Parametric solvable polynomial rings and applications

Heinz Kredel, University of Mannheim

CASC 2015, Aachen

Overview

- Introduction
- Solvable Polynomial Rings
  - Parametric Solvable Polynomial Rings
  - Solvable Quotient and Residue Class Rings
  - Solvable Quotient Rings as Coefficient Rings
- Implementation of Solvable Polynomial Rings
  - Recursive Solvable Polynomial Rings
  - Solvable Quotient and Residue Class Rings
- Applications
- Conclusions

Introduction

- solvable polynomial rings fit between commutative and free non-commutative polynomial rings
- share many properties with commutative case: being Noetherian, tractable by Gröbner bases
- free non-commutative case no more Noetherian, so eventually infinite ideals and non terminating computations
- though, solvable polynomials are not easy to compute either

Introduction (cont.)

- problems have been explored mainly in theory
- solvable polynomials can share representations with commutative polynomials and reuse implementations, "only" multiplication to be done
- implementation is generic in the sense that various coefficient rings can be used in a strongly type safe way and still good performing code
- parametric coefficient rings with commutator relations between variables and coefficient variables new
- solvable quotient ring elements as coefficients new

Related work (selected)

- enveloping fields of Lie algebras [Apel, Lassner]
- solvable polynomial rings [Kandri-Rodi, Weispfenning]
- free-noncommutative polynomial rings [Mora]
- parametric solvable polynomial rings and comprehensive Gröbner bases [Weispfenning, Kredel]
- PBW algebras in Singular / Plural [Levandovskyy]
- primary ideal decomposition [Gomez-Torrecillas]

Solvable Polynomial Rings

Solvable polynomial ring S: associative Ring (S,0,1,+,-,*), K a (skew) field, in n variables

$$S = \mathbf{K}\{X_1, \ldots, X_n; Q; Q'\}$$

commutator relations between variables, $lt(p_{ij}) < X_i X_j$

$$Q = \{X_j * X_i = c_{ij} X_i X_j + p_{ij} : 0 \neq c_{ij} \in \mathbf{K}, X_i X_j > p_{ij} \in S, 1 \leq i < j \leq n\}$$

commutator relations between variables and coefficients

$$Q' = \{X_i * a = c_{ai} a X_i + p_{ai} : 0 \neq c_{ai} \in \mathbf{K}, p_{ai} \in \mathbf{K}, 1 \leq i \leq n, a \in \mathbf{K}\}$$

< a *-compatible term order on S x S: a < b ⇒ a*c < b*c and c*a < c*b for a, b, c in S

Parametric Solvable Polynomial Rings

$$S = \mathbf{R}[U_1, \ldots, U_m]\{X_1, \ldots, X_n; Q\}$$

domain R, parameters U, variables X_i, Q' empty

Lemma 1 (7.1.2 in [12]). *Let* $\mathbf{R}$ *be a commutative Noetherian domain,* $m \in \mathbb{N}$*,* $R = \mathbf{R}[U_1, \ldots, U_m]$*. Let* $S = R\{X_1, \ldots, X_n; Q\}$ *be a parametric solvable polynomial ring as defined in Axioms 7.1.1 in [12] with respect to a* $*$*-compatible term order* $<$*. Let* C *be the multiplicative subset of* R *generated by the* c_{ij} *from the commutator relations* Q*. Then for* $0 \neq f, g \in S$ *one can compute* $0 \neq c \in C$ *and* $p \in S$ *with* $p < f \cdot g$ *such that*

$$f * g = c \cdot f \cdot g + p.$$

c *and* p *are uniquely determined by these properties and the coefficients of* p *in* R *are polynomials in the* c_{ij}*, the coefficients of all* p_{ij} *from the commutator relations* Q *and of the coefficients of* f, g*. Furthermore these polynomials are formed uniformly, independently of the ring* R*.*

Solvable Polynomial Coefficient Rings

$$S = \mathbf{R}\{U_1, \ldots, U_m; Q_u\}\{X_1, \ldots, X_n; Q_x; Q'_{ux}\}$$

$$Q_u = \{\ U_j * U_i = c_{uij} U_i U_j + p_{uij} :$$
$$0 \neq c_{uij} \in \mathbf{R}, U_i U_j > p_{uij} \in R, 1 \leq i < j \leq m\ \}$$

$$Q_x = \{\ X_j * X_i = c_{xij} X_i X_j + p_{xij} :$$
$$0 \neq c_{xij} \in R, X_i X_j > p_{xij} \in S, 1 \leq i < j \leq n\ \}$$

$$Q'_{ux} = \{\ X_j * U_i = c_{ij} U_i X_j + p_{ij} :$$
$$0 \neq c_{ij} \in \mathbf{R}, U_i X_j > p_{ij} \in S, 1 \leq i \leq m, 1 \leq j \leq n\ \}$$

recursive solvable polynomial rings

$$S_k = \mathbf{R}\{X_1, \ldots, X_k; Q_k\}\{X_{k+1}, \ldots, X_n; Q_n; Q'_{kn}\}, \quad 0 \leq k \leq n$$

Solvable Quotient and Residue Class Rings

- solvable quotient rings, skew fields

$$\mathbf{R}(U_1, \ldots, U_m; Q_u)$$

- solvable residue class rings modulo an ideal

$$\mathbf{R}\{U_1, \ldots, U_m; Q_u\}_{/\mathcal{I}}$$

- solvable local ring, localized by an ideal

$$\mathbf{R}\{U_1, \ldots, U_m; Q_u\}_{\mathcal{I}}$$

- solvable quotient and residue class ring modulo an ideal, if ideal completly prime, then skew field

$$\mathbf{R}(U_1, \ldots, U_m; Q_u)_{/\mathcal{I}}$$

Ore condition

- for a, b in R there exist
  - c, d in R with c*a = d*b   left Ore condition
  - c', d' in R with a*c' = b*d'   right Ore condition
- Theorem: Noetherian rings satify the Ore condition
  - left / left and right / right
- can be computed by left respectively right syzygy computations in R [6]
- Theorem: domains with Ore condition can be embedded in a skew field
- a/b * c/d :=: (f*c)/(e*b) where e,f with e*a = f*d

Solvable Quotient and Residue Class Rings as coefficients

$$S = \mathbf{R}(U_1, \ldots, U_m; Q_u)\{X_1, \ldots, X_n; Q_x, Q'_{ux}\}$$

Lemma 2. *Assume* $c_{ij} = 1$ *in* Q'_{ux}. *Let* $x^e * d = dx^e + p$, $dx^e > p \in S$, *then*

$$x^e * \frac{1}{d} = \frac{1}{d}(x^e - (p * \frac{1}{d})).$$

Proof. The identity can be derived under the assumption that all $c_{ij} = 1$ in Q'_{ux}, as follows. If not all $c_{ij} = 1$ with some more care a corresponding factor c as product of c_{ij}'s can be established. Let $z = x^e$, then from $z * \frac{1}{d} * d = z$ and $p = z * d - dz$ together with the assumption $z * \frac{1}{d} = \frac{1}{d}z + q$ for some q, it follows $(\frac{1}{d}z + q) * d = z$ and $\frac{1}{d}(dz + p) + q * d = z$. Multiplying out, we get $\frac{1}{d}d * z + \frac{1}{d}p + q * d = z$ and so $\frac{1}{d}p + q * d = 0$ must hold. Multiplying with $\frac{1}{d}$ from right, we have $\frac{1}{d}p * \frac{1}{d} + q = 0$ and so $q = -\frac{1}{d}p * \frac{1}{d}$. With this, we get $z * \frac{1}{d} = \frac{1}{d}z - \frac{1}{d}p * \frac{1}{d} = \frac{1}{d}(z - p * \frac{1}{d})$. Since by assumption $p < z$ the claim follows by induction as finally $p \in R$ and the multiplication can be carried out in R. $\square$

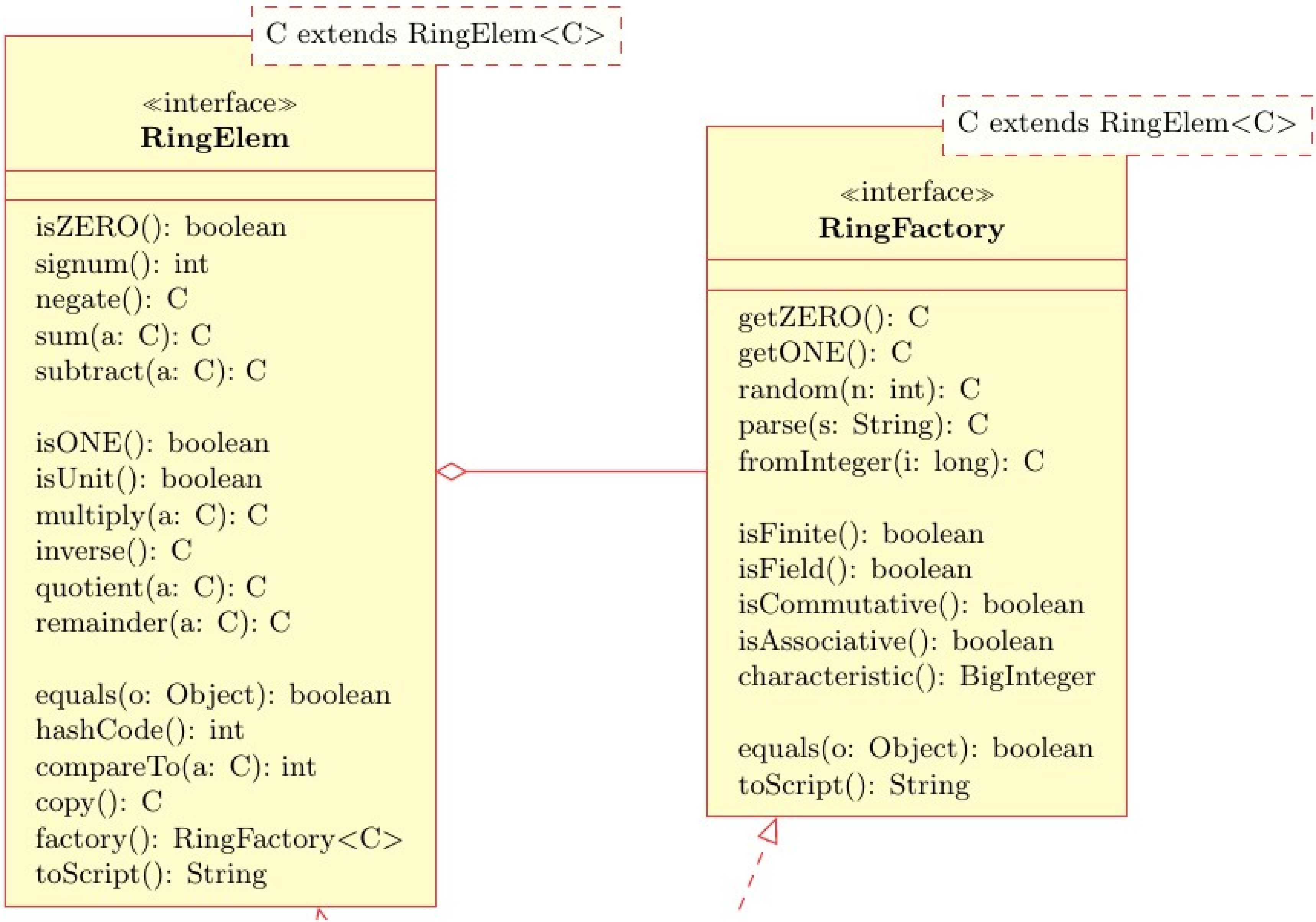
Overview

- Introduction
- Solvable Polynomial Rings
  - Parametric Solvable Polynomial Rings
  - Solvable Quotient and Residue Class Rings
- Implementation of Solvable Polynomial Rings
  - Recursive Solvable Polynomial Rings
  - Solvable Quotient and Residue Class Rings
  - Solvable Quotient Rings as Coefficient Rings
- Applications
- Conclusions

Implementation of Solvable Polynomial Rings

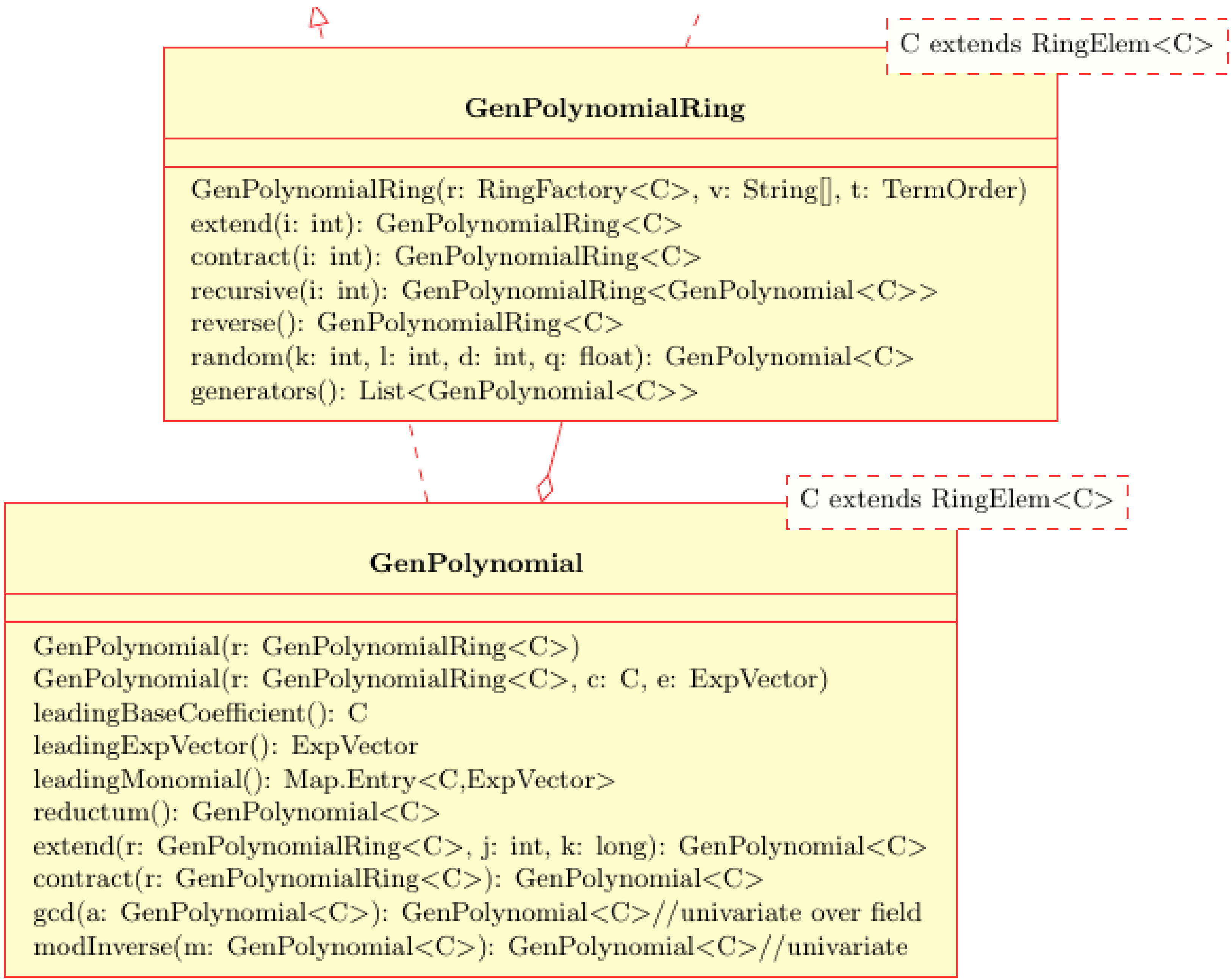
- Java Algebra System (JAS)
- generic type parameters : RingElem<C>
- type safe, interoperable, object oriented
- has greatest common divisors, squarefree decomposition factorization and Gröbner bases
- scriptable with JRuby, Jython and interactive
- parallel multi-core and distributed cluster algorithms
- with Java from Android to Compute Clusters

Ring Interfaces

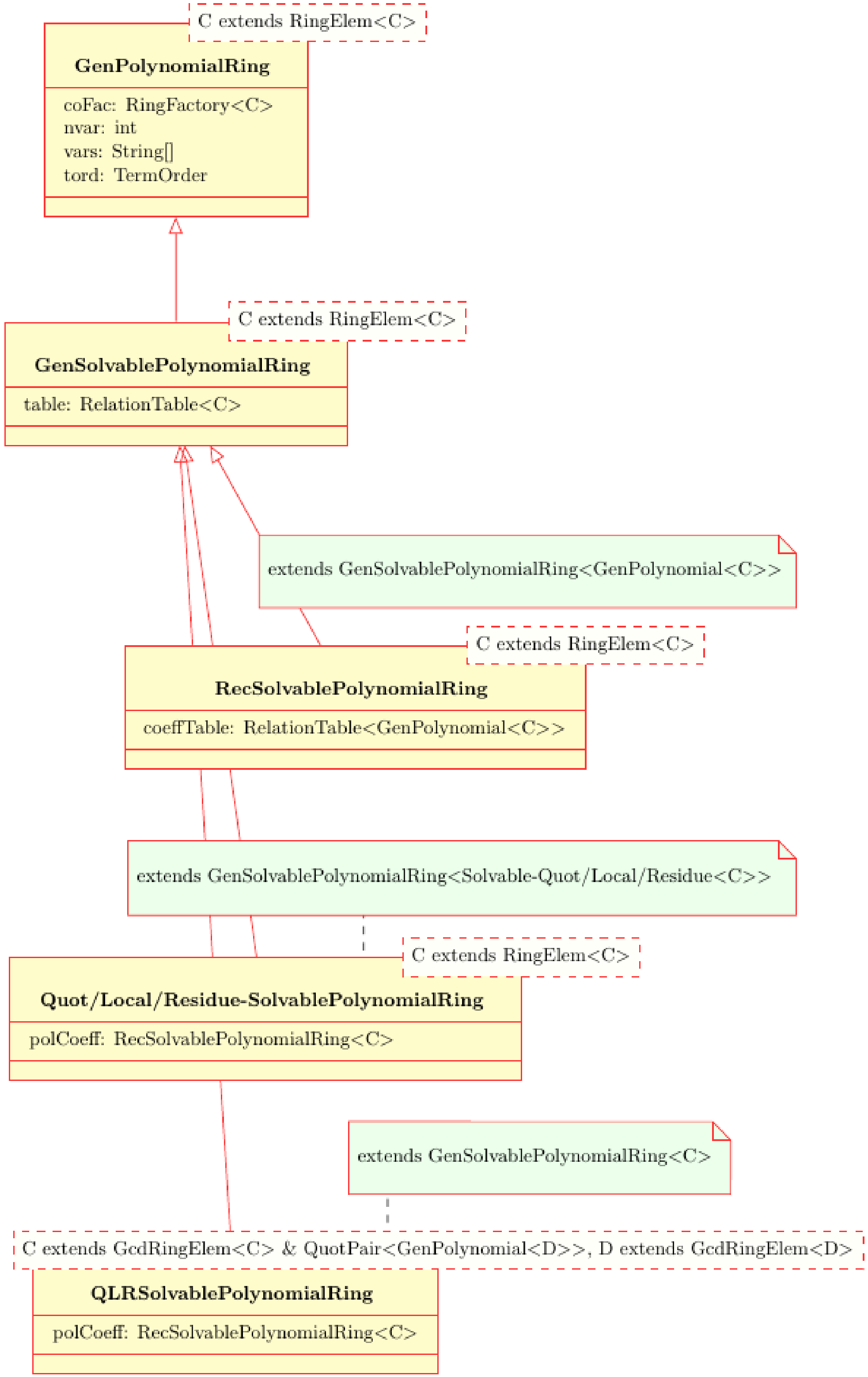


CASC 2015

Generic Polynomial Rings



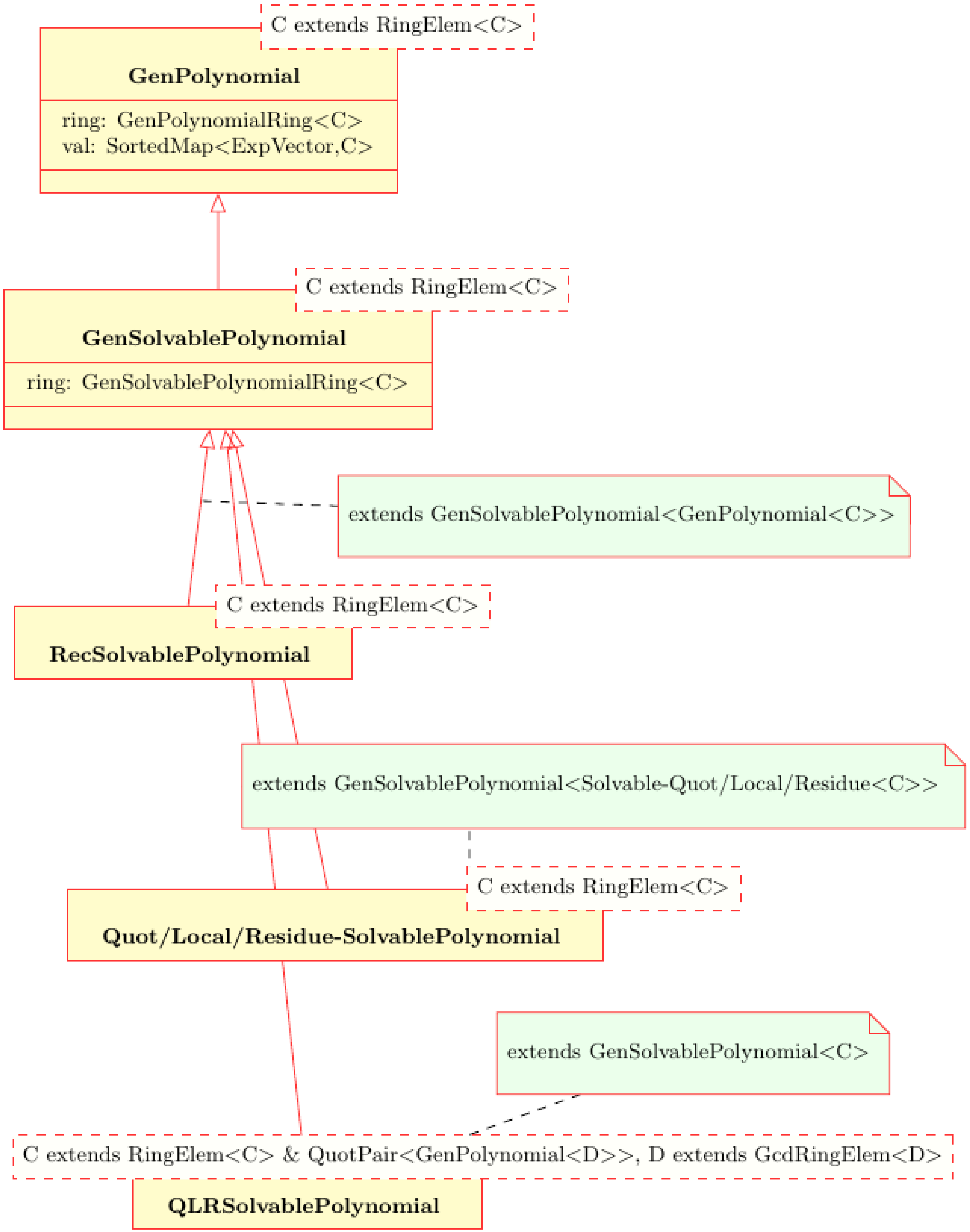
Solvable Polynomial Ring Overview



Polynomial ring implementation

- commutative polynomial ring
  - coefficient ring factory
  - number of variables
  - name of variables
  - term order
- solvable polynomial ring
  - relation table
  - commutator relations: $X_j * X_i = c_{ij} X_i X_j + p_{ij}$
  - missing relations treated as commutative
  - relations for powers are stored for lookup

Solvable Polynomial Overview



Recursive solvable polynomial ring

- implemented in `RecSolvablePolynomial` and `RecSolvablePolynomialRing`
- extends `GenSolvablePolynomial<GenPolynomial<C>>`
- new relation table `coeffTable` for relations from Q'_{ux}, with type `RelationTable<GenPolynomial<C>>`
- recording of powers of relations for lookup instead of recomputation
- new method `rightRecursivePolynomial()` with coefficients on the right side

recursive *-multiplication

1. loop over terms of first polynomial:

   $a\, x^e = a'\, u^{e'}\, x^e$

2. loop over terms of second polynomial:

   $b\, x^f = b'\, u^{f'}\, x^f$

3. compute $(a\, x^e) \ast (b\, x^f)$ as $a \ast ((x^e \ast b) \ast x^f)$

   (a) $x^e \ast b = p_{eb}$, iterate lookup of $x_i \ast u_j$ in Q'_{ux}

   (b) $p_{eb} \ast x^f = p_{ebf}$, iterate lookup of $x_j \ast x_i$ in Q_x

   (c) $a \ast p_{ebf} = p_{aebf}$, in recursive coefficient ring
   lookup $u_j \ast u_i$ in Q_u

4. sum up the p_{aebf}

Solvable Quotient and Residue Rings

1. the solvable quotient ring, $R(U_1 , . . . , U_m; Q_u)$, is implemented by classes SolvableQuotient and SolvableQuotientRing, implements RingElem<.<C>>
2. the solvable residue class ring modulo I, $R\{U_1 , . . . , U_m ; Q_u \}_{/I}$, is implemented by classes SolvableResidue and SolvableResidueRing
3. the solvable local ring, localized by ideal I, $R\{U_1, . . . , U_m; Q_u\}_I$, is implemented by classes SolvableLocal and SolvableLocalRing
4. the solvable quotient and residue class ring modulo I, $R(U_1 , . . . , U_m ; Q_u)_{/I}$, is implemented by classes SolvableLocalResidue and SolvableLocalResidueRing

Implementation of + and *

- Ore condition in `SolvableSyzygy`
  - `leftOreCond()` and `rightOreCond()`
- simplification difficult
  - reduction to lower terms
  - `leftSimplifier()` after [7] using module Gröbner bases of syzygies of quotients
  - require common divisor computation
    - not unique in solvable polynomial rings
  - package `edu.jas.fd`
- very high complexity and (intermediate) expression swell, only small examples feasible

with solvable quotient coefficients

- reuse recursive solvable polynomial multiplication with `polCoeff` ring internally
- extend multiplication to quotients or residues
- class `QLRSolvablePolynomial`, `QLRSolvablePolynomialRing`
- abstract quotient structure, additional to ring element, `QuotPair` and `QuotPairFactory`
- conversion
  - `fromPolyCoefficients()`
  - `toPolyCoefficients()`

*-multiplication with 1/d

- **recursion base, denominator = 1**: $x^e * n/1$. It computes $x^e * n$ from the recursive solvable polynomial ring `polCoeff`, looking up $x^e * n$ in Q'_{ux}, and then converting the result to a polynomial with quotient coefficients
- **recursion base, denominator != 1**: $x^e * 1/d$. Let p be computed by $x^e * d = d\, x^e + p$ then compute $x^e * 1/d$ as $1/d\, (x^e - (p * 1/d))$ by lemma 2. Since $p < x^e$, $p * 1/d$ uses recursion on a polynomial with smaller head term, so the algorithm will terminate
- **numerator != 1**: let $p_{xed} = x^e * 1/d$ and compute $p_{xed} * n/1$ by recursion

Overview

- Introduction
- Solvable Polynomial Rings
- Implementation of Solvable Polynomial Rings
- Applications
  - comprehensive Gröbner bases
  - left, right and two-sided Gröbner bases
  - examples
  - extensions to free non-commutative coefficient rings
- Conclusions

Applications (1)

- Comprehensive Gröbner bases

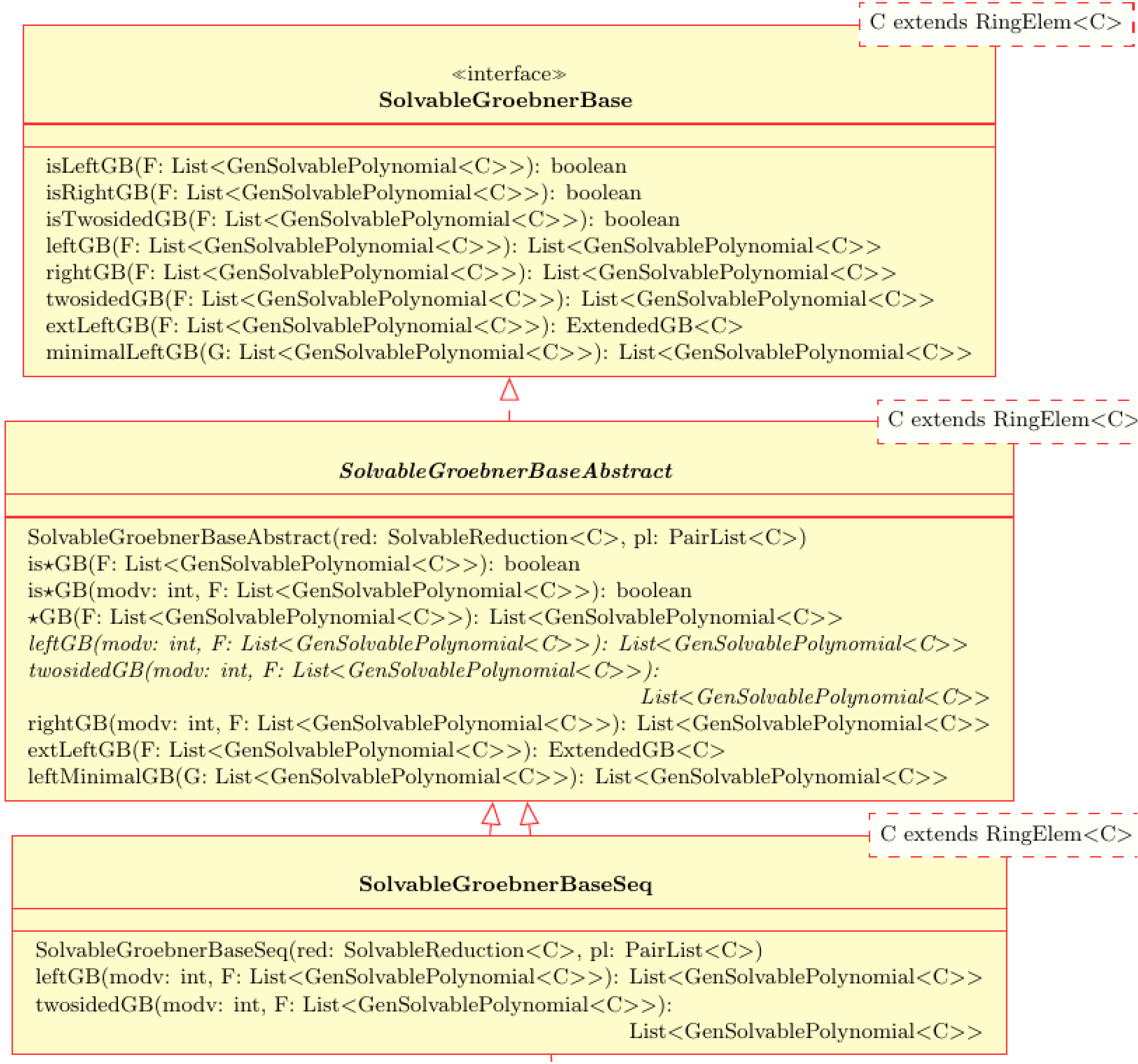
  commutative $S = \mathbf{R}[U_1, \ldots, U_m][X_1, \ldots, X_n]$

  solvable $\mathbf{R}[U_1, \ldots, U_m]\{X_1, \ldots, X_n, Q\}$

  - silght modfication of commutative algorithm works for solvable case: use `multiplyLeft()`
- also commutative transcendental field extension coefficients works
- fraction free coefficients by taking primitive parts work

CASC 2015

Solvable Gröbner bases



Applications (2)

- applications with solvable quotient coefficient
  - verify multiplication by coefficients is correct, so existing algorithms can be reused
  - gives left, right and two-sided Gröbner bases
    - for two-sided case more right multiplications with coefficent generators required
  - gives also left and right syzygies
  - same for left, right and two-sided module Gröbner bases
- recursive solvable polynomials with pseudo reduction using Ore condition to adjust coefficient multipliers

Examples (1)

$$\mathbb{Q}(x,y,z,t;Q_x)_{/\mathcal{I}}\{r;Q_r\}$$

$$Q_x = \{z*y = yz + x, t*y = yt + y, t*z = zt - z\} \qquad Q_r = \emptyset$$

$$\mathcal{I} = (t^2 + z^2 + y^2 + x^2 + 1)$$

Ruby syntax in JAS jRuby interface

```
pcz = PolyRing.new(QQ(),"x,y,z,t")
zrel = [z, y, ( y * z + x ), t, y, ( y * t + y ),
        t, z, ( z * t - z )]
pz = SolvPolyRing.new(QQ(),"x,y,z,t",PolyRing.lex,zrel)
ff = pz.ideal("", [t**2 + z**2 + y**2 + x**2 + 1])
ff = ff.twosidedGB()

SolvIdeal.new(
 SolvPolyRing.new(QQ(),"x,y,z,t",PolyRing.lex,
 rel=[z, y, ( y * z + x ), t, z, ( z * t - z ),
      t, y, ( y * t + y )]),
 "",[x, y, z, ( t**2 + 1 )])
```

Examples (2)

```
construction: SLR(ideal, numerator, denominator)

f0 = SLR(ff, t + x + y + 1)
f1 = SLR(ff, z**2+x+1 )
```

f2 = f1*f0: `z**2 * t + x * t + t + y * z**2 + x * z**2 + z**2 + 2 * x * z + x * y + y + x**2 + 2 * x + 1`

fi = 1/f1: `1 / ( z**2 + x + 1 )`

fi*f1 = f1*fi: `1`

f0*fi: `( x**2 * z * t**2 + ... ) / ( ... + 23 * x + 7 )`

```
( 2 * t**2 + 7 ) / ( 2 * t + 7 )
```

want x, y, z simplified to 0

Examples (3)

```
pt = SolvPolyRing.new(f0.ring, "r", PolyRing.lex)

fr = r**2 + 1
iil = pt.ideal( "", [ fr ] )
rgll = iil.twosidedGB()

SolvIdeal.new(...,[( r**2 + 1 )])

e = fr.evaluate( t )
e: 0
```

```
fp = (r-t)

fr / fp: (r+t)
fr % fp: 0
```

```
frp = fp*(r+t)

frp: ( r**2 - t**2 )
frp-fr: 0
frp == fr: true
```

Examples (4)

$$\mathbb{Q}(x, y, z, t; Q_x)_{/\mathcal{I}}(r; Q_r)_{/(r^2+1)}$$

```
rf = SLR(rgll, r)

rf**2 + 1: 0

ft = SLR(rgll, t)

ft**2 + 1: 0
(rf-ft)*(rf+ft): 0
```

CASC 2015

Extension to free non-commutative polynomial coefficients

Free non-commutative generic polynomial ring K<x,y,z>

implementation in classes `GenWordPolynomial` and `GenWordPolynomialRing`

```
r = WordPolyRing.new(QQ(),"x,y"); one,x,y = r.gens();

f1 = x*y - 1/10;
f2 = y*x + x + y;
ff = r.ideal( "", [f1,f2] ); gg = ff.GB();

WordPolyIdeal.new(WordPolyRing.new(QQ(),"x,y"),"",
     [( y + x + 1/10 ), ( x*x + 1/10 * x + 1/10 )])
```

integro-differential Weyl algebra :

$$\mathbf{K}\langle \ell, \partial \rangle_{/(\partial\ell=1)}\{x; Q\}, \quad Q = \{x * \partial = \partial x - 1,\ x * \ell = \ell x + \ell^2\}$$

Conclusions

- presented parametric solvable polynomial rings, with definition of commutator relations between polynomial variables and coefficient variables
- enables the computation in recursive solvable polynomial rings
- possible to construct and compute in localizations with respect to two-sided ideals in such rings
- using these as coefficient rings of solvable polynomial rings makes computations of roots, common divisors and ideal constructions over skew fields feasible

Conclusions (cont.)

- algorithms implemented in JAS in a type-safe, object oriented way with generic coefficients
- the high complexity of the solvable multiplication and the lack of efficient simplifiers to reduce (intermediate) expression swell hinder practical computations
- this will eventually be improved in future work

Thank you for your attention

Questions ?

Comments ?

http://krum.rz.uni-mannheim.de/jas/

Acknowledgments

- thanks to: Thomas Becker, Raphael Jolly, Wolfgang K. Seiler, Axel Kramer, Thomas Sturm, Victor Levandovskyy, Joachim Apel, Hans-Günther Kruse, Markus Aleksy
- thanks to the referees