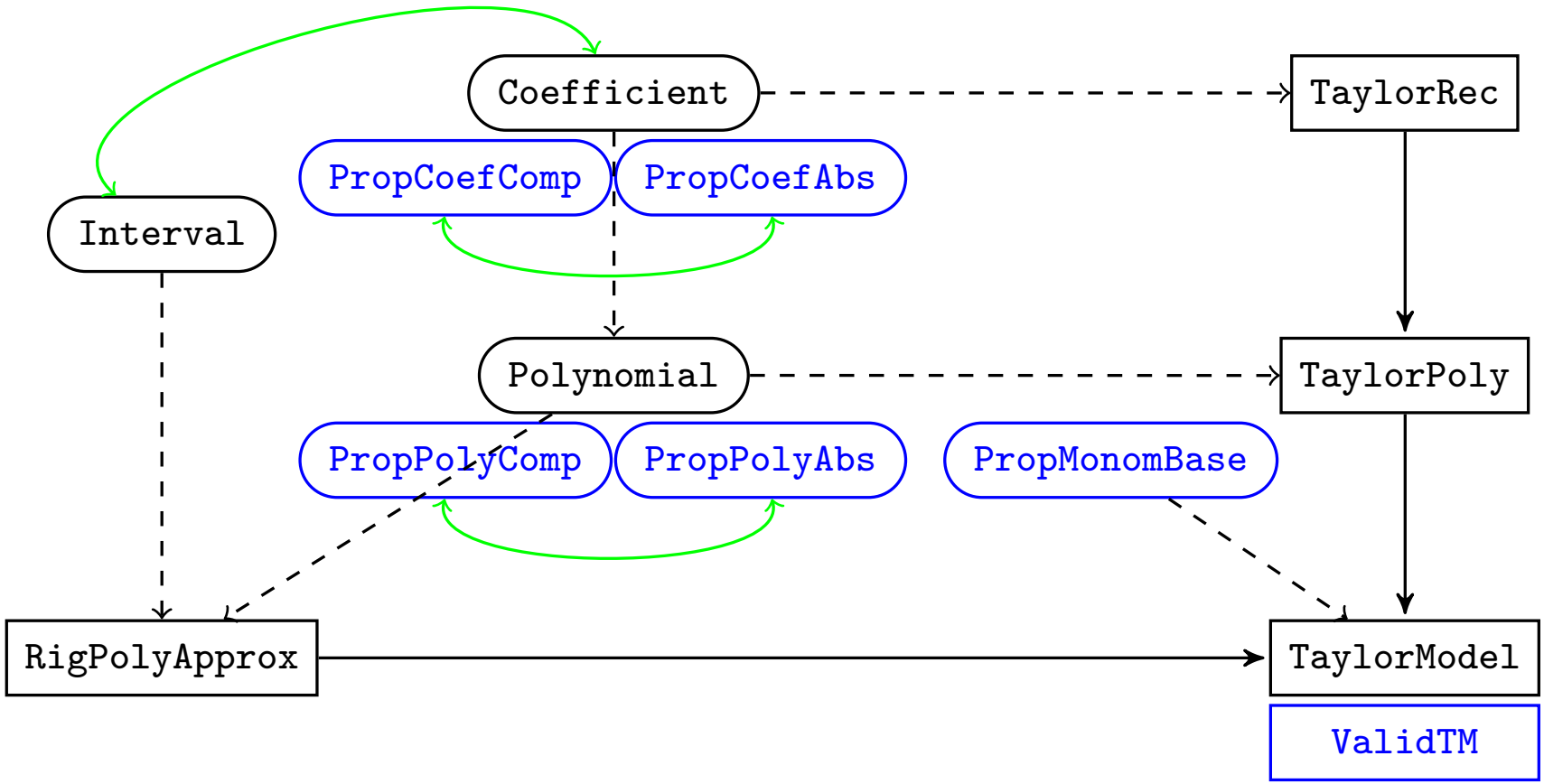
Adding specifications to the hierarchy



Adding specifications to the hierarchy



Adding specifications to the hierarchy



Proof for TMs of base functions

- take advantage by the fact that they are defined in a uniform way
- have a generic proof based on Taylor-Lagrange theorem
- instantiate to each function

Example: exp

$\mathtt{TM}_{\mathsf{exp}}(\boldsymbol{I}, \boldsymbol{x_0}, n) := (\boldsymbol{a_0} :: \ldots :: \boldsymbol{a_n}, \boldsymbol{\Delta})$ with

$$\boldsymbol{x_0} \subseteq \boldsymbol{I}, \quad \boldsymbol{a_0} = \mathbf{Exp}(\boldsymbol{x_0}), \quad \boldsymbol{a_{n+1}} = \frac{\boldsymbol{a_n}}{n+1}, \quad \boldsymbol{\Delta} = \frac{\mathbf{Exp}(\boldsymbol{I})}{(n+1)!} * (\boldsymbol{I} - \boldsymbol{x_0})^{n+1}$$

Example: exp

$\mathtt{TM}_{\mathtt{exp}}(\boldsymbol{I}, \boldsymbol{x_0}, n) := (\boldsymbol{a_0} :: \ldots :: \boldsymbol{a_n}, \boldsymbol{\Delta})$ with

$$\boldsymbol{x_0} \subseteq \boldsymbol{I}, \quad \boldsymbol{a_0} = \mathbf{Exp}(\boldsymbol{x_0}), \quad \boldsymbol{a_{n+1}} = \frac{\boldsymbol{a_n}}{n+1}, \quad \boldsymbol{\Delta} = \frac{\mathbf{Exp}(\boldsymbol{I})}{(n+1)!} * (\boldsymbol{I} - \boldsymbol{x_0})^{n+1}$$

We want to show $\mathtt{TM}_{\mathtt{exp}}(\boldsymbol{I}, \boldsymbol{x_0}, n)$ is a valid TM for $\exp$.

- $\boldsymbol{x_0} \subseteq \boldsymbol{I}$
- $0 \in \boldsymbol{\Delta}$
- $\forall \xi_0 \in \boldsymbol{x_0}, \exists \alpha_0 \in \boldsymbol{a_0}, \ldots, \alpha_n \in \boldsymbol{a_n},$

  $$\forall x \in \boldsymbol{I}, \exp(x) - \sum_{i=0}^{n} \alpha_i \left(x - \xi_0\right)^i \in \boldsymbol{\Delta}$$

Example: exp

$\mathtt{TM}_{\mathtt{exp}}(\boldsymbol{I}, \boldsymbol{x_0}, n) := (\boldsymbol{a_0} :: \ldots :: \boldsymbol{a_n}, \boldsymbol{\Delta})$ with

$$\boldsymbol{x_0} \subseteq \boldsymbol{I}, \quad \boldsymbol{a_0} = \mathbf{Exp}(\boldsymbol{x_0}), \quad \boldsymbol{a_{n+1}} = \frac{\boldsymbol{a_n}}{n+1}, \quad \boldsymbol{\Delta} = \frac{\mathbf{Exp}(\boldsymbol{I})}{(n+1)!} * (\boldsymbol{I} - \boldsymbol{x_0})^{n+1}$$

We want to show $\mathtt{TM}_{\mathtt{exp}}(\boldsymbol{I}, \boldsymbol{x_0}, n)$ is a valid TM for $\exp$.

- $\boldsymbol{x_0} \subseteq \boldsymbol{I}$
- $0 \in \boldsymbol{\Delta}$
- $\forall \xi_0 \in \boldsymbol{x_0}, \exists \alpha_0 \in \boldsymbol{a_0}, \ldots, \alpha_n \in \boldsymbol{a_n},$

  $$\forall x \in \boldsymbol{I}, \exp(x) - \sum_{i=0}^{n} \alpha_i \left(x - \xi_0\right)^i \in \boldsymbol{\Delta}$$

$$\exists \alpha_i = \frac{\exp(\xi_0)}{i!} \in \boldsymbol{a_i} \text{ s.t.}$$

$$\exp(x) - \sum_{i=0}^{n} \frac{\exp(\xi_0)}{i!} \left(x - \xi_0\right)^i = \frac{\exp(c_i)}{(n+1)!} * (x - \xi_0)^{n+1} \in \boldsymbol{\Delta}, \text{ as } c_i \in \boldsymbol{I}$$

Generalization to an arbitrary D-finite function f

Difficulty:

find minimal assumptions on the function f

- the derivative (in the sense of COQ) is compatible with the recurrence relation
- we have a compatible interval evaluator for f
- f propagates `NaN`s

provide the Taylor-Lagrange theorem for standard Reals

In practice

- a generic proof for first order recurrences `proofOfRec1`
- another generic proof for second order recurrences
- a generic proof for recurrences of order N is future work

Example

```
Theorem TM_exp_valid: validTM TM_exp X0 I n Rexp.
Proof.
...
apply proofOfRec1.
...
Qed.
```

Proof for composite functions

Proof of the algorithm based on the specification

- addition: straightforward
- multiplication: almost straightforward
- composition: based on multiplication, addition and constant function TMs
- division: it's a multiplication and a composition with $x \mapsto \frac{1}{x}$

Proof status

Fun/Op	Reals	CoqInterval	Implemented in CoqApprox	Proved in CoqApprox
cst	☒	☒	☒	☒
id	☒	☒	☒	☒
inv	☒	☒	☒	☒
sqrt	☒	☒	☒	☒
$\frac{1}{\sqrt{\cdot}}$	☒	☒	☒	☒
exp	☒	☒	☒	☒
sin	☒	☒	☒	☒
cos	☒	☒	☒	☒
arctan	☒	☒	☒	☐
ln	☒	☐	☒	☐
arcsin	☐	☐	☒	☐
arccos	☐	☐	☒	☐
$\mathtt{TM}_{\mathtt{add}}$			☒	☒
$\mathtt{TM}_{\mathtt{mul}}$			☒	☒
$\mathtt{TM}_{\mathtt{comp}}$			☒	☒
$\mathtt{TM}_{\mathtt{div}}$			☒	☒

Missing pieces

functions missing from the Reals library

- cannot provide a proof for the Taylor model
- find a generic way of adding a new function to Reals
- e.g. define them by a differential equation or a recurrence rel.

Missing pieces

functions missing from the Reals library

- cannot provide a proof for the Taylor model
- find a generic way of adding a new function to Reals
- e.g. define them by a differential equation or a recurrence rel.

functions missing from CoqInterval

- cannot provide an initial value for the Taylor model
- just implement the missing functions in CoqInterval
- use other techniques (fixed point theorems, majorazing series)

Other issues or what a formal proof reveals

From arithmetic

- Does the constant function propagate `NaN`?
- What is the interval $[\texttt{NaN}, \texttt{NaN}]$? Does it contain `NaN`?
- Is the null polynomial a valid Taylor model?
- Is the interval $[1, 0]$ empty?

Other issues or what a formal proof reveals

From arithmetic

- Does the constant function propagate `NaN`?
- What is the interval $[\texttt{NaN}, \texttt{NaN}]$? Does it contain `NaN`?
- Is the null polynomial a valid Taylor model?
- Is the interval $[1, 0]$ empty?

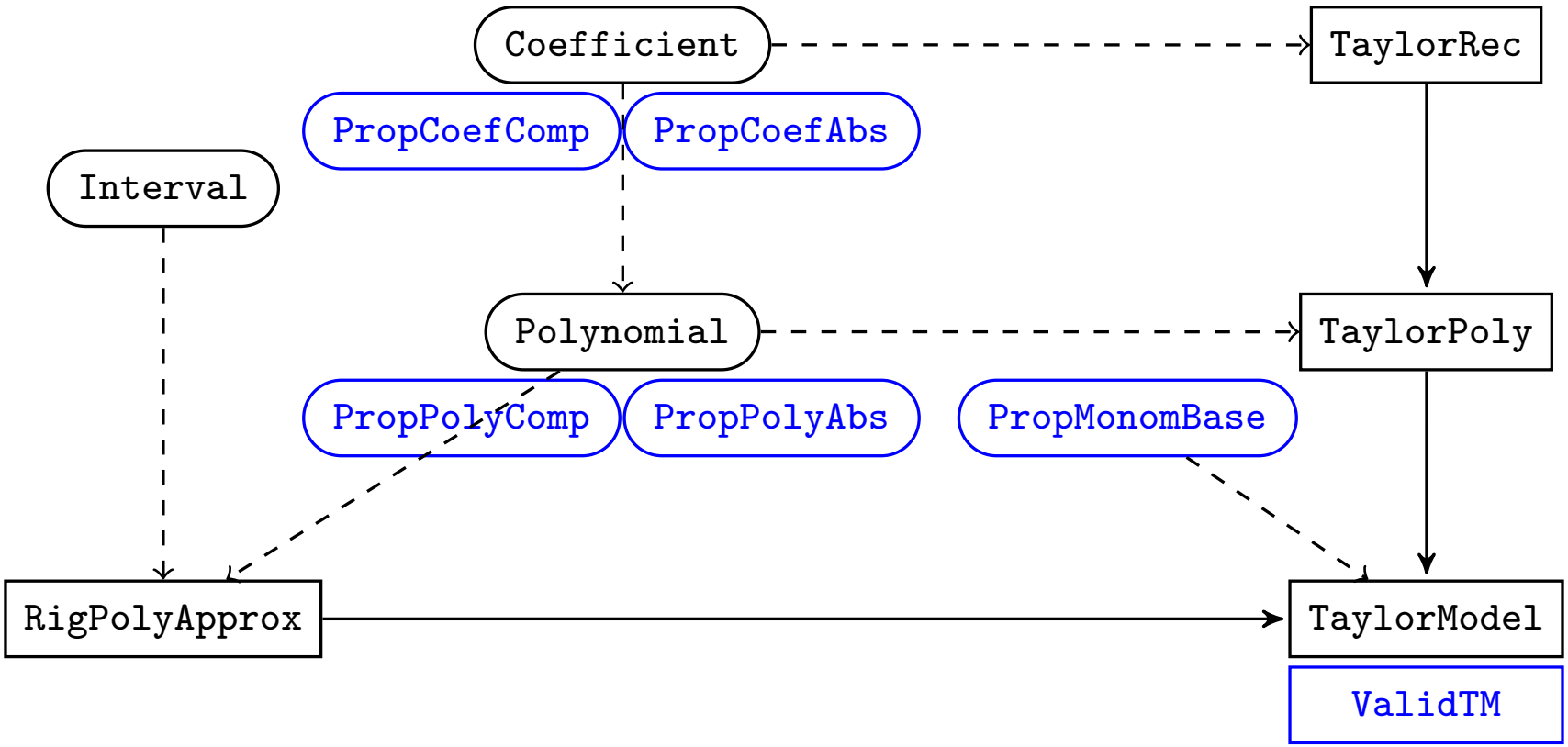
From COQ

- dealing with extended standard reals: lack of automatic tactics
- dealing with derivation in COQ

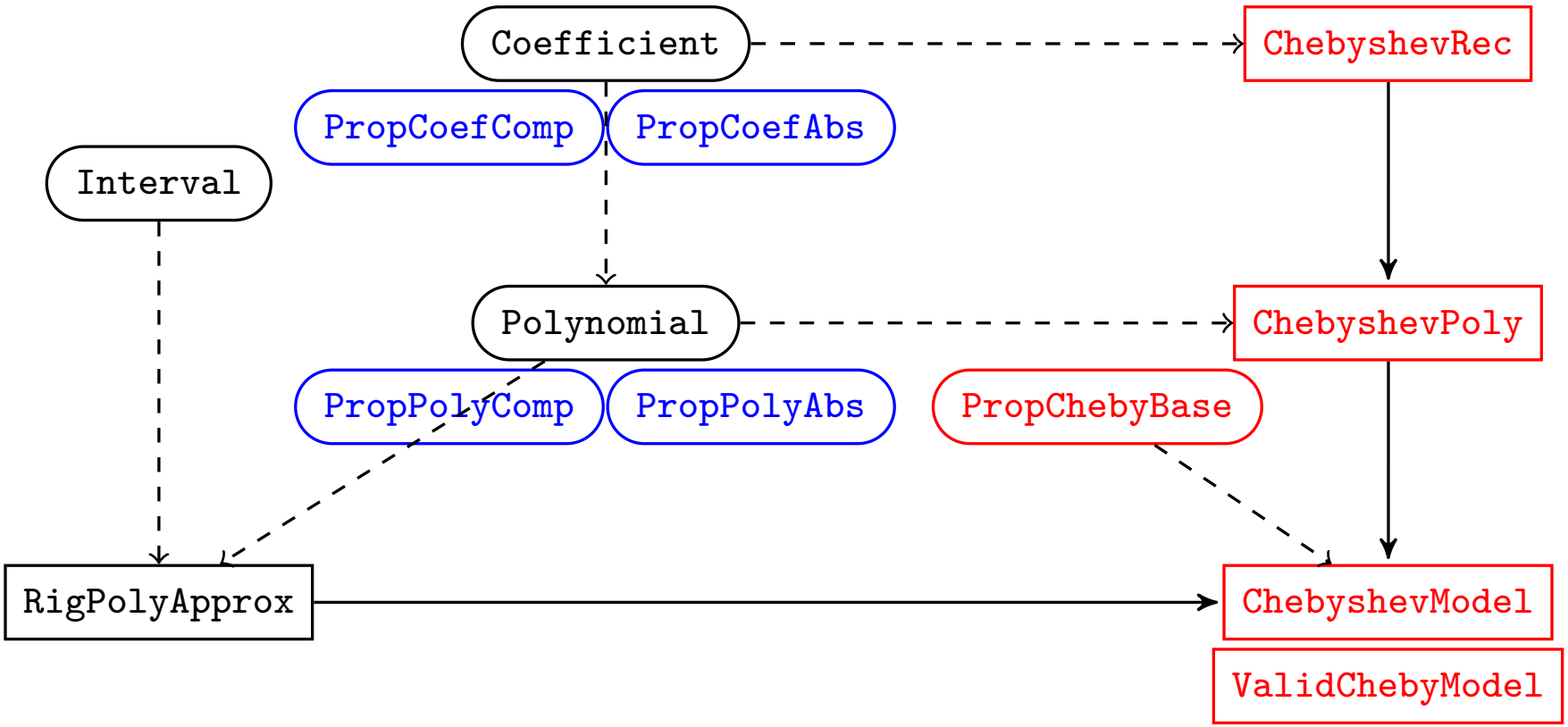
Future Work

- optimize algorithms for existing base functions
- add more functions
- consider functions in several dimensions
- consider other rigorous polynomial approximations, like Chebyshev Models

Future work


Coefficient
TaylorRec
PropCoefComp
PropCoefAbs
Interval
Polynomial
TaylorPoly
PropPolyComp
PropPolyAbs
PropMonomBase
RigPolyApprox
TaylorModel
ValidTM

Future work


Coefficient
ChebyshevRec
PropCoefComp
PropCoefAbs
Interval
Polynomial
ChebyshevPoly
PropPolyComp
PropPolyAbs
PropChebyBase
RigPolyApprox
ChebyshevModel
ValidChebyModel

Related work

- Francisco Cháves, *Utilisation et certification de l'arithmétique d'intervalles dans un assistant de preuves*. PhD Thesis. 2007
- Roland Zumkeller, *Global Optimization in Type Theory*. PhD Thesis. 2008
- P. Collins, M. Niqui and N. Revol, *A Validated Real Function Calculus*. Mathematics in Computer Science. 2011

Overview

Work in collaboration between the formal proof community and arithmetic, symbolic and numeric computation communities.

Interesting for formal proofs:

- computing power of COQ: is it enough?
- comprehensive library on Reals, CoqInterval?
- state of the art algorithms

Interesting for arithmetic, symbolic and numeric computation:

- real algorithms, but with a proof of correctness