Formalized foundations of polynomial real analysis

Cyril Cohen

INRIA Saclay – Île-de-France
LIX École Polytechnique
INRIA Microsoft Research Joint Centre
Cyril.Cohen@inria.fr

14/10/2010

This work has been partially funded by the FORMATH project, nr. 243847, of the FET program within the 7th Framework program of the European Commission.

Purpose of this formalization

Formalize real polynomial analysis **in the** SSReflect **extension of the** Coq **proof assistant**.

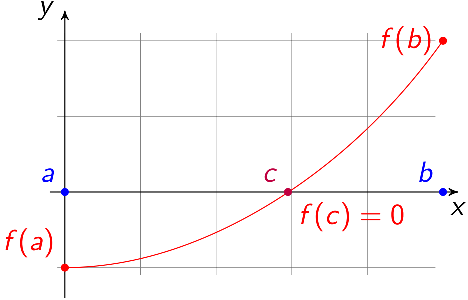
SSReflect provides a lot of **tools** and uses a lot of specific **programming techniques** in the domain of *finite groups* and *combinatorics*.

Reuse theses techniques to handle more « continuous » theories.

Real Closed Fields

Algebraic structure of reals : *Real Closed Fields* (*RCF*)

Field + Ordered + Intermediate value theorem for polynomials



Decidable Equality

In SSReflect, structures have decidable equality.
We can define this (implicit) coercion in Coq

```
Coercion is_true (b : bool) : Prop := (b = true).
```

SSReflect

- uses intensively this coercion
- has facilities to go from one point of view to the other (bool-Prop reflection).

We then see boolean equality as propositional equality, for free.

What do we need ?

- ⇒ Make case analysis on $x \leq y$
- ⇒ Combine statements (using transitivity with both $\leq$ $<$, compatibility with operations, etc ..)
- ⇒ Speak about signs and absolute value
- ⇒ Use max and min

Taking advantage of the boolean predicate

Making `le` a boolean predicate.

Like before, consider this boolean predicate as proposition through the coercion `is_true`

⇒ Use equalities to rewrite expressions with order

- `(x+z <= y+z)= (x <= y)`
- `(sign x == 1)= (0 < x)`
- ...

Taking advantage of the boolean predicate

Making `le` a boolean predicate.

Like before, consider this boolean predicate as proposition through the coercion `is_true`

⇒ Use equalities to rewrite expressions with order

- `(x+z <= y+z)= (x <= y)`
- `(sign x == 1)= (0 < x)`
- ...

⇒ Use `if x <= y then ... else ...` in programs

Tools

- Multiple lemmas about transitivity and compatibility between `le`, `lt` and field operations

$\Rightarrow$ Need for good naming conventions.

Strict comparison

We'll define the strict order `lt` from the large one `le` by :

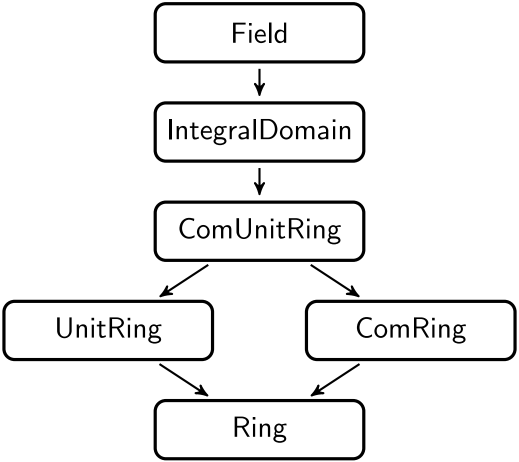
```
Definition lt x y := ~~ (le y x).
```

and prove its properties.

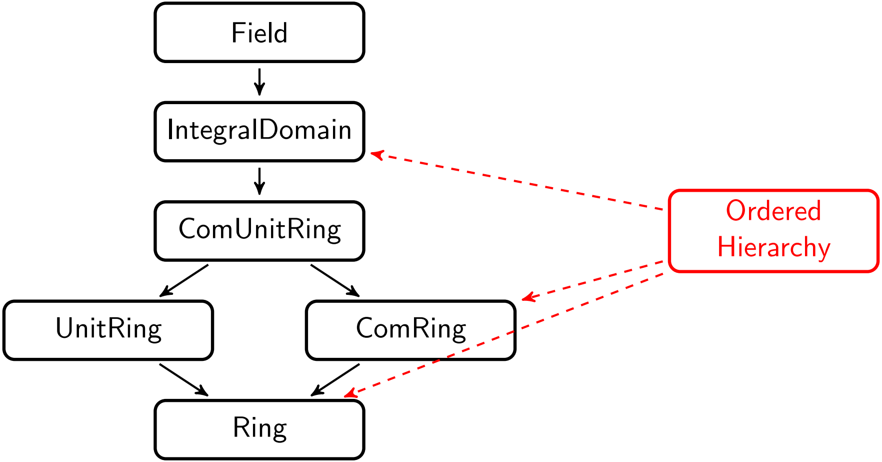
Ordered ring mixin

```
Record mixin_of (R : ringType) := Mixin {
  le : rel R;
  _ : antisymmetric le;
  _ : transitive le;
  _ : total le;
  _ : forall z x y, le x y -> le (x + z) (y + z);
  _ : forall x y, le 0 x -> le 0 y -> le 0 (x * y)
}.
```

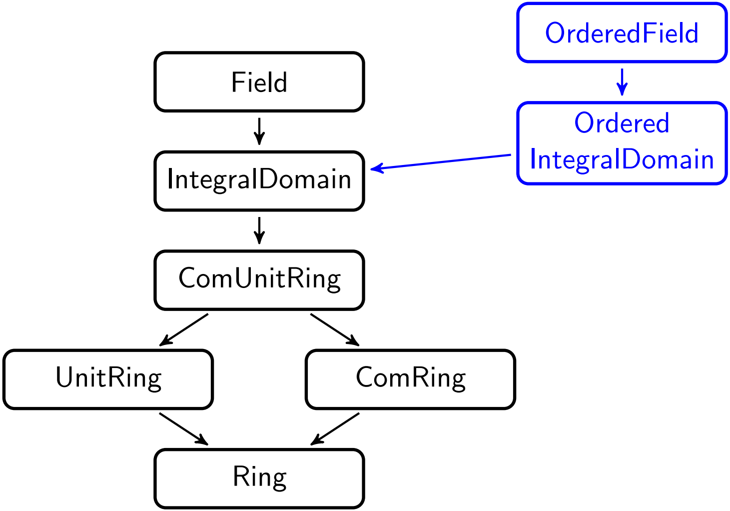
Integration in existing SSReflect algebraic hierarchy



Integration in existing SSReflect algebraic hierarchy



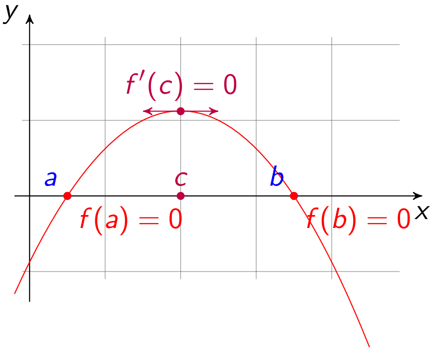
Integration in existing SSReflect algebraic hierarchy



Rolle Theorem for polynomials

A first hint that *RCF* is a good abstraction of reals :
We are able to prove :

```
Lemma rolle : forall (a b : R) (p : {poly R}),
  a < b -> p.[a] = 0 -> p.[b] = 0 ->
  exists c, a < c < b /\ p^`().[c] == 0.
```



Sketch of the constructive proof

```
Lemma rolle_weak : forall (a b : R) (p : {poly R}),
   a < b -> p.[a] = 0 -> p.[b] = 0 ->
   exists c , a < c < b
        /\ (p^`().[c] = 0 \/ p.[c] = 0).
```

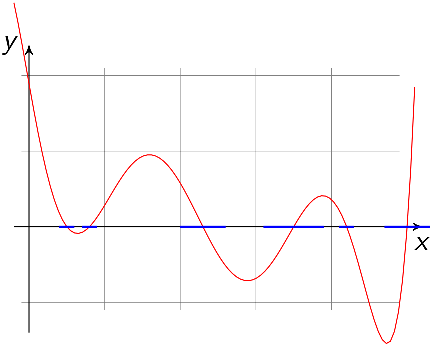
And conclude `rolle` from it by iterating `rolle_weak`. It terminates because P has less than $\deg(P)$ roots.

What else can we do with *IVT*?

Particularly useful examples

- Rolle Theorem
- Mean Value Theorem
- Write a function that computes the real roots of any polynomial
- Prove that given a polynomial P, and a root x of P, one can find a neighborhood of x on which P has no root except x.
- ...

Isolation of roots



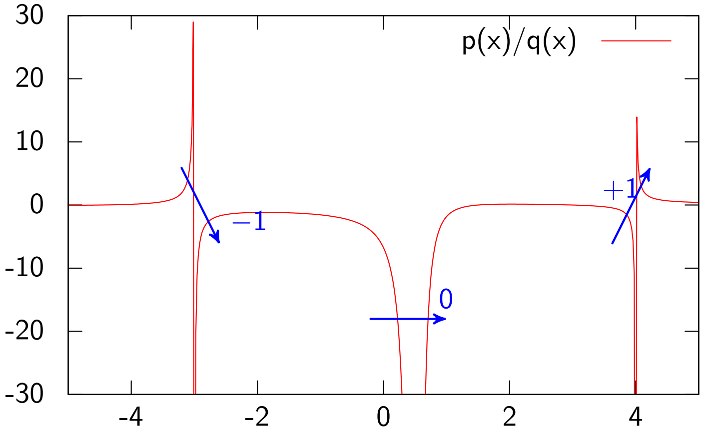
Towards quantifier elimination

First step of Quantifier Elimination in *RCF*.
Which entails decidability of the theory of *RCF*.

Let's pick one concept from it : Cauchy Index (proof almost done).

$$\mathrm{CInd}(\frac{P}{Q},]a, b[) =$$

$$\text{number of positive jumps} - \text{number of negative jumps}$$



Useful property of Cauchy Index

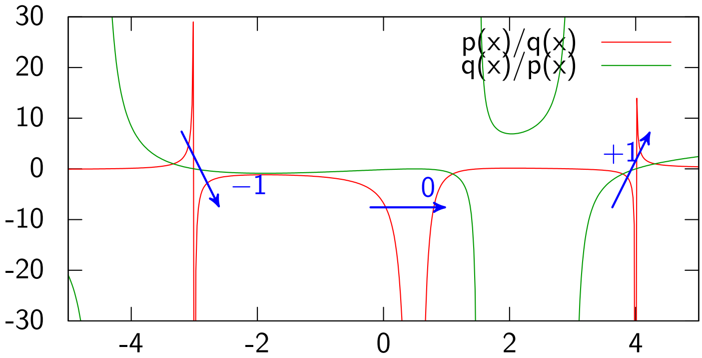
Property

If $P(a), P(b), Q(a), Q(b) \neq 0$ then,

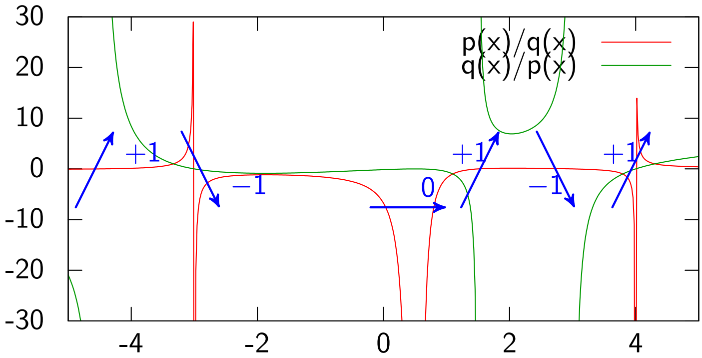
$$\mathrm{CInd}\left(\frac{P}{Q},]a,b[\right) + \mathrm{CInd}\left(\frac{Q}{P},]a,b[\right) =$$

$$\begin{cases} \mathrm{sign}\,(PQ(b)) & \text{if } PQ(a)PQ(b) < 0 \\ 0 & \text{else} \end{cases}$$

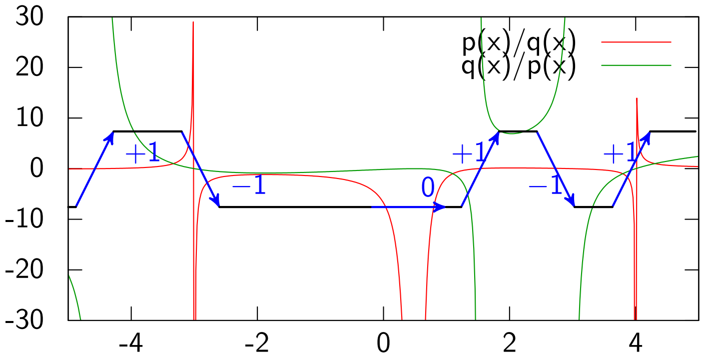
Idea of the proof : combinatorics



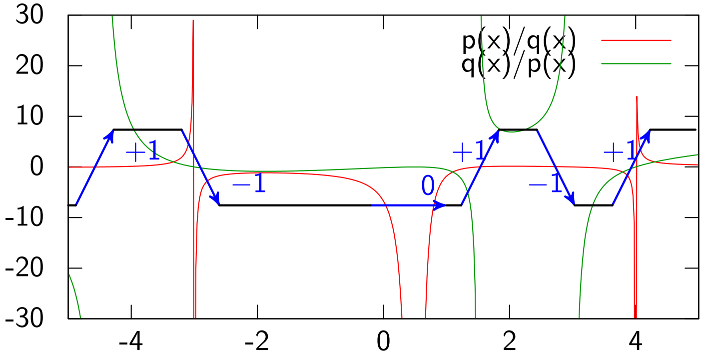
Idea of the proof : combinatorics



Idea of the proof : combinatorics



Idea of the proof : combinatorics

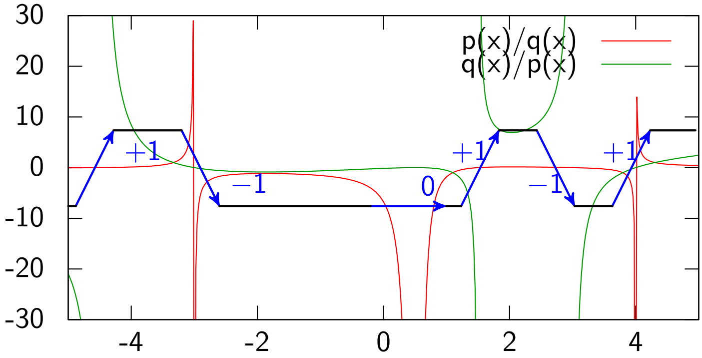


Jumps in the list of *signs* of PQ. `[-1;1;-1;-1;1;-1;1]`

Trick of the proof

The sum of jumps of a list $l = x_0, \ldots, x_n \in \{-1, 1\}^*$ verifies a useful property : it's the jump between x_0 and x_n.
i.e.

$$\begin{cases} \mathrm{sign}(x_n) & \text{if } x_0 x_n < 0 \\ 0 & \text{else} \end{cases}$$



Jumps in the list of *signs* of PQ. [-1;1;-1;-1;1;-1;1]
Jump between the first sign −1 and the last one 1, i.e.

$$\begin{cases} \text{sign}\,(PQ(b)) & \text{if } PQ(a)PQ(b) < 0 \\ 0 & \text{else} \end{cases}$$

Conclusion

A library which provides usable tools.
It is used in works in progress on

- Quantifier elimination in *RCF*
- Formalisation of Bernstein Polynomials

What next ?

- Instantiate the *Real Closed Fields* Structure
- Prove some reflexive tactics using it
- ... to provide a little more automation
- Generalize notion of continuity in this context
- Extend to further real analysis

The End

Thank you for your attention. Any questions ?