Coherent and Strongly Discrete Rings in Type Theory

Thierry Coquand, **Anders Mörtberg** and Vincent Siles

Department of Computer Science and Engineering – Chalmers University of Technology and University of Gothenbug, Sweden

December 15 – CPP 2012

Introduction

$$\begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix}^T \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$$

Introduction

$$\begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix}^T \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \end{pmatrix}$$

Introduction

```
A=[ 0 2 ; 1 0 ];
b=[ 2 ; 0 ];
A'\b
```

Introduction

```
A=[ 0 2 ; 1 0 ];
b=[ 2 ; 0 ];
A'\b

ans =
0
1
```

Introduction

$$\begin{pmatrix} 0 \\ 2 \end{pmatrix} \neq \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Introduction

- *"The bug seems to be new in release R2009b and applies to Linux and Windows 32 and 64 bit versions."*
  `http://www.netlib.org/na-digest-html/09/v09n48.html`

- *A serious bug in Matlab2009b?*
  `http://www.walkingrandomly.com/?p=1964`

Introduction

Goal: Formally verified algorithms for solving (in)homogeneous systems of equations over *commutative rings* in Coq/SSReflect

Motivation: Simplifying systems of differential equations, computing homology groups...

Coherent rings

For every row matrix $M \in R^{1\times m}$ there exists $L \in R^{m\times n}$ such that

$$MX = 0 \leftrightarrow \exists Y . X = LY$$

L generate the module of solutions for $MX = 0$

Coherent rings

Theorem: Possible to solve $MX = 0$ where $M \in R^{m \times n}$, i.e. we can compute L such that

$$MX = 0 \leftrightarrow \exists Y . X = LY$$

Constructive proof $\Rightarrow$ algorithm computing generators of solutions

Coherent rings

Theorem: Possible to solve $MX = 0$ where $M \in R^{m \times n}$, i.e. we can compute L such that

$$MX = 0 \leftrightarrow \exists Y.X = LY$$

Constructive proof $\Rightarrow$ algorithm computing generators of solutions

Constructive proof $=$ Program $+$ Specification $+$ Correctness proof

Coherent rings

```
Fixpoint solve m n : forall (M : 'M_(m,n)),
  'M_(n,size_solve M) := match m with
    | S p => fun (M : 'M_(1+p,n)) =>
      let G1 := solve_row (usubmx M) in
      G1 *m solve (dsubmx M *m G1)
    | _ => fun _ => 1%:M
  end.

Lemma solveP m n (M : 'M[R]_(m,n)) (X : 'cV[R]_n) :
 reflect (exists Y, X = solve M *m Y) (M *m X == 0).
Proof.
...
Qed.
```

Coherent rings

How do we prove that R is coherent?

Coherent rings

How do we prove that R is coherent?

Theorem: If $I \cap J$ is finitely generated for finitely generated ideals I and J in R then R is coherent.

Coherent rings

How do we prove that R is coherent?

Theorem: If $I \cap J$ is finitely generated for finitely generated ideals I and J in R then R is coherent.

Problems: How do we represent $I \cap J$?

Coherent rings

How do we prove that R is coherent?

Theorem: If $I \cap J$ is finitely generated for finitely generated ideals I and J in R then R is coherent.

Problems: How do we represent $I \cap J$? How do we represent ideals?

Strongly discrete rings

There exists an algorithm testing if $x \in I$ for finitely generated ideal I and if this is the case produce a witness.

Strongly discrete rings

There exists an algorithm testing if $x \in I$ for finitely generated ideal I and if this is the case produce a witness.

That is, if $I = (x_1, \dots, x_n)$ compute $w_1, \dots, w_n$ such that

$$x = x_1 w_1 + \cdots + x_n w_n$$

Strongly discrete rings

- Suitable for developing ideal theory in type theory:
  - Decidable ideal membership: $x \in I$
  - Decidable ideal inclusion: $I \subseteq J$
  - Decidable ideal equality: $I = J \leftrightarrow I \subseteq J \wedge J \subseteq I$

- Key idea: Represent *finitely generated* ideals as row matrices

Ideals

```
Definition subid (I : 'rV[R]_m) (J : 'rV[R]_n) := ...

Notation "I <= J" := (subid I J).

Lemma subidP (I : 'rV[R]_m) (J : 'rV[R]_n) :
  reflect (exists D, I = J *m D) (I <= J)%IS.

Definition addid (I : 'rV[R]_m) (J : 'rV[R]_n) :=
  row_mx I J.

Notation "I +i J" := (addid I J).

Lemma subid_addidC (I : 'rV[R]_m) (J : 'rV[R]_n) :
  (I +i J <= J +i I)%IS.
```

$I \cap J$ revisited

- $I \cap J$ can be defined as an ideal such that:

$$I \cap J \subseteq I$$

$$I \cap J \subseteq J$$

$$\forall x.\ (x \in I \wedge x \in J) \rightarrow x \in I \cap J$$

- Now we can prove (*constructively*) that if $I \cap J$ is finitely generated then R is coherent

Coherent strongly discrete rings

Recap:

- R is *coherent* if we can find generators for solutions of $MX = 0$
- R is *strongly discrete* if we can decide if $x \in I$ for a finitely generated ideal I in R

Theorem: If R is coherent and strongly discrete we can decide if a system $MX = A$ has a solution

Coherent rings

Examples of coherent rings:

- Fields – Gaussian elimination
- Bézout domains – $\mathbb{Z}$, $\mathbb{Q}[x]$, ...
- Prüfer domains – $\mathbb{Z}[\sqrt{-5}]$, $\mathbb{Q}[x,y]/(y^2-1+x^4)$, ...
- Polynomial rings – $k[x_1, \ldots, x_n]$ via Gröbner bases
- ...

Coherent rings

Examples of coherent rings:

- Fields – Gaussian elimination
- **Bézout domains** – $\mathbb{Z}$, $\mathbb{Q}[x]$, $\ldots$
- **Prüfer domains** – $\mathbb{Z}[\sqrt{-5}]$, $\mathbb{Q}[x, y]/(y^2 - 1 + x^4)$, $\ldots$
- Polynomial rings – $k[x_1, \ldots, x_n]$ via Gröbner bases
- ...

Bézout domains

- Bézout domains: Every *finitely generated* ideal is principal, i.e. for all finitely generated ideals I there is $a \in R$ such that $I = (a)$
- Equivalent definition:

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

Bézout domains

Theorem: Bézout domains (with explicit divisibility) are strongly discrete

Theorem: Bézout domains are coherent

Get algorithm for solving $MX = A$ over $\mathbb{Z}$ and $k[x]$

Prüfer domains

- First-order characterization:

$$\forall ab. \exists uvw.\ ua = vb \wedge (1-u)b = wa$$

- Has many nice ideal properties
- Examples: Bézout domains, algebraic numbers ($\mathbb{Z}[\sqrt{-5}]$), algebraic curves ($\mathbb{Q}[x,y]/(y^2-1+x^4)$), ...

Prüfer domains

Theorem: Prüfer domains (with explicit divisibility) are strongly discrete

Theorem: Prüfer domains are coherent

Get algorithm for solving system $MX = A$ over Prüfer domains

Computation in Coq

- Matrices in SSReflect are represented as:

```
Inductive matrix R m n :=
  Matrix of {ffun 'I_m * 'I_n -> R}.
```

- Well suited for proofs, but not for computation...

Computation in Coq

- Matrices in SSReflect are represented as:

```
Inductive matrix R m n :=
  Matrix of {ffun 'I_m * 'I_n -> R}.
```

- Well suited for proofs, but not for computation...
- Solution: Data refinements

Data refinements

```
'M[R]_(m,n)
```

```
seq (seq R)
```

Data refinements

$$\begin{tikzcd}
\texttt{'M[R]_(m,n)} \arrow[d, "\texttt{seqmx_of_mx}"] \\
\texttt{seq (seq R)}
\end{tikzcd}$$

Data refinements

$$\begin{tikzcd}
\begin{array}{c} \texttt{'M[R]_(m,n)} \\ +,\ \times,\ \cdot^{T},\ldots \end{array} \arrow[d, "\texttt{seqmx_of_mx}"] \\
\begin{array}{c} \texttt{seq (seq R)} \\ \texttt{addseqmx,} \\ \texttt{mulseqmx,} \\ \texttt{trseqmx},\ \ldots \end{array}
\end{tikzcd}$$

Computations

Solve

$$\begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix}^T \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$$

Computations

Solve

$$\begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix}^T \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$$

```
Eval compute in
  csolveGeneral 2 2
    (trseqmx [::[:: 0; 2];[:: 1; 0]])
     [::[:: 2];[:: 0]].
  = Some [:: [:: 0]; [:: 2]]
```

Summary and conclusions

- Implementation of formally verified algorithms for solving systems of equations over:
  - Bézout domains: $\mathbb{Z}$, $\mathbb{Q}[x]$, ...
  - Prüfer domains: $\mathbb{Z}[\sqrt{-5}]$, $\mathbb{Q}[x,y]/(y^2 - 1 + x^4)$,...
- Using data refinements to implement efficient algorithms works well
- CoqEAL[1] – The Coq Effective Algebra Library

[1] http://www-sop.inria.fr/members/Maxime.Denes/coqeal/

Thank you!

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

Extra slides

Bézout domains: Coherent

Theorem: Bézout domains are coherent

- It suffices to show that $I \cap J$ is finitely generated for finitely generated ideals I and J.
- Compute a and b such that $I = (a)$ and $J = (b)$.
- Can prove $I \cap J = (lcm(a, b))$

Prüfer domains

Theorem: Every nonzero finitely generated ideal is *invertible*

Given a finitely generated ideal I over R there exists I^{-1} such that $II^{-1} = (a)$ for some $a \in R$

Coherence of Prüfer domains

Theorem: Prüfer domains are coherent

- Given finitely generated I and J we have $(I + J)(I \cap J) = IJ$
- To compute $I \cap J$ invert $I + J$ and multiply on both sides:

$$(a)(I \cap J) = (I + J)^{-1} IJ$$

- Hence we can compute $I \cap J$

Optimizations

- We can also do program refinements to optimize our algorithms
- Simple example: Ideal addition
  - Problems with naive implementation: Zeroes? Duplicate elements?
  - Solution: Implement more efficient version $+'$ such that $I + J = I +' J$ and then refine $+'$ to lists

Future work

- Polynomial rings – $k[x_1, \ldots, x_n]$ via Gröbner bases
- Implement library of homological algebra – HOMALG project[2]

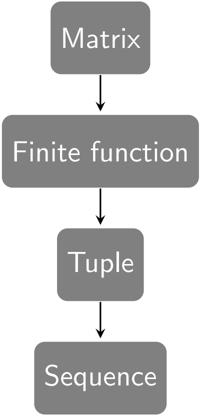
[2] `http://homalg.math.rwth-aachen.de/`

SSReflect matrices

```
Inductive matrix R m n :=
  Matrix of {ffun 'I_m * 'I_n -> R}.
```

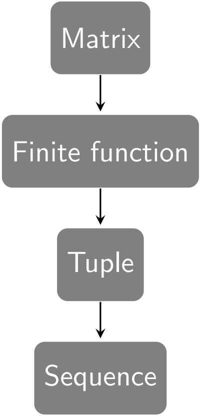
SSReflect matrices

```
Inductive matrix R m n :=
  Matrix of {ffun 'I_m * 'I_n -> R}.
```



SSReflect matrices

```
Inductive matrix R m n :=
  Matrix of {ffun 'I_m * 'I_n -> R}.
```



- Fine-grained architecture
- Proof-oriented design
- Had to be locked to avoid term size explosion
- Not suited for computation

Objective

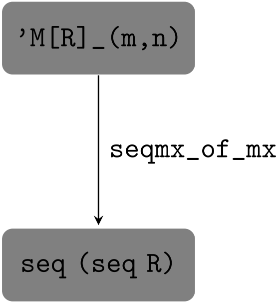
- Define concrete and executable representations and operations on matrices, using a relaxed datatype (unsized lists)
- Devise a way to link them with the theory in SSReflect
- Still be able to use convenient tools to reason about the algorithms we implement

Methodology

`'M[R]_(m,n)`

`seq (seq R)`

Methodology



Methodology

'M[R]_(m,n)

$+, \times, .^T, \ldots$

$\big\downarrow$ seqmx_of_mx

seq (seq R)

addseqmx,

mulseqmx,

trseqmx, ...

List-based representation of matrices

```
Variable R : ringType.
Definition seqmatrix := seq (seq R).
```

List-based representation of matrices

```
Variable R : ringType.
Definition seqmatrix := seq (seq R).

Definition seqmx_of_mx (M : 'M_(m,n)) : seqmatrix :=
  map (fun i => map (fun j => M i j) (enum 'I_n))
      (enum 'I_m).
```

List-based representation of matrices

```
Variable R : ringType.
Definition seqmatrix := seq (seq R).

Definition seqmx_of_mx (M : 'M_(m,n)) : seqmatrix :=
  map (fun i => map (fun j => M i j) (enum 'I_n))
     (enum 'I_m).

Lemma seqmx_eqP (M N : 'M_(m,n)) :
  reflect (M = N) (seqmx_of_mx M == seqmx_of_mx N).
```

List-based representation of matrices

```
Variable R : ringType.
Definition seqmatrix := seq (seq R).

Definition seqmx_of_mx (M : 'M_(m,n)) : seqmatrix :=
  map (fun i => map (fun j => M i j) (enum 'I_n))
     (enum 'I_m).

Lemma seqmx_eqP (M N : 'M_(m,n)) :
  reflect (M = N) (seqmx_of_mx M == seqmx_of_mx N).

Definition addseqmx (M N : seqmatrix) : seqmatrix :=
  zipwith (zipwith (fun x y => x + y)) M N.

Lemma addseqmxE (M N : 'M[R]_(m,n)) :
  seqmx_of_mx (M + N) =
    addseqmx (seqmx_of_mx M) (seqmx_of_mx N).
```

Methodology

$$\begin{tikzcd} \begin{array}{c}\texttt{'M[R]\_(m,n)}\\ +,\ \times,\ .^{T},\ldots\end{array} \arrow[d, "\texttt{seqmx\_of\_mx}"] \\ \begin{array}{c}\texttt{seq (seq R)}\\ \texttt{addseqmx,}\\ \texttt{mulseqmx,}\\ \texttt{trseqmx,}\ \ldots\end{array}\end{tikzcd}$$

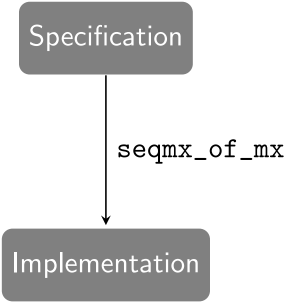
Methodology

Specification

↓ `seqmx_of_mx`

`seq (seq R)`
`addseqmx,`
`mulseqmx,`
`trseqmx, ...`

Methodology



Methodology