Comprehensive Gröbner Bases

in a

Java Computer Algebra System

Heinz Kredel
University of Mannheim

ASCM 2009, Fukuoka

ASCM
MACIS
2009

Overview

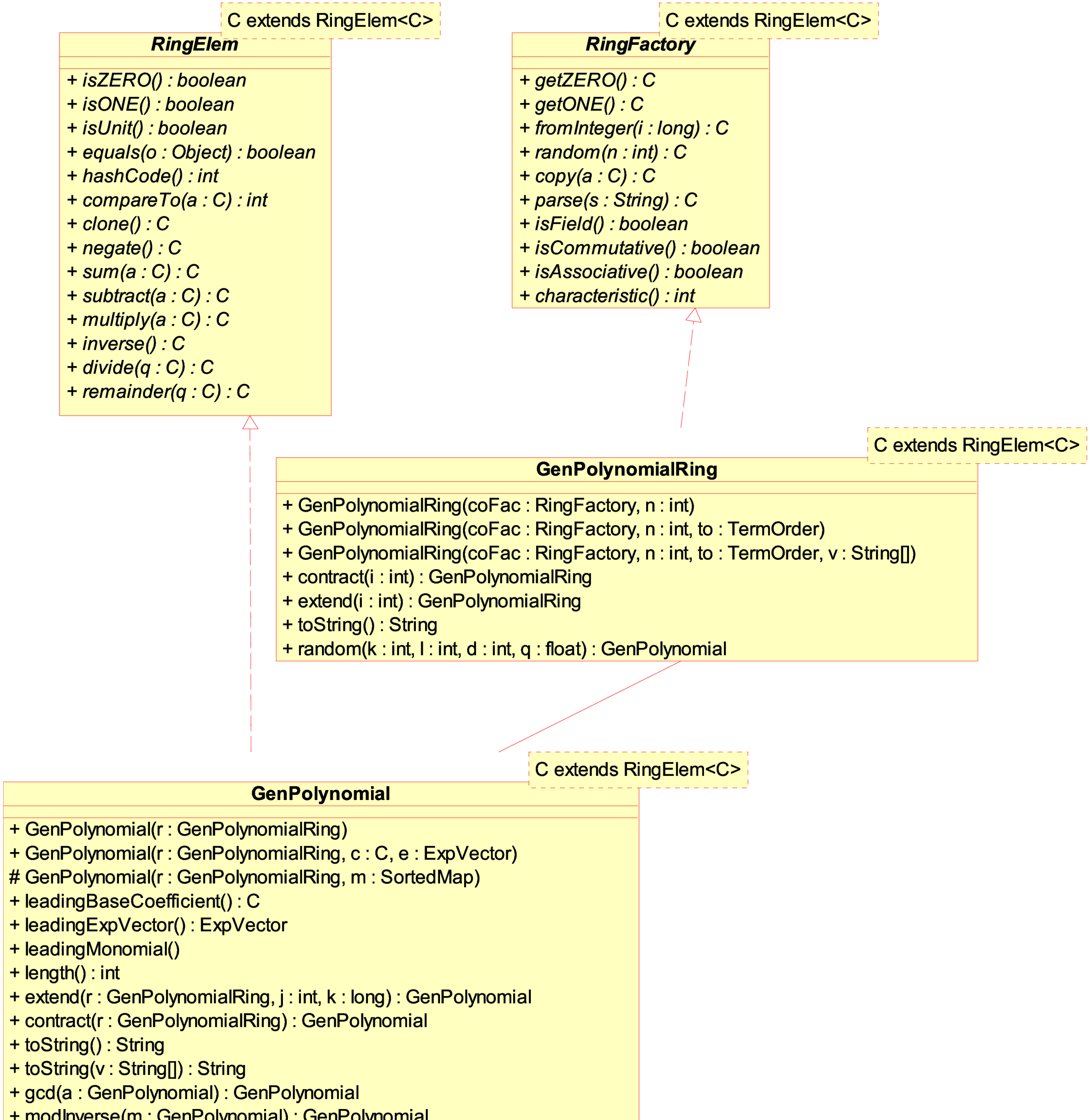
- Introduction to JAS
  - example with regular ring coefficients
- Comprehensive Gröbner Bases (CGB)
  - class layout
  - colored polynomials and conditions
  - parametric reductions and colored systems
  - Gröbner systems and CGB
- Examples
- Conclusions

Java Algebra System (JAS)

- object oriented design of a computer algebra system
  - = software collection for symbolic (non-numeric) computations
- type safe through Java generic types
- thread safe, ready for multi-core CPUs
- use dynamic memory system with GC
- jython (Java Python) interactive scripting front end

Implementation overview

- 230+ classes and interfaces
- plus 100+ JUnit test cases
- uses JDK 1.6 with generic types
  - Javadoc API documentation
  - logging with Apache Log4j
  - build tool is Apache Ant
  - revision control with Subversion
- jython (Java Python) scripts
  - support for Sage like polynomial expressions
- open source, license is GPL or LGPL



ASCM MACIS 2009

Polynomials over regular rings

example:

```
List<GenPolynomial<Product<Residue<BigRational>>>>
```

$$R=\mathbb{Q}[x_1,\ldots,x_n]$$

$$S'=(\prod_{\wp\in spec(R)} R/\wp)[y_1,\ldots,y_r]$$ a von Neuman regular ring

$$L\subset S=(\underbrace{\mathbb{Q}[x_0,x_1,x_2]/ideal(F)})^4[a,b]$$

```
rr = ResidueRing[ BigRational( x0, x1, x2 ) IGRLEX
     ( ( x0^2 + 295/336  ),
     ( x2 - 350/1593 x1 - 1100/2301 ) ) ]
L = [
     {0=x1 - 280/93 , 2=x0 * x1 - 33/23 } a^2 * b^3
   + {0=122500/2537649 x1^3 + 770000/3665493 x1^2
       + 14460385/47651409 x1 + 14630/89739 ,
      3=350/1593 x1 + 23/6 x0 + 1100/2301 } ,
    ... ]
```

ASCM MACIS 2009

Regular ring construction

```
 1 List<GenPolynomial<Product<Residue<BigRational>>>> L
    = new ArrayList<GenPolynomial<Product<Residue<BigRational>>>>();

 2 BigRational bf = new BigRational(1);
 3 GenPolynomialRing<BigRational> pfac
    = new GenPolynomialRing<BigRational>(bf,3);
 4 List<GenPolynomial<BigRational>> F
    = new ArrayList<GenPolynomial<BigRational>>();
 5 GenPolynomial<BigRational> pp = null;
 6 for ( int i = 0; i < 2; i++) {
 7     pp = pfac.random(5,4,3,0.4f);
 8     F.add(pp);
 9 }
10 Ideal<BigRational> id = new Ideal<BigRational>(pfac,F);
11 id.doGB();
12 ResidueRing<BigRational> rr = new ResidueRing<BigRational>(id);
13 System.out.println("rr = " + rr);
14 ProductRing<Residue<BigRational>> pr
    = new ProductRing<Residue<BigRational>>(rr,4);
```

Polynomial construction and GB

```
List<GenPolynomial<Product<Residue<BigRational>>>> L = ...

String[] vars = new String[] { "a", "b" };
GenPolynomialRing<Product<Residue<BigRational>>> fac
  = new GenPolynomialRing<Product<Residue<BigRational>>>(pr,2,vars)
GenPolynomial<Product<Residue<BigRational>>> p;
for ( int i = 0; i < 3; i++) {
    p = fac.random(2,4,4,0.4f);
    L.add(p);
}
System.out.println("L = " + L);

GroebnerBase<Product<Residue<BigRational>>> bb
  = new RGroebnerBasePseudoSeq<Product<Residue<BigRational>>>(pr);

List<GenPolynomial<Product<Residue<BigRational>>>> G = bb.GB(L);
System.out.println("G = " + G);
```

compute Gröbner base

Overview

- Introduction to JAS
  - example with regular ring coefficients
- Comprehensive Gröbner Bases (CGB)
  - class layout
  - colored polynomials and conditions
  - parametric reductions and colored systems
  - Gröbner systems and CGB
- Examples
- Conclusions

ASCM
MACIS
2009

CGB definitions

- parametric polynomial ring

$$R = K[U_1, \dots, U_m] = K[\boldsymbol{U}]$$
$$S = R[X_1, \dots, X_n] = R[\boldsymbol{X}]$$

- specialization

$$sigma : R \rightarrow K', S \rightarrow K'[X_1, \dots, X_n]$$

- comprehensive GB

$$F \subset S,\ ideal(F), G \subset S, given \leqslant$$

$$sigma(G)\ is\ GB\ \ for\ \ ideal(sigma(F))$$

- Gröbner System

$$GS = \{(gamma, G_{gamma}) \mid G_{gamma} \subset S\}$$

- Condition

$$gamma = \{z_i(\boldsymbol{U}) = 0\} \cup \{n_j(\boldsymbol{U}) \neq 0\}$$

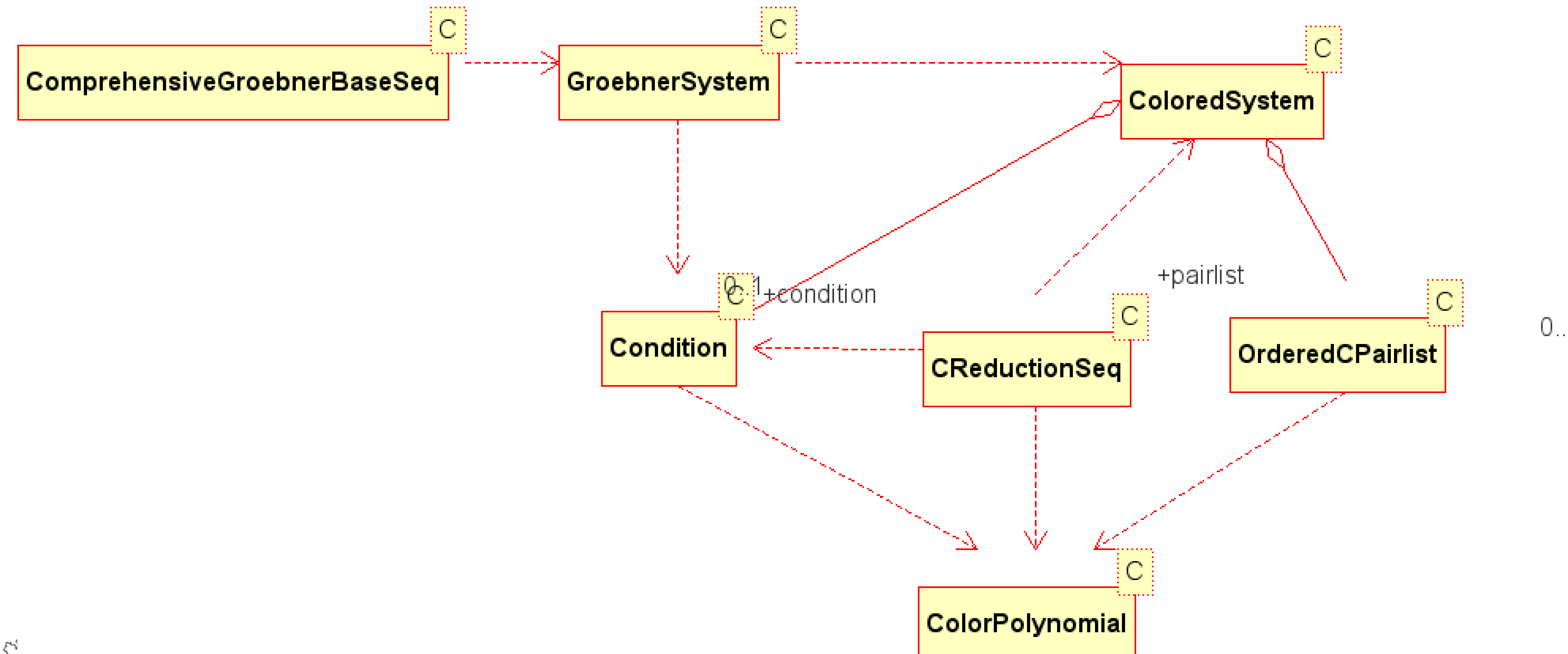
- colorings

$$col(a) = green,\ if\ a \in \{z_i(\boldsymbol{U}) = 0\}$$
$$col(a) = red,\ if\ a \in \{n_j(\boldsymbol{U}) \neq 0\}$$
$$col(a) = white, else$$

- determined polynomials

$$p \in S,\quad p = p_{green} + p_{red} + p_{white}$$
$$with\quad p_{green} > p_{red} > p_{white}$$

Classes overview

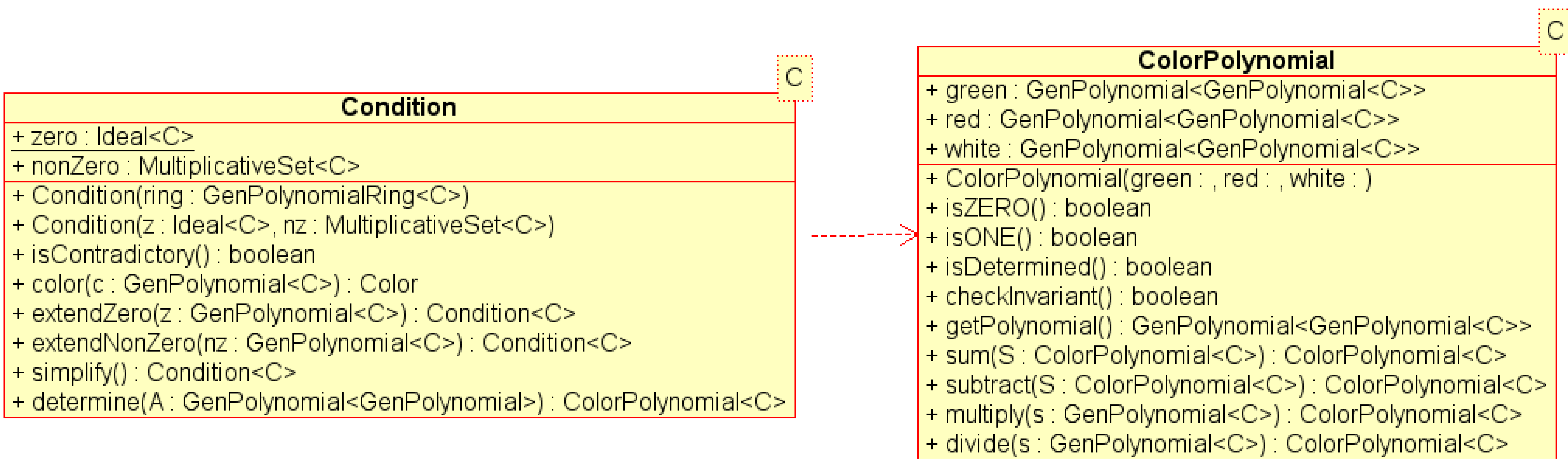


Classes overview (cont.)

- Gröbner systems as lists of colored systems
- a colored system consists of a condition, a list of colored polynomials and a critical pair list
- colored polynomial is a tuple (green, red, white) of polynomials with coefficients colored with respect to a condition
- parametric reduction relative to a condition
- implementation for parametric polynomials `GenPolynomial<GenPolynomial<C>>`
  - classes have type parameters `C extends RingElem<C>`

ASCM MACIS 2009

Condition and colored polynomial



Condition

- condition $gamma = \{z_i(\boldsymbol{U}) = 0\} \cup \{n_j(\boldsymbol{U}) \neq 0\}$
- two finite sets of
  - polynomial equations $z(\boldsymbol{U}) = 0$
  - polynomial inequalities $n(\boldsymbol{U}) \neq 0$
- method `color(c)` returns green, red or white if `c` is contained in the respective set
- method `determine(A)` returns a colored polynomial
- methods to extend the condition
  - `extendZero(z): Condition`
  - `extendNonZero(n): Condition`

Colored Polynomial

- consists of a green, red and white part
- with green > red > white for non-zero parts with respect to the given term order >
- test if the restriction holds: `checkInvariant()`
- test if the red part is non-zero or the white part is also zero: `isDetermined()`
  - arithmetic as far as is required by parametric reduction
  - tests if the polynomial is zero or one by ignoring the green part
  - methods to extract parametric polynomials

Condition implementation (1)

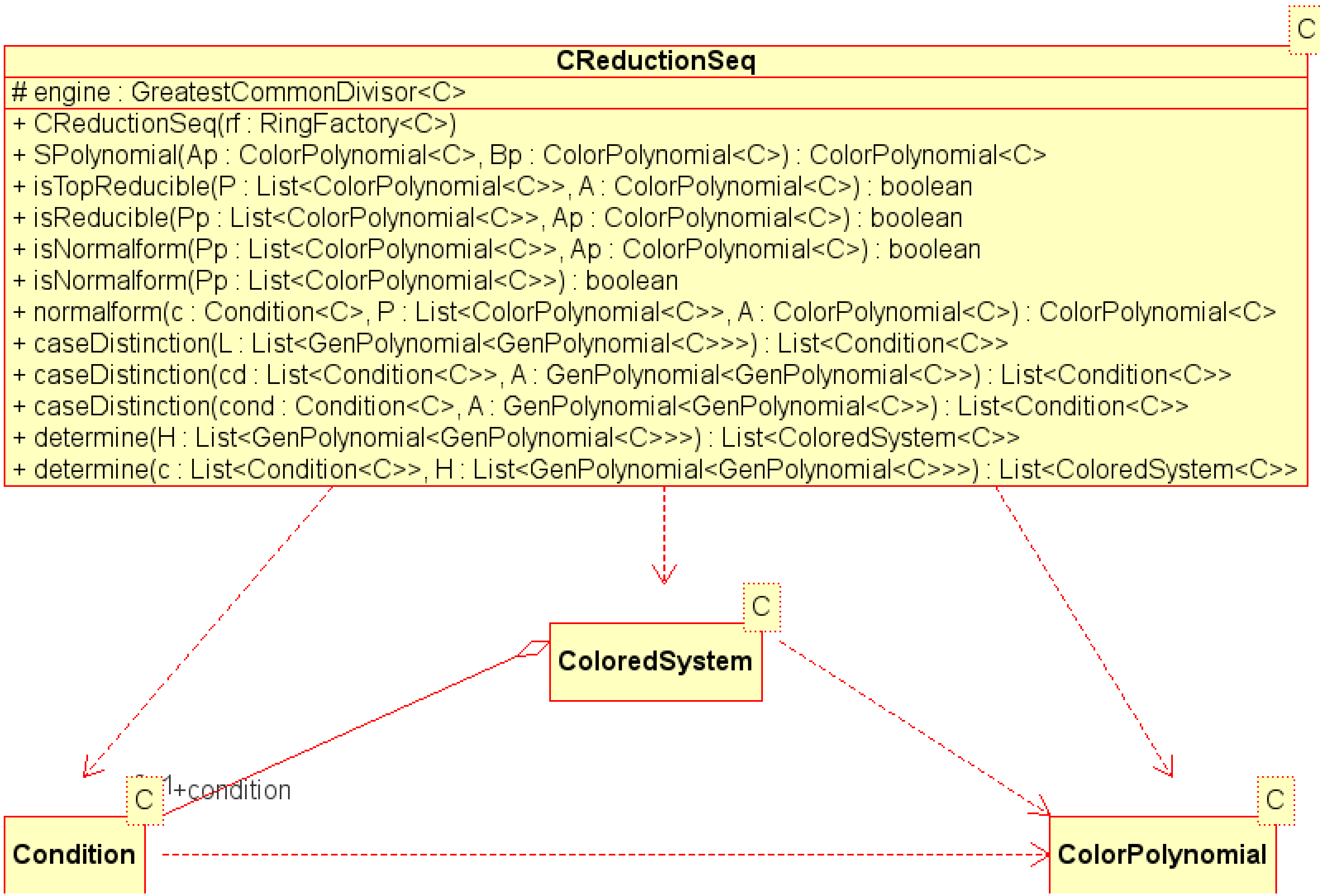
- replace 'zero set' by 'ideal' to have more efficient containment test
  - set containment then ideal membership test
  - by lazy Gröbner base computation
  - moreover square-free polynomials give radical membership test
- replace 'non-zero set' by 'multiplicative set'
  - set containment then product of factors test
  - elements are kept co-prime, or square-fee and co-prime, or irreducible
  - default is square-free and co-prime

ASCM
MACIS
2009

Condition implementation (2)

- the extension methods try to add only 'small' polynomials to the respective set
  - take residues with respect to the zero ideal
  - remove factors from multiplicative set
- recursively simplify the resulting condition `simplify()`
  - make zero ideal polynomials square-free
  - reduce multiplicative set modulo zero ideal
  - take co-prime (etc) factors for multiplicative set
  - remove factors from zero set which are contained in non-zero set
  - do recursion if simplifications where possible

Parametric reduction



Reduction implementation

- works on colored polynomials
- parametric reduction ignores green terms
- but green terms are updated during computation: gives **faith-full** Gröbner system
- in normal-form green terms are copied to the result polynomial
- red or white terms are reduced with respect to a suitable (colored) polynomial in the reduction list
  - top-reduction stops if a non-reducible term is encountered
  - colored S-polynomials computed as usual

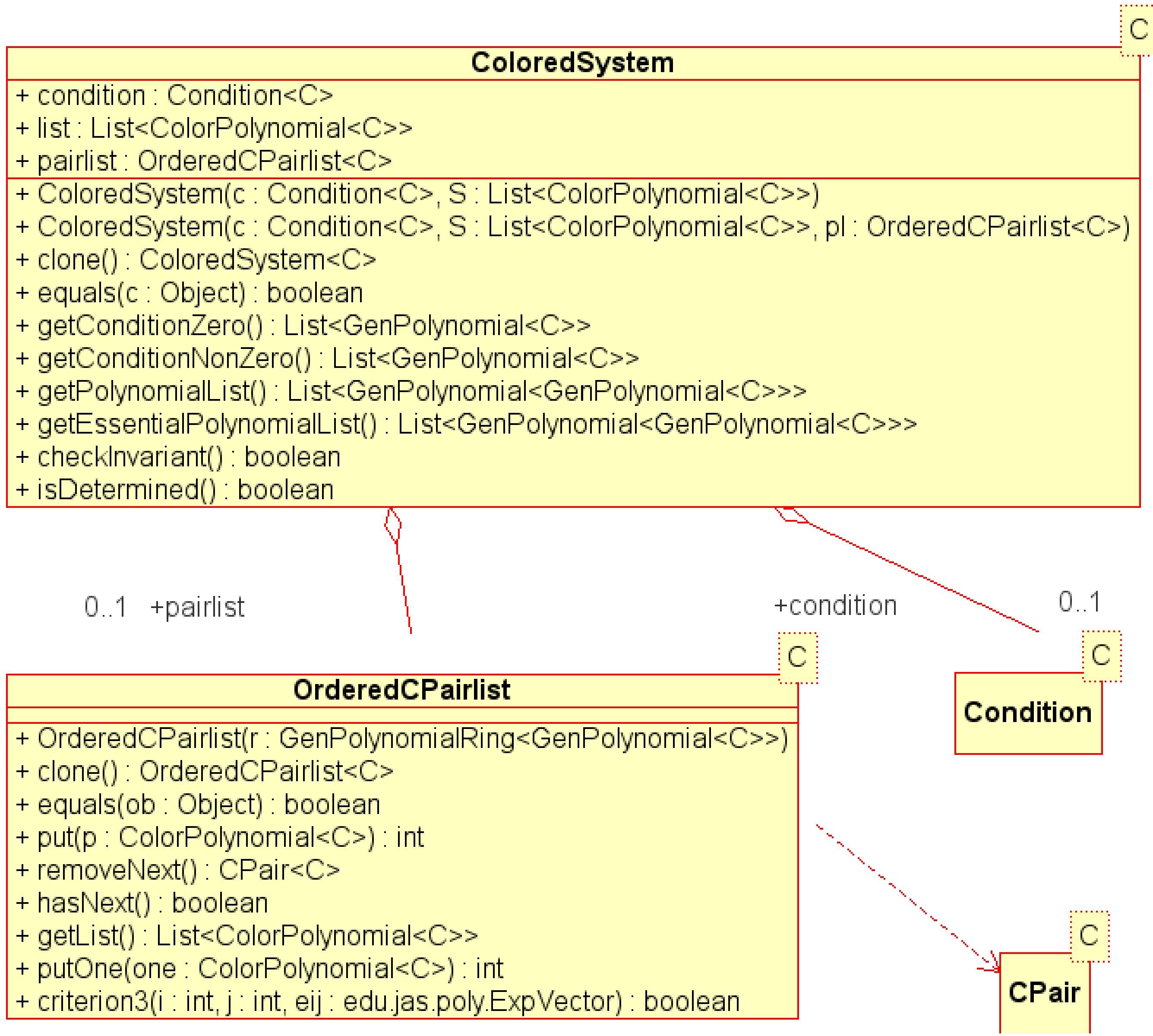
Construction of conditions (1)

- method `determine(L)` constructs a list of colored systems for a list of polynomials `L`, by
  - computing a list of conditions with method `caseDistinction(L)`
  - for each condition a colored system is computed with `determine(C,L)`
- a colored system consists of a condition together with a list of colored polynomials wrt. this condition
- the case distinction is constructed such that each colored polynomial has non-zero red term (or the white term is zero)

Construction of conditions (2)

- the algorithm checks the color each coefficient of each polynomial (in term order sequence) with respect to each existing condition
  - green coefficients are skipped
  - if a red coefficient appears, the polynomial is done
  - for a white coefficient the current condition is extended by adding the coefficient to the set of zero conditions and to the set of non-zero conditions
- initially the list of conditions is made from one empty condition

Colored system

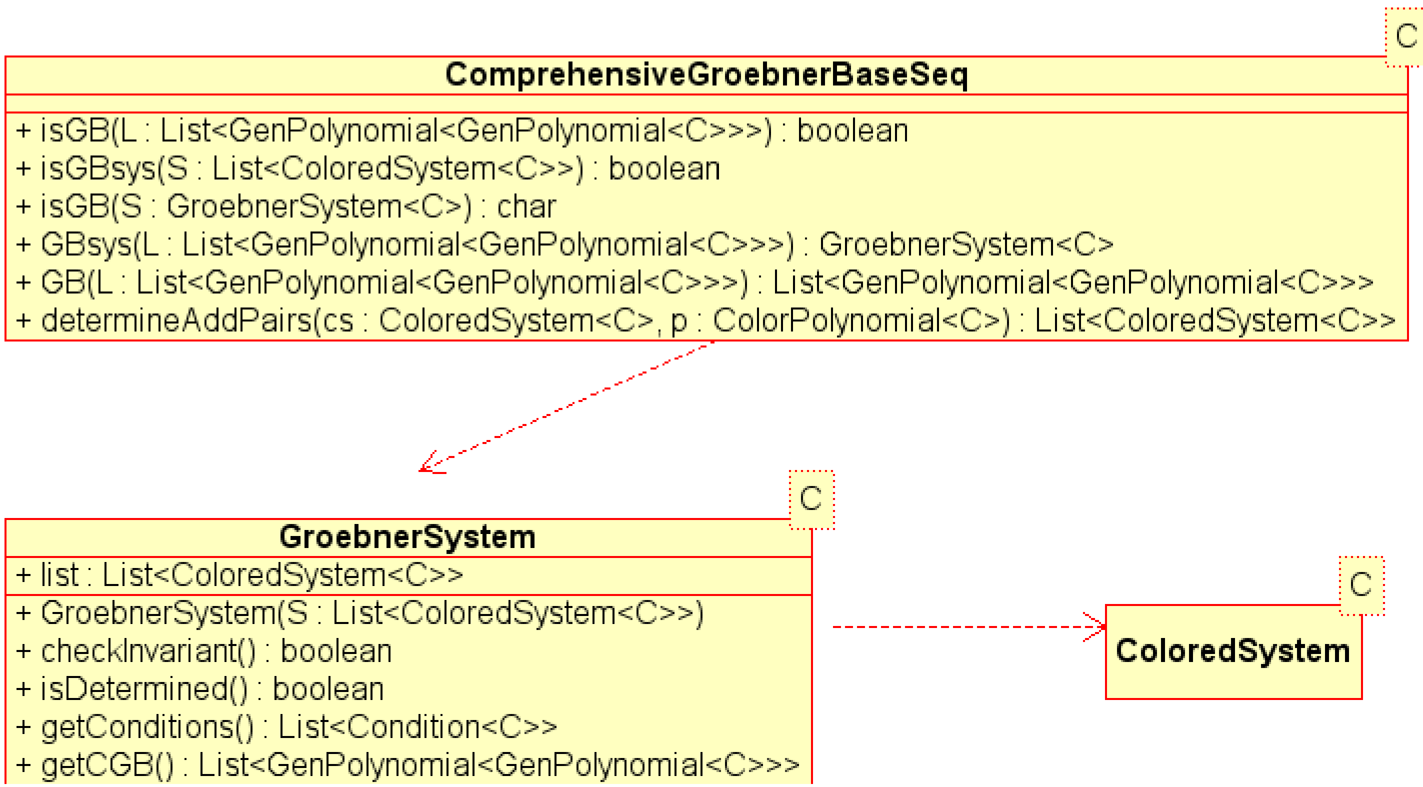


ASCM MACIS 2009

Gröbner systems

- method `GB()` of `ComprehensiveGroebnerBaseSeq` computes a Gröbner system with `GBsys()`, then extracts a comprehensive Gröbner base
- `GroebnerSystem` is a container for a list of colored systems
- method `getCGB()` extracts comprehensive Gröbner base as union of parametric polynomials from all contained colored polynomials
- has methods to check invariants or if each system is determined

Gröbner system and CGB



ASCM MACIS 2009

CGB construction (1)

- the list of `ColoredSystem`s is initially constructed
- then each `ColoredSystem` is augmented by a critical pair list as in the standard Buchberger algorithm
- for each critical pair a parametric S-polynomial is constructed and reduced with respect to the list of colored polynomials
- if all reductions lead to the zero polynomial a colored system is done
- for a non-zero reduction polynomial the condition is eventually refined
- for each condition the list of colored systems is updated

CGB construction (2)

- branching and critical pair generation is done in method `determineAddPairs()`
- new generated colored systems are merged with the existing list of colored systems in method `addToList()` with test of equal conditions and lists of polynomials
- upon termination each colored polynomial list in each colored system is a Gröbner base for this condition
  - termination is guaranteed by König's tree lemma together with Dickson's lemma

CGB tests (1)

- there are two tests to check if a given list of parametric polynomials is a CGB
- one test determines the polynomials and constructs a Gröbner system
  - for each colored system all critical pairs are constructed and the S-polynomials are parametrically reduced
  - if all these reductions lead to the zero polynomial (ignoring green parts), it is concluded that it is a Gröbner system

CGB tests (2)

- the other test also determines the polynomials and constructs the list of colored systems
  - for each condition a residue class ring modulo the zero ideal is constructed
  - the given polynomials are mapped to these residue class coefficient rings
  - over these rings a standard `isGB()` test is performed
  - additionally a test with random ideal is done
  - if all these tests succeed, it is concluded that the given list of polynomials is a CGB

Overview

- Introduction to JAS
  - example with regular ring coefficients
- Comprehensive Gröbner Bases (CGB)
  - class layout
  - colored polynomials and conditions
  - parametric reductions and colored systems
  - Gröbner systems and CGB
- Examples
- Conclusions

ASCM
MACIS
2009

Raksanyi and Hawes examples

example	MAS time	conditions	JAS time	conditions
Raksanyi, S, Gr	40	3	520 / 229 / 190	5
Raksanyi, Lr	not impl.	–	344 / 134 / 94	3
Raksanyi, L	5630	22	511 / 225 / 175	4
Raksanyi, G	30	3	337 / 147 / 99	3
Hawes2, G	> 20 min	–	1119 / 603 / 578	5

time in milliseconds, timings in slashes are for subsequent runs,
Term order: G = graded, L = lexicographical, S = Gr = reverse graded,
Lr = reverse lexicographical

Nabeshima examples

example	from [16]	cond	JAS, AMD, L	cond	JAS, AMD, G	cond
F_1	31	4	285 / 151 / 97	7	270 / 142 / 99	7
F_2	93	6	2299 / 1765 / 1664	12	509 / 281 / 165	10
F_3	2203	22	1186 / 720 / 660	29	1199 / 967 / 681	29
F_4	234	15	1231 / 722 / 674	34	1365 / 845 / 751	34
F_5	109	6	359 / 184 / 126	11	367 / 187 / 125	8
F_6	359	17	95 / 43 / 34	4	90 / 42 / 34	4
F_7	375	7	392 / 194 / 117	6	424 / 242 / 128	6
F_8	133200	458	2548 / 1856 / 1788	32	4883 / 4043 / 3664	32

time in milliseconds, timings in slashes are for subsequent runs,
Term order: G = graded, L = lexicographical,
cond = number of conditions.

Montes examples

example	JAS time	conditions	DISPGB time	conditions
11.1, L	777 / 308 / 327	23	8800	6
11.2, L	490 / 246 / 143	10	5200	6
11.3, L	1013 / 600 / 516	9	115900	7
11.4, L	371939 / 359274 / 355794	7	33000	7
5.1 simpl., L	248 / 95 / 86	3	8400	4

time in milliseconds, timings in slashes are for subsequent runs.
DISPGB times from [14],
Term order: G = graded, L = lexicographical.

Regular ring example (jython)

```
r = PolyRing(PolyRing(QQ(),"a1,a2,a3,a4",PolyRing.grad),
             "x1,x2,x3,x4",PolyRing.lex);
[one,a1,a2,a3,a4,x1,x2,x3,x4] = r.gens();

pl = [ ( x4 - ( a4 - a2 ) ),
      ( x1 + x2 + x3 + x4 - ( a1 + a3 + a4 ) ),
      ( x1 * x3 + x1 * x4 + x2 * x3 + x3 * x4 - ( a1 * a4 + a1 *
a3 + a3 * a4 ) ),
      ( x1 * x3 * x4 - ( a1 * a3 * a4 ) )
     ];
f = ParamIdeal(r,list=pl);
gs = f.CGBsystem();
bg = gs.isCGBsystem(); # → true

rs.regularRepresentationBC();
print "boolean closed regular representation: "+str(rs);
bg = rs.isRegularGB(); # → true
```

Conclusions

- design and implementation of (faith-full) comprehensive Gröbner bases in Java
- generic object oriented design provides all required mathematical objects and structures
- conditions implemented as
  - “zero eq set” as ideal with membership test
  - “non zero set” as multiplicative set
- computing times in same magnitude as others
- use residue class coefficient rings

Future work

- when multivariate polynomial factorization becomes ready, use it for multiplicative sets in conditions
- comprehensive Gröbner bases for solvable polynomial rings
- parallel versions of comprehensive Gröbner base computation

Thank you

- Questions or Comments?
- http://krum.rz.uni-mannheim.de/jas
- `git http://krum.rz.uni-mannheim.de/jas.git`
- Thanks to
  - Raphael Jolly, Thomas Becker
  - Markus Aleksy, Hans-Günther Kruse
  - W. K. Seiler, Dongming Wang, Th. Sturm
  - the referees
  - and other colleagues