Coq formalization of polynomial interpretations on $\mathbb{Q}$

Kim Quyen LY

INRIA Internship

October 02, 2010
Supervisor: Frédéric BLANQUI

Outline

Problem

Problem:

- More complex definitions ⇒ termination more difficult to check
- Idea: use external tools to find termination proofs
- But how to be sure that these proofs are correct?

Solution: check the proofs they provide

- Formalize the theorem of polynomial interpretations in Coq
- Be able to check the hypotheses of this theorem automatically

Goal of internship

CoLoR: a **Co**q **L**ibrary **o**n **R**ewriting and termination

Polynomial interpretations on $\mathbb{Z}^+$ only

Goal:

- generalize to any (ordered) ring structure on a setoid
- prove the well-foundedness of the δ-ordering on $\mathbb{Q}^+$
- improve the criteria for monotony

Outline

Terms and rewriting

- Symbols: $f \in F$ n-ary
- Variables: $x \in V$
- Terms: $x | f(t_1, ..., t_n) \in T(F, V)$
- Subtitution $\sigma : V \to T(F, V)$:
  $x\sigma = \sigma(x)$, $f(t_1, ..., t_n)\sigma = f(t_1\sigma, ..., t_n\sigma)$
- Rewrite rule: pair of terms $l \to r$
- Rewrite step: $t \to u$ if $t = C[l\sigma]$ and $u = C[r\sigma]$

Reduction orderings

A *reduction ordering* is a relation $>$ on terms that is well-founded and stable by substitution and context

Theorem

A TRS R is **terminating** iff there exists a reduction order $>$ such that $R \subseteq >$

Special case of reduction ordering: interpretations

let $(D, >_D)$ be a well-founded domain and I and
I_f: $D^n \to D$ an interpretation function for every $f \in F$ n-ary

Definition: $t >_I r$, if $\forall v$: $\chi \to I$, $[\![t]\!]_I(v) >_D [\![u]\!]_I(v)$

Theorem

$>_I$ is a reduction ordering
if every $I_f : D^n \to D$ is monotone in each variable

Proof: If $>_D$: $t_1 >_I t_2 >_I$ ...., then $t_1 \to t_2 \to$ .... by the definition of $>_I$ then $[\![t_1]\!]_I(v) >_D [\![t_2]\!]_I(v) >_D$ ... is impossible because $>_I$ is well-founded by hypothesis. $>_I$ is a reduction ordering.

Special case of interpretation: Polynomials

$D \in \{\mathbb{Z}^+, \mathbb{Q}^+, \mathbb{R}^+\}$ and $I_f \in D[X_1, ..., X_n]$

Each condition can be written in terms of positivity checks:

1. monotone in each variable:
   I_f has positive coefficients and the coefficient of X_i is > 0
2. $l \to r$ is compatible with $>_I$:
   $P_l - P_r - 1$ has positive coefficients

Problem: well-founded relation $>_D$ for $D \in \{\mathbb{Q}^+, \mathbb{R}^+\}$?

Outline

Variables and function symbols

```
Notation variable := nat.

Record Signature : Type := mkSignature {
  symbol :> Type;
  arity : symbol -> nat;
  beq_symb : symbol -> symbol -> bool;
  beq_symb_ok : forall x y, beq_symb x y = true <-> x = y}.
```

Terms

```
Inductive term : Type :=
  | Var : variable -> term
  | Fun : forall f : Sig, vector term (arity f) -> term.

Inductive vector (A:Type) : nat -> Type :=
  | Vnil : vector 0
  | Vcons : forall (a:A)(n:nat), vector n -> vector (S n).
```

Rewriting

```
Record rule : Type := mkRule {lhs : term; rhs : term}.

Definition red u v := exists l r c s, In (mkRule l r) R /\
   u = fill c (sub s l) /\ v = fill c (sub s r).
```

Polynomials

```
Notation monom := (vector nat).
Definition poly n := list (Z * monom n).
  (* n = number of variables *)
```

Example: $3XY + 2X^2 + 1$ is represented by
$[(3, [|1; 1|]); (2, [|2; 0|]); (1, [|0; 0|)]$

```
Fixpoint coef n (m:monom n) (p:poly n) {struct p}: Z :=
  match p with
    | nil => 0
    | cons (c,m') p' =>
      match monom_eq_dec m m' with
        | left _ => c + coef m p'
        | right _ => coef m p'
      end
  end.
```

Outline

First solution

```
Module Type Ring.
Parameter A : Type.
Parameter A0 : A.
Parameter A1 : A.
Parameter Aadd : A -> A -> A.
...
Parameter Aring: ring_theory A0 A1 Aadd Amul Asub Aopp eqA.
End Ring.
```

problem: does not work with $\mathbb{Q}$!

Setoid equality

```
Module Type SetA.
  Parameter A : Type.
End SetA.

Module Type EqSet.
  Parameter A : Type.
  Parameter eqA : A -> A -> Prop.

  Notation "X =A= Y" := (eqA X Y).

  Parameter sid_theoryA : Setoid_Theory A eqA.
  ...
End Eqset.
```

Decidable equality

Eqset_dec is a module type of equality

```
Module Type Eqset_dec.
  Declare Module Export Eq : Eqset.
  Parameter eqA_dec : forall x y, {x =A= y} + {~x =A= y}.
End Eqset_dec.
```

alternative definition:

```
Definition beqA x y :=
  match eqA_dec x y with
    | left _ => true
    | right _ => false
  end.
```

```
Lemma beqA_ok : forall x y, beqA x y = true <-> x =A= y.
```

Final solution

```
Module Type Ring.
  Declare Module Export ES : Eqset_dec.
  Parameter A0 : A.
  Parameter A1 : A.
  Parameter Aadd : A -> A -> A.
  Add Morphism Aadd with signature
    eqA ==> eqA ==> eqA as Aadd_mor.
  ...
End Ring.
```

Ring theory functor

```
Module RingTheory (Export R : Ring).

  Add Setoid A eqA sid_theoryA as A_Setoid.

  Add Ring Aring : Aring.

  Fixpoint power x n {struct n} := ...

  Lemma power_add : forall x n1 n2,
    x ^ (n1 + n2) =A= x ^ n1 * x ^ n2.

  ...
End RingTheory.
```

Notations

RingTheory also provides common notations:

```
Notation "0" := A0.
Notation "1" := A1.
Notation "x + y" := (Aadd x y).
Notation "x * y" := (Amul x y).
Notation "- x" := (Aopp x).
Notation "x - y" := (Asub x y).
```

Ring of integers

```
Require Import ZArith.

Module Int <: SetA.
  Definition A := Z.
End Int.

Module IntRing <: Ring.
  Add Setoid A eqA sid_theoryA as A_Setoid.
  Definition A0 := 0%Z.
  Definition A1 := 1%Z.
  ...
End IntRing.

Module IntRingTheory := RingTheory IntRing.
```

Ring of rationals

```
Require Import QArith.

Module Rat_Eqset <: Eqset.
  Definition A := Q.
  Definition eqA := Qeq.
End Rat_Eqset.
...

Module RatRing <: Ring.
  Add Setoid A eqA sid_theoryA as A_Setoid.
  Definition A0 := 0#1.
  ...
End RatRing.

Module RatRingTheory := RingTheory RatRing.
```

Outline

1. Goal
2. TRSs and polynomial interpretations
3. Current formalization in CoLoR
4. My contributions
   - A generic interface for rings
   - A generic interface for ordered rings
   - Well-foundedness proof of the δ-ordering on $\mathbb{Q}^+$
   - Improved monotony criteria
5. Conclusion

First solution

```
Module Type OrdRing.
  ...
  Parameter gtA : A -> A -> Prop.
  Notation "x >A y" := (gtA x y) (at level 70).
  Parameter gtA_trans : transitive gtA.
  Parameter gtA_irrefl : irreflexive gtA.
  Parameter bgtA : A -> A -> bool.
  Parameter bgtA_ok : forall x y, bgtA x y = true <-> x >A y.
  Parameter one_gtA_zero : 1 >A 0.
  Parameter add_gtA_mono_r:
   forall x y z, x >A y -> x + z >A y + z.
  Parameter mul_gtA_mono_r:
   forall x y z, z >A 0 -> x >A y -> x * z >A y * z.
End OrdRing.
```

Ordered rings theory

```
Module OrdRingTheory (Export ORT : OrdRing).
  Module Export RT := RingTheory R.
  Definition geA x y := x >A y \/ x =A= y.
  Lemma geA_refl : reflexive geA.
  Lemma geA_trans : transitive geA.
  Lemma power_geA_0 : forall x n, x >=A 0 ->  x ^ n >=A 0.
  ...
End OrderRingTheory.
```

Outline

Problem

Problem: Standard order on $\mathbb{Q}$ or $\mathbb{R}$ not WF

Dershowitz (1979) proposal: use $>_\delta$ with $\delta > 0$

$$x >_\delta y \text{ if } x - y \geq \delta$$

```
Variable delta : Q.
Variable delta_pos : delta > 0.
Definition gtA x y :=  x - y >= delta.
Notation "x >A y" := (gtA x y).
```

Well-foundedness of $>_\delta$ on $\mathbb{Q}^+$

```
Lemma wf_Q_N:
  forall x y, x>=0 -> y>=0 -> x>Ay -> f(x)>f(y)
```

$$f(x) = \lfloor \frac{x}{\delta} \rfloor$$

```
  Definition f (x : Q) : nat := Zabs_nat (f_Z x).
  Definition f_Q (x : Q) : Q := inject_Z (f_Z x).
  Definition f_Z (x : Q) : Z := Qfloor (Qdiv x delta).
```

- $\forall x \in \mathbb{Q}^+, \exists t \in \mathbb{Q}, x = f(x)\delta + t$ and $0 \leq t < \delta$
- $\forall x, y \in \mathbb{Q}^+, \exists t, u \in \mathbb{Q},$
  $x - y = (f(x) - f(y))\delta + t - u$ and $-\delta < t - u < \delta$
- $x >_\delta y \Rightarrow f(x) > f(y)$

Outline

First improvement

Current:

```
Definition pstrong_monotone n (p : poly n) :=
  pweak_monotone p
  /\ forall i (H:i<n), coef (mxi H 1) p >A 0.
```

Improve:

```
Definition pstrong_monotone2 n (p : poly n) :=
  pweak_monotone p
  /\ forall i (H:i<n), exists k, coef (mxi H k) p >A 0.
```

Boolean function for pstrong_monotone2

```
Record nat_lt (n : nat) : Type :=
  mk_nat_lt { val : nat; prf : val < n }.

Definition nats_lt : forall n : nat, list (nat_lt n) := ...
  (* list of numbers smaller than n with the proofs *)

Variable kmax : nat.
Definition bpstrong_monotone2 n (p : poly n) :=
  bcoef_not_neg p
  && existsb
       (fun k =>
          forallb
            (fun x => bgtA (coef (mxi (prf x) k) p) 0)
            (nats_lt n))
     (nfirst kmax).
```

Polynomials of degree 2 with negative coefficients

quadratic polynomial function:

$$f_{\mathbb{N}}(x_1, ..., x_n) = c + \sum_{i=1}^{n} bx_i + \sum_{i=1}^{n} \sum_{j=i}^{n} a_{ij}x_{ij} \in \mathbb{Z}[x_1, ..., x_n].$$

Theorem[Neurauter, 2010]

The function $f_{\mathbb{N}}$ is strictly (weakly) monotone and well-defined iff $c \geq 0$, $a_{ij} \geq 0$ and $b > -a_{ii}$ ($b \geq -a_{ii}$) for all $1 \leq i \leq j \leq n$.

Definition

```
Variables (n: nat) (p: poly n)
  (hyp: forall m:monom n, degree m >= 3 -> coef m p =A= 0).

Definition mxij i (hi:i<n) j (hj:j<n) :=
                              mmult (mxi hi 1) (mxi hj 1).
Definition c := coef (mone n) p.
Definition b i (hi:i<n) := coef (mxi hi 1) p.
Definition a i (hi:i<n) j (hj:j<n) := coef (mxij hi hj) p.

Definition monotone :=
  c >=A 0
  /\ forall j (hj:j<n), b hj >=A - a hj hj
    /\ forall i j (hi:i<n) (hj:j<n), a hi hj >=A 0.
```

Deciding: forall i, i<n -> P i

```
Variable P: nat -> Prop.

Definition forall_lt n := forall i, i<n -> P i.

Variables (bP: nat -> bool)
          (bP_ok: forall i, bP i = true <-> P i).
Fixpoint bforall_lt n :=
  match n with
  | O => true
  | S n' => bP n' && bforall_lt n'
  end.

Lemma bforall_lt_ok :forall n,
  bforall_lt n = true <-> forall_lt n.
```

Deciding: forall j (hj:j<n), b hj >=A - a hj hj

```
Definition P j := forall (hj: j<n), b hj >=A - a hj hj.

Definition bP j :=
   match lt_ge_dec j n with
     | left hj => bgeA (b hj) (- a hj hj)
     | _ => true
   end.

Lemma bP_ok : forall j, bP j = true <-> P j.
```

forall i j (hi:i<n) (hj:j<n), a hi hj >=A 0

```
Definition R j:= forall (hi: i<n)(hj: j<n), a hi hj >=A 0.

Definition bR j :=
  match lt_ge_dec i n, lt_ge_dec j n with
  | left hi, left hj => bnot_neg (a hi hj)
  | _, _ => true
  end.

Lemma bR_ok : forall j, bR j = true <-> R j.

Definition Q i := forall_lt (R i) n.
Definition bQ i := bforall_lt (bR i) n.

Lemma bQ_ok : forall i, bQ i = true <-> Q i.
```

Deciding monotony

```
Definition monotone :=
     c >=A 0
     /\ forall j (hj: j<n), b hj >=A - a hj hj
       /\ forall i (hi: i<n) j (hj: j<n), a hi hj >=A 0.

Definition monotone' :=
  not_neg c /\ forall_lt P n /\ forall_lt Q n.

Lemma monotone_eq' : monotone <-> monotone'.

Definition bmonotone :=
  bnot_neg c && bforall_lt bP n && bforall_lt bQ n.

Lemma bmonotone_ok : bmonotone = true <-> monotone.
```

Conclusion

formalized:

- generic interface for rings and ordered rings
- well-foundedness of the δ-ordering on $\mathbb{Q}^+$
- improved the monotony criterion for monotony

Future work: use this Coq development for verifying the correctness of termination certificates in the competition format CPF (Rainbow).

Thank you for your attention!!!