16.7 BIBASIS: A Package for Calculating Boolean Involutive Bases

Authors: Yuri A. Blinkov and Mikhail V. Zinin

16.7.1 Introduction

Involutive polynomial bases are redundant Gröbner bases of special structure with some additional useful features in comparison with reduced Gröbner bases [1]. Apart from numerous applications of involutive bases [2] the involutive algorithms [3] provide an efficient method for computing reduced Gröbner bases. A reduced Gröbner basis is a well-determined subset of an involutive basis and can be easily extracted from the latter without any extra reductions. All this takes place not only in rings of commutative polynomials but also in Boolean rings.

Boolean Gröbner basis already have already revealed their value and usability in practice. The first impressive demonstration of practicability of Boolean Gröbner bases was breaking the first HFE (Hidden Fields Equations) challenge in the public key cryptography done in [4] by computing a Boolean Gröbner basis for the system of quadratic polynomials in 80 variables. Since that time the Boolean Gröbner bases application area has widen drastically and nowadays there is also a number of quite successful examples of using Gröbner bases for solving SAT problems.

During our research we had developed [5, 6, 7] Boolean involutive algorithms based on Janet and Pommaret divisions and applied them to computation of Boolean Gröbner bases. Our implementation of both divisions has experimentally demonstrated computational superiority of the Pommaret division implementation. This package BIBASIS is the result of our thorough research in the field of Boolean Gröbner bases. BIBASIS implements the involutive algorithm based on Pommaret division in a multivariate Boolean ring.

In section 2 the Boolean ring and its peculiarities are shortly introduced. In section 3 we briefly argue why the involutive algorithm and Pommaret division are good for Boolean ring while the Buhberger's algorithm is not. And finally in section 4 we give the full description of BIBASIS package capabilities and illustrate it by examples.

16.7.2 Boolean Ring

Boolean ring perfectly goes with its name, it is a ring of *Boolean functions* of n variables, i.e mappings from $\{0,1\}^n$ to $\{0,1\}^n$. Considering these variables are $\mathbf{X} := \{x_1, \ldots, x_n\}$ and $\mathbb{F}_2$ is the finite field of two elements $\{0,1\}$, Boolean ring

can be regarded as the quotient ring

$$\mathbb{B}\left[\mathbf{X}\right] := \mathbb{F}_2[\mathbf{X}] \,/ < x_1^2 + x_1, \ldots, x_n^2 + x_n > .$$

Multiplication in $\mathbb{B}\left[\mathbf{X}\right]$ is *idempotent* and addition is *nilpotent*

$$\forall\, b \in \mathbb{B}\left[\mathbf{X}\right] \;:\; b^2 = b\,,\; b + b = 0.$$

Elements in $\mathbb{B}\left[\mathbf{X}\right]$ are *Boolean polynomials* and can be represented as finite sums

$$\sum_{j} \prod_{x \in\, \Omega_j \subseteq \mathbf{X}} x$$

of *Boolean monomials*. Each monomial is a conjunction. If set Ω is empty, then the corresponding monomial is the unity Boolean function 1. The sum of zero monomials corresponds to zero polynomial, i.e. is zero Boolean function 0.

16.7.3 Pommaret Involutive Algorithm

Detailed description of involutive algorithm can found in [3]. Here we note that result of both involutive and Buhberger's algorithms depend on chosen monomial ordering. At that the ordering must be admissible, i.e.

$$m \neq 1 \Longleftrightarrow m \succ 1, \qquad m_1 \succ m_2 \Longleftrightarrow m_1 m \succ m_2 m \qquad \forall\, m, m_1, m_2.$$

But as one can easily check the second condition of admissibility does not hold for any monomial ordering in Boolean ring:

$$x_1 \succ x_2 \quad \xrightarrow{*x_1} \quad x_1 * x_1 \succ x_2 * x_2 \quad \longrightarrow \quad x_1 \prec x_1 x_2$$

Though $\mathbb{B}\left[\mathbf{X}\right]$ is a principal ideal ring, boolean singleton $\{p\}$ is not necessarily a Gröbner basis of ideal $< p >$, for example:

$$x_1, x_2 \in < x_1 x_2 + x_1 + x_2 > \subset \mathbb{B}\left[x_1, x_2\right].$$

That the reason why one cannot apply the Buhberger's algorithm directly in a Boolean ring, using instead a ring $\mathbb{F}_2[\mathbf{X}]$ and *the field binomials* $x_1^2 + x_1, \ldots, x_n^2 + x_n$.

The involutive algorithm based on Janet division has the same disadvantage unlike the Pommaret division algorithm as shown in [5]. The Pommaret division algorithm can be applied directly in a Boolean ring and admits effective data structures for monomial representation.

16.7.4 BIBASIS Package

The package BIBASIS implements the Pommaret division algorithm in a Boolean ring. The first step to using the package is to load it:

```
1: load_package bibasis;
```

The current version of the BIBASIS user interface consists only of 2 functions: `bibasis` and `bibasis_print_statistics`.

The `bibasis` is the function that performs all the computation and has the following syntax:

```
bibasis(initial_polynomial_list, variables_list,
    monomial_ordering, reduce_to_groebner);
```

Input:

- `initial_polynomial_list` is the list of polynomials containing the known basis of initial Boolean ideal. All given polynomials are treated modulo 2. See Example 1.
- `variables_list` is the list of independent variables in decreasing order.
- `monomial_ordering` is a chosen monomial ordering and the supported ones are:

  `lex` – pure lexicographical ordering;

  `deglex` – degree lexicographic ordering;

  `degrevlex` – degree reverse lexicographic.

  See Examples 2—4 to check that Gröbner (as well as involutive) basis depends on monomial ordering.
- `reduce_to_groebner` is a Boolean value, if it is `t` the output is the reduced Boolean Gröbner basis, if `nil`, then the reduced Boolean Pommaret basis. Examples 5,6 show distinctions between these two outputs.

Output:

- The list of polynomials which constitute the reduced Boolean Gröbner or Pommaret basis.

The syntax of `bibasis_print_statistics` is simple:

```
bibasis_print_statistics();
```

This function prints out a brief statistics for the last invocation of `bibasis` function. See Example 7.

16.7.5 Examples

Example 1:

```
1: load_package bibasis;
2: bibasis({x+2*y}, {x,y}, lex, t);
{x}
```

Example 2:

```
1: load_package bibasis;
2: variables :={x0,x1,x2,x3,x4}$
3: polynomials := {x0*x3+x1*x2,x2*x4+x0}$
4: bibasis(polynomials, variables, lex, t);
{x0 + x2*x4,x2*(x1 + x3*x4)}
```

Example 3:

```
1: load_package bibasis;
2: variables :={x0,x1,x2,x3,x4}$
3: polynomials := {x0*x3+x1*x2,x2*x4+x0}$
4: bibasis(polynomials, variables, deglex, t);
{x1*x2*(x3 + 1),
 x1*(x0 + x2),
 x0*(x2 + 1),
 x0*x3 + x1*x2,
 x0*(x4 + 1),
 x2*x4 + x0}
```

Example 4:

```
1: load_package bibasis;
2: variables :={x0,x1,x2,x3,x4}$
3: polynomials := {x0*x3+x1*x2,x2*x4+x0}$
4: bibasis(polynomials, variables, degrevlex, t);
{x0*(x1 + x3),
 x0*(x2 + 1),
```

```
x1*x2 + x0*x3,
x0*(x4 + 1),
x2*x4 + x0}
```

Example 5:

```
1: load_package bibasis;
2: variables :={x,y,z}$
3: polinomials := {x, z}$
4: bibasis(polinomials, variables, degrevlex, t);
{x,z}
```

Example 6:

```
1: load_package bibasis;
2: variables :={x,y,z}$
3: polinomials := {x, z}$
4: bibasis(polinomials, variables, degrevlex, nil);
{x,z,y*z}
```

Example 7:

```
1: load_package bibasis;
2: variables :={u0,u1,u2,u3,u4,u5,u6,u7,u8,u9}$
3: polinomials := {u0*u1+u1*u2+u1+u2*u3+u3*u4+u4*u5+u5*u6+u6*u7+u7*u8
3:                 u0*u2+u1+u1*u3+u2*u4+u2+u3*u5+u4*u6+u5*u7+u6*u8+u7
3:                 u0*u3+u1*u2+u1*u4+u2*u5+u3*u6+u3+u4*u7+u5*u8+u6*u9,
3:                 u0*u4+u1*u3+u1*u5+u2+u2*u6+u3*u7+u4*u8+u4+u5*u9,
3:                 u0*u5+u1*u4+u1*u6+u2*u3+u2*u7+u3*u8+u4*u9+u5,
3:                 u0*u6+u1*u5+u1*u7+u2*u4+u2*u8+u3+u3*u9+u6,
3:                 u0*u7+u1*u6+u1*u8+u2*u5+u2*u9+u3*u4+u7,
3:                 u0*u8+u1*u7+u1*u9+u2*u6+u3*u5+u4+u8,
3:                 u0+u1+u2+u3+u4+u5+u6+u7+u8+u9+1}$
4: bibasis(polinomials, variables, degrevlex, t);
{u3*u6,
 u3*u7,
 u7*(u6 + 1),
 u3*u8,
 u6*u8 + u6 + u7,
 u7*u8,
 u3*(u9 + 1),
 u6*u9 + u7,
 u7*(u9 + 1),
 u8*u9 + u6 + u7 + u8,
 u0 + u3 + u6 + u9 + 1,
 u1 + u7,
```

```
 u2 + u7 + u8,
 u4 + u6 + u8,
 u5 + u6 + u7 + u8}
5: bibasis_print_statistics();
        Variables order = u0 > u1 > u2 > u3 > u4 > u5 > u6 > u7 > u8 > u9
Normal forms calculated = 216
  Non-zero normal forms = 85
        Reductions made = 4488
Time: 270 ms
GC time: 0 ms
```

Bibliography

[1] V.P.Gerdt and Yu.A.Blinkov. *Involutive Bases of Polynomial Ideals*. Mathematics and Computers in Simulation, 45, 519–542, 1998; *Minimal Involutive Bases*, ibid. 543–560.

[2] W.M.Seiler. *Involution: The Formal Theory of Differential Equations and its Applications in Computer Algebra*. Algorithms and Computation in Mathematics, 24, Springer, 2010. arXiv:math.AC/0501111

[3] Vladimir P. Gerdt. *Involutive Algorithms for Computing Gröbner Bases*. Computational Commutative and Non-Commutative Algebraic Geometry. IOS Press, Amsterdam, 2005, pp.199–225.

[4] J.-C.Faugère and A.Joux. Algebraic Cryptanalysis of Hidden Field Equations (HFE) Using Gröbner Bases. *LNCS* 2729, Springer-Verlag, 2003, pp.44–60.

[5] V.P.Gerdt and M.V.Zinin. A Pommaret Division Algorithm for Computing Gröbner Bases in Boolean Rings. *Proceedings of ISSAC 2008*, ACM Press, 2008, pp.95–102.

[6] V.P.Gerdt and M.V.Zinin. Involutive Method for Computing Gröbner Bases over F_2. *Programming and Computer Software*, Vol.34, No. 4, 2008, 191–203.

[7] Vladimir Gerdt, Mikhail Zinin and Yuri Blinkov. On computation of Boolean involutive bases, Proceedings of International Conference Polynomial Computer Algebra 2009, pp. 17-24 (International Euler Institute, April 7-12, 2009, St. Peterburg, Russia)