Rings with Explicit Divisibility Formalized in Coq

Anders Mörtberg
University of Gothenburg

Cyril Cohen
INRIA Saclay – Île-de-France
Microsoft Research – INRIA Joint Center
LIX – École Polytechnique

September 9, 2011

Goal

- Formalize constructive algebra in Coq
- Executable within Coq - can be used for computation in proofs

Overview

- Rings with explicit divisibility
  - GCD rings
  - Bezout rings
  - Euclidean rings
- Some polynomial theory
- Smith normal form
  - Constructive PIDs
- Standard examples: $\mathbb{Z}$ and $k[x]$ where k is a field

Formalization

- Formalized using the SSReflect extension to Coq
- Based on chapter 4, Divisibility in discrete domains, in *A course in constructive algebra* by Mines, Richman and Ruitenburg

Rings with explicit divisibility

- A ring R has *explicit divisibility* if it has a divisibility test that give witnesses:

$$a \mid b \leftrightarrow \exists x. b = xa$$

DvdRing

```
CoInductive div_spec (a b : R) : option R -> Type :=
| Dvd x of a = x * b : div_spec a b (Some x)
| NDvd of (forall x, a != x * b) : div_spec a b None.

Record mixin_of (R : ringType) : Type := Mixin {
  div : R -> R -> option R;
  _ : forall a b, div_spec a b (div a b)
}.
```

GCD rings

- A ring R is a *GCD ring* if every pair of elements have a greatest common divisor

$$\forall ab.\exists g.(g \mid a) \land (g \mid b) \land (\forall g'.g' \mid a \land g' \mid b \rightarrow g' \mid g)$$

GcdRing

```
Record mixin_of (R : dvdRingType) : Type := Mixin {
  gcd : R -> R -> R;
  _ : forall a b g,
    g %| gcd a b = (g %| a) && (g %| b)
}.
```

Bezout rings

- Non-Noetherian analogue of principal ideal domains
- Principal ideal domains: Every ideal is principal
- Bezout ring: Every *finitely generated* ideal is principal
- Equivalent definition:

$$\forall ab.\exists xy.ax + by = gcd(a, b)$$

BezoutRing

```
CoInductive bezout_spec (a b : R) : R * R -> Type :=
  BezoutSpec x y of
   gcdr a b %= x * a + y * b : bezout_spec a b (x, y)

Record mixin_of (R : gcdRingType) : Type := Mixin {
  bezout : R -> R -> (R * R);
   _ : forall a b, bezout_spec a b (bezout a b)
}.
```

Euclidean rings

- Euclidean norm: $f : R \to \mathbb{N}$
- Euclidean division: $\forall ab.\exists qr.a = bq + r$ and either $f(r) < f(b)$ or $r = 0$.
- Examples: $\mathbb{Z}$ with absolute value and $k[x]$ with degree

Some polynomial theory

- If R has explicit divisibility then $R[x]$ also has
- If R is a GCD ring then $R[x]$ also is a GCD ring
- This proof is based on the presentation in *The Art of Computer Programming* by Knuth and **it doesn't use the field of fractions** of R as in Mines, Richman, Ruitenburg
- Give GCD algorithm for $\mathbb{Z}[x_1, \ldots, x_n]$ and $k[x_1, \ldots, x_n]$

Smith normal form

- Generalization of Gauss elimination algorithm to work over any principal ideal domain instead of field
- Given a matrix M compute invertible matrices L and R such that LMR is diagonal and $a_{ii} \mid a_{(i+1)(i+1)}$
- Motivation: Computation of homology groups of simplicial complexes from algebraic topology

Constructive PIDs

- In order to formalize the Smith normal form algorithm we need constructive approximation of principal ideal domains
- Mines, Richman, Ruitenburg: A *constructive PID* is a Bezout domain such that if we have a sequence of $u(n)$:s with $u(n+1) \mid u(n)$ then there is some k such that $u(k) \mid u(k+1)$
- Formalized in type theory by having that strict divisibility is well founded

Constructive PIDs

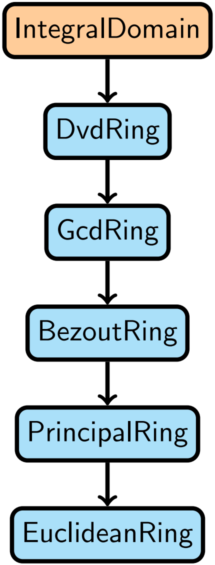
```
Definition sdvdr (R : dvdRingType) (x y : R) :=
  (x %| y) && ~~(y %| x).

Record mixin_of (R : dvdRingType) : Type := Mixin {
  _ : well_founded (@sdvdr R)
}.
```

Constructive PIDs

- This has been used to implement GCD algorithm showing that constructive PIDs are GCD rings
- Vincent Siles used this to implement Smith normal form algorithm in SSReflect

Overview


IntegralDomain
DvdRing
GcdRing
BezoutRing
PrincipalRing
EuclideanRing

Examples

- Have proved that $\mathbb{Z}$ and $k[x]$ where k is a field are Euclidean rings and hence the other structures as well

```
> Time Eval compute in (gcdr 11466 1428)%Z.
     = 42%Z
Finished transaction in 0. secs (0.109993u,0.s)

> Time Eval compute in (123123 %/ 1234)%Z.
     = 99%Z
Finished transaction in 0. secs (0.013333u,0.s)
```

Future work

- Efficient implementation of polynomials
- Implement executable version of Smith normal form algorithm
- Certified computation of homology groups of simplicial complexes

Questions?

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