A Formal Proof of Sasaki-Murao Algorithm
(jww. Thierry Coquand and Vincent Siles)

Anders Mörtberg – mortberg@chalmers.se

September 20, 2012

Introduction

- Want: Polynomial time algorithm for computing the determinant of a matrix with coefficients in any commutative ring (not necessarily with division)
- Formally verified implementation in Coq

Naive algorithm

- Laplace expansion: Express the determinant in terms of determinants of submatrices

$$\begin{aligned}\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} &= a\begin{vmatrix} e & f \\ h & i \end{vmatrix} - b\begin{vmatrix} d & f \\ g & i \end{vmatrix} + c\begin{vmatrix} d & e \\ g & h \end{vmatrix} \\ &= a(ei - hf) - b(di - fg) + c(dg - eg) \\ &= aei - ahf - bdi + bfg + cdg - ceg\end{aligned}$$

- Not suited for computation (factorial complexity)

Gaussian elimination

- Gaussian elimination: Convert the matrix into triangular form using elementary operations
- Polynomial time algorithm
- Relies on division – Is there a polynomial time algorithm not using division?

Division free Gaussian algorithm

- We want an operation doing:

$$\begin{pmatrix} a & l \\ c & M \end{pmatrix} \rightsquigarrow \begin{pmatrix} a & l \\ 0 & M' \end{pmatrix}$$

Division free Gaussian algorithm

- We want an operation doing:

$$\begin{pmatrix} a & l \\ c & M \end{pmatrix} \rightsquigarrow \begin{pmatrix} a & l \\ 0 & M' \end{pmatrix}$$

- We can do:

$$\begin{pmatrix} a & l \\ c & M \end{pmatrix} \rightsquigarrow \begin{pmatrix} a & l \\ ac & aM \end{pmatrix} \rightsquigarrow \begin{pmatrix} a & l \\ 0 & aM - cl \end{pmatrix}$$

Division free Gaussian algorithm

- We want an operation doing:

$$\begin{pmatrix} a & l \\ c & M \end{pmatrix} \rightsquigarrow \begin{pmatrix} a & l \\ 0 & M' \end{pmatrix}$$

- We can do:

$$\begin{pmatrix} a & l \\ c & M \end{pmatrix} \rightsquigarrow \begin{pmatrix} a & l \\ ac & aM \end{pmatrix} \rightsquigarrow \begin{pmatrix} a & l \\ 0 & aM - cl \end{pmatrix}$$

- Problems:
  - Computes $a^n \cdot det$
  - $a = 0$?
  - Exponential growth of coefficients

Bareiss algorithm

- Erwin Bareiss: *"Sylvester's Identity and Multistep Integer-Preserving Gaussian Elimination"* (1968)
- Compute determinant of integer matrices in polynomial time
- Only do divisions that are guaranteed to be exact

Bareiss algorithm: Example

$$\begin{pmatrix} 2 & 2 & 4 & 5 \\ 5 & 8 & 9 & 3 \\ 1 & 2 & 8 & 5 \\ 6 & 6 & 7 & 1 \end{pmatrix}$$

Bareiss algorithm: Example

$$\begin{pmatrix} 2 & 2 & 4 & 5 \\ 5 & 8 & 9 & 3 \\ 1 & 2 & 8 & 5 \\ 6 & 6 & 7 & 1 \end{pmatrix} \rightsquigarrow \begin{pmatrix} 2 & 2 & 4 & 5 \\ 0 & 2*8-5*2 & 2*9-5*4 & 2*3-5*5 \\ 0 & 2*2-1*2 & 2*8-1*4 & 2*5-1*5 \\ 0 & 2*6-6*2 & 2*7-6*4 & 2*1-6*5 \end{pmatrix}$$

Bareiss algorithm: Example

$$= \begin{pmatrix} 2 & 2 & 4 & 5 \\ 0 & 6 & -2 & -19 \\ 0 & 2 & 12 & 5 \\ 0 & 0 & -10 & -28 \end{pmatrix}$$

Bareiss algorithm: Example

$$\rightsquigarrow \begin{pmatrix} 2 & 2 & 4 & 5 \\ 0 & 6 & -2 & -19 \\ 0 & 0 & 6*12-2*(-2) & 6*5-2*(-19) \\ 0 & 0 & 6*(-10)-0*(-2) & 6*(-28)-0*(-19) \end{pmatrix}$$

Bareiss algorithm: Example

$$= \begin{pmatrix} 2 & 2 & 4 & 5 \\ 0 & 6 & -2 & -19 \\ 0 & 0 & 76 & 68 \\ 0 & 0 & -60 & -168 \end{pmatrix}$$

Bareiss algorithm: Example

$$\rightsquigarrow' \begin{pmatrix} 2 & 2 & 4 & 5 \\ 0 & 6 & -2 & -19 \\ 0 & 0 & 76/2 & 68/2 \\ 0 & 0 & -60/2 & -168/2 \end{pmatrix}$$

Bareiss algorithm: Example

$$= \begin{pmatrix} 2 & 2 & 4 & 5 \\ 0 & 6 & -2 & -19 \\ 0 & 0 & 38 & 34 \\ 0 & 0 & -30 & -84 \end{pmatrix}$$

Bareiss algorithm: Example

$$\rightsquigarrow \begin{pmatrix} 2 & 2 & 4 & 5 \\ 0 & 6 & -2 & -19 \\ 0 & 0 & 38 & 34 \\ 0 & 0 & 0 & 38*(-84)-(-30)*34 \end{pmatrix}$$

Bareiss algorithm: Example

$$= \begin{pmatrix} 2 & 2 & 4 & 5 \\ 0 & 6 & -2 & -19 \\ 0 & 0 & 38 & 34 \\ 0 & 0 & 0 & -2172 \end{pmatrix}$$

Bareiss algorithm: Example

$$\rightsquigarrow' \begin{pmatrix} 2 & 2 & 4 & 5 \\ 0 & 6 & -2 & -19 \\ 0 & 0 & 38 & 34 \\ 0 & 0 & 0 & -2172/6 \end{pmatrix}$$

Bareiss algorithm: Example

$$= \begin{pmatrix} 2 & 2 & 4 & 5 \\ 0 & 6 & -2 & -19 \\ 0 & 0 & 38 & 34 \\ 0 & 0 & 0 & -362 \end{pmatrix}$$

Bareiss algorithm: Example

$$\begin{vmatrix} 2 & 2 & 4 & 5 \\ 5 & 8 & 9 & 3 \\ 1 & 2 & 8 & 5 \\ 6 & 6 & 7 & 1 \end{vmatrix} = -362$$

Bareiss algorithm

- $a = 0$?
- Generalize to any commutative ring?

Bareiss algorithm

- $a = 0$?
- Generalize to any commutative ring?
  - No, we need explicit divisibility
  - Examples: $\mathbb{Z}, k[x], \mathbb{Z}[x, y], k[x, y, z], \ldots$

Sasaki-Murao algorithm

- Apply the algorithm to $x \cdot Id - M$
- Compute on $R[x]$ with pseudo-division
- Put $x = 0$ in the result

Sasaki-Murao algorithm

```
dvd_step :: R[x] -> Matrix R[x] -> Matrix R[x]
dvd_step g M = mapM (\x -> g | x) M

sasaki_rec :: R[x] -> Matrix R[x] -> R[x]
sasaki_rec g M = case M of
  Empty -> g
  Cons a l c M ->
     let M' = a * M - c * l in
     sasaki_rec a (dvd_step g M')

sasaki :: Matrix R -> R[x]
sasaki M = sasaki_rec 1 (x * Id - M)
```

Sasaki-Murao algorithm

- Very simple functional program!
- No problem with 0 (x along the diagonal)
- Get characteristic polynomial for free
- Works for any commutative ring
- Standard correctness proof is complicated – relies on Sylvester identities

Sasaki-Murao algorithm: Correctness proof

Invariant for recursive call (`sasaki_rec g M`):

- g is regular
- g^k divides all $k+1$ minors of M
- All principal minors of M are regular

Some Sylvester identities are corollaries of our proof

Sasaki-Murao algorithm: Computations in Coq

```
Definition M10 := (* Random 10x10 matrix *).

Time Eval vm_compute in sasaki 10 M10.
     = (-406683286186860)%Z
  Finished transaction in 1. secs (1.316581u,0.s)

Definition M20 := (* Random 20x20 matrix *).

Time Eval vm_compute in sasaki 20 M20.
     = 75728050107481969127694371861%Z
  Finished transaction in 63. secs
    (62.825904u,0.016666s)
```

Sasaki-Murao algorithm: Computations with Haskell

```
> time ./Sasaki 10
-406683286186860

real 0m0.009s

> time ./Sasaki 20
75728050107481969127694371861

real 0m0.267s

> time ./Sasaki 50
-3353887303469... (73 more digits...)

real 1m6.159s
```

Conclusions

- Sasaki-Murao algorithm: Simple functional program for computing determinant over any commutative ring
- (Arguably) Simpler proof and Coq formalization:

  A Formal Proof of Sasaki-Murao Algorithm[1]
  To appear in Journal of Formalized Reasoning

[1] http://www.cse.chalmers.se/~mortberg/papers/det.pdf

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