学位申請論文

On the Efficient Implementation of Boolean Gröbner Bases (ブーリアン・グレブナー基底の効率的な実装について)

平成 29 年 3 月

Akira Nagai
(永井 彰)

Department of Mathematical Science for Information Sciences,
Tokyo University of Science
1-3 Kagurazaka, Shinjuku-ku,
Tokyo, Japan

Contents

Acknowledgments

I would like to thank Professor Yosuke Sato for his valuable comments and fruitful discussion during the course of this study. I also thank all members of Sato Laboratory for their helpful comment and support. Finally, I thank my wife, son, daughter and all of my friends.

Chapter 1

Introduction

For solving polynomial equations, Gröbner bases computation is a powerful tool. Though Gröbner bases were originally introduced by B.Buchberger in polynomial rings over fields([4]), there also have been done many works concerning Gröbner bases of polynomial rings with coefficient rings that are not fields. Among them Gröbner bases of Boolean polynomial rings (*Boolean Gröbner bases*) introduced in [19, 20] have a nice property.

A residue class ring $\mathbf{B}[X_1, \ldots, X_n]/\langle X_1^2 + X_1, \ldots, X_n^2 + X_n\rangle$ over a Boolean ring $\mathbf{B}$ is called a *Boolean polynomial ring*. A Gröbner basis in a Boolean polynomial ring is called a *Boolean Gröbner basis*, which is first introduced in [20] together with its computation algorithm. An ideal in a polynomial ring over the Galois field $\mathbb{GF}_2$ is the simplest Boolean ring. Since $\mathbb{GF}_2$ is actually a field, such a Boolean Gröbner basis is easily computed, with no novel theoretical advances. When the Boolean ring $\mathbf{B}$ is $\mathcal{P}_{FC}(St)$ that consists of all finite or co-finite subsets of St (Here, St is a set of all strings of the computer language), Boolean Gröbner bases are of great importance for solving certain types of combinatorial problems. Since we need a special data structure to encode a Boolean ring $\mathcal{P}_{FC}(St)$, it is not straightforward to implement the computation of Boolean Gröbner bases in a general purpose computer algebra system. In fact the first implementation was done in the logic programing languages Prolog and Klic [21, 22]. When a Boolean ring $\mathbf{B}$ is the Galois field $\mathbb{GF}_2$ with characteristic 2, we can easily compute Boolean Gröbner bases in most computer algebra system with a facility to compute Gröbner bases in a polynomial ring over a finite field.

By the technique introduced in [14], we can now compute Boolean Gröbner bases for $\mathcal{P}_{FC}(St)$ by the computation of Boolean Gröbner bases of $\mathbb{GF}_2$. This method is implemented in the computer algebra system Risa/Asir [12]. It brings us a much faster program than those of [21, 22], which enables us to obtain the recent work of Sudoku puzzles [13]. Though the purpose of the application of Boolean Gröbner bases to Sudoku puzzles is not making a fast solver, the program can solve any Sudoku puzzle in acceptable length of time, while other existing Sudoku solvers by the computation of Gröbner bases such as [1, 6] can solve only limited types of puzzles. Nevertheless, we can not say it is a real time Sudoku solver since the program takes more than 10 seconds for solving most puzzles by a standard laptop computer.

In this paper we describe our implementation of Boolean Gröbner bases for $\mathcal{P}_{FC}(St)$ in the computer algebra system SageMath using the PolyBoRi library [3]. In chapter 5 our experimental data suggest that SageMath is faster and more optimal software to compute Boolean Gröbner bases than other computer algebra systems such as Risa/Asir, Singlar, Mathematica or Maple.

Since PolyBoRi has an optimal data structure for the computation of a Boolean polynomial ring over $\mathbb{GF}_2$, our program of Sudoku puzzles achieves about 15 times speed-up than the previous program in Risa/Asir. It enables us to have a first-ever real time solver of Sudoku puzzles by the computation of Gröbner bases.

In this paper we also introduce our parallel and distributed computation method of Boolean Gröbner bases, which is implemented in the computer algebra system SageMath using the PolyBoRi library [3]. We parallelize a Boolean Gröbner bases program introduced in [15]. Note that a parallel computation program of Boolean Gröbner bases for $\mathcal{P}_{FC}(St)$ is already implemented in [24, 22] using a parallel logic programming language KLIC. Unfortunately, their program does not achieve enough speed-up, our program achieves satisfactory speed-up.

We have also recomputed s-ranks of 735 Sudoku puzzles treated in [13] and found serious mistakes on the computation data reported in it. The s-rank of a Sudoku puzzle is a mathematical index which represents its level of difficulty that was introduced in [13]. In order to compute a s-rank of a Sudoku puzzle, we need to compute much more Boolean Gröbner bases. For some Sudoku puzzles, a sequential program needs computation time beyond several minutes. We need parallel and distributed computation of Boolean Gröbner bases, we have used a parallel computation facility of SageMath. In this paper we use the word "parallel computation" for parallel computation by one computer with a multi-core processor, "distributed computation" for distributed computation by several computers connected on the Internet. So, "parallel and distributed computation" means parallel computation using several connected computers. (See [16].)

We put our prototype program as an open software at the following URL:

http://www.mi.kagu.tus.ac.jp/~nagai/BoolGB_Sage/.

We also show that for a given ideal $I = \langle f_1(\bar{A}, \bar{X}), \ldots, f_l(\bar{A}, \bar{X})\rangle$ in a Boolean polynomial ring $\mathbf{B}(\bar{A}, \bar{X})$, we can construct the elimination ideal $I \cap \mathbf{B}(\bar{A})$ by computing a Boolean Gröbner basis of the ideal $\langle f_1(\bar{c}, \bar{X}), \ldots, f_l(\bar{c}, \bar{X})\rangle$ in the Boolean polynomial ring $B(\bar{X})$ for each $0, 1$ specialization $\bar{c}$ of $\bar{A}$. We also give some example of our computation experiments to show that our method is quite effective in case we do not have many parameters.

The paper is organized as follows. In chapter 2, we show two classical results of Boolean algebra. In chapter 3 and 4, we give a quick review of Boolean Gröbner bases and comprehensive Boolean Gröbner bases. Especially section 4.4 is devoted to our new algorithm for the computation of elimination ideals. In chapter 5, we describe our implementation method of Boolean Gröbner bases using the PolyBoRi library. we also describe parallel computation and distributed computation of Boolean Gröbner bases. In chapter 6, we discuss applications of Boolean Gröbner bases. The reader is referred to [27] for a comprehensive description of Boolean polynomial rings and Boolean Gröbner bases, also to [13] for more detailed description of the application of Boolean Gröbner bases to Sudoku puzzles.

Chapter 2

Boolean Polynomial Ring

In this chapter, we show two classical results of Boolean algebra in terms of Boolean polynomial rings. More details can be found in many text books of Boolean algebra such as [18] for example.

2.1 Boolean Polynomial Ring

Definition 1 A commutative ring $\mathbf{B}$ with an identity 1 is called a Boolean ring if every element a of $\mathbf{B}$ is idempotent, i.e. $a^2 = a$.

$\langle \mathbf{B}, \vee, \wedge, \neg \rangle$ becomes a Boolean algebra with the Boolean operations $\vee, \wedge, \neg$ defined by $a \vee b = a + b + a \cdot b$, $a \wedge b = a \cdot b$, $\neg\, a = 1 + a$. Conversely, for a Boolean algebra $\langle \mathbf{B}, \vee, \wedge, \neg \rangle$, if we define $+$ and $\cdot$ by $a + b = (\neg\, a \wedge b) \vee (a \wedge \neg\, b)$ and $a \cdot b = a \wedge b$, $\langle \mathbf{B}, +, \cdot \rangle$ becomes a Boolean ring. We use the symbol $\succeq$ to denote a partial order of a Boolean ring, that is $a \succeq b$ if and only if $ab = b$ for elements a, b of a Boolean ring $\mathbf{B}$. Since $-a = a$ in a Boolean ring, we do not need to use the symbol $'-'$, however, we also use $-$ when we want to stress its meaning.

Definition 2 A non-zero element e of a Boolean ring $\mathbf{B}$ is said to be atomic, if there does not exist a non-zero element c such that $ce = c$ except for $c = e$. (An atomic element is nothing but a non-zero minimal element w.r.t. $\succeq$.)

Lemma 3 If $\mathbf{B}$ is a finite Boolean ring, it has at least one atomic element. Let $e_1, \dots, e_k$ be all the atomic elements of $\mathbf{B}$, then $e_i e_j$ = 0 for any i $\neq$ j and $e_1 + \cdots + e_k = 1$.

proof We show the last equation, the rests are obvious. If $e_1 + \cdots + e_k \neq 1$, $e_1 + \cdots + e_k + 1 \neq 0$. Let c be a minimal element(an atomic element) of $\mathbf{B}$ such that $e_1 + \cdots + e_k \neq c$, i.e. $c(e_1 + \cdots + e_k + 1) = c$. It follows that $c(e_1 + \cdots + e_k) = 0$. Since c is a minimal element, $c = e_i$ for some e_i, which leads us to a contradiction $e_i = e_i(e_1 + \cdots + e_k) = 0$. □

Definition 4 Let $\mathbf{B}$ be a Boolean ring. A quotient ring $\mathbf{B}[X_1, \dots, X_n]/\langle X_1^2 - X_1, \dots, X_n^2 - X_n \rangle$ with an ideal $\langle X_1^2 - X_1, \dots, X_n^2 - X_n \rangle$ becomes a Boolean ring. It is called a Boolean polynomial ring and denoted by $\mathbf{B}(X_1, \dots, X_n)$, its element is called a Boolean polynomial.

Note that a Boolean polynomial of $\mathbf{B}(X_1,\ldots,X_n)$ is uniquely represented by a polynomial of $\mathbf{B}[X_1,\ldots,X_n]$ that has at most degree 1 for each variable X_i. In what follows, we identify a Boolean polynomial with such a representation. Multiple variables such as $X_1,\ldots,X_n$ or $Y_1,\ldots,Y_m$ are abbreviated to $\bar{X}$ or $\bar{Y}$ respectively. Lower small Greek letters such as a,b,c are usually used for elements of a Boolean ring $\mathbf{B}$. The symbol $\bar{a}$ denotes an n-tuple of element of $\mathbf{B}$ for some n. For $\bar{a}=(a_1,\ldots,a_n)$ and $\bar{b}=(b_1,\ldots,b_m)$, $(\bar{a},\bar{b})$ denotes an $n+m$ tuple $(a_1,\ldots,a_n,b_1,\ldots,b_m)$. For a Boolean polynomial $f(\bar{X},\bar{Y})$ with variables $\bar{X}$ and $\bar{Y}$, $f(\bar{a},\bar{Y})$ denote a Boolean polynomial in $B(\bar{Y})$ obtained by specializing $\bar{X}$ with $\bar{a}$.

Definition 5 Let I be an ideal of $\mathbf{B}(X_1,\ldots,X_n)$. For a subset S of $\mathbf{B}$, $V_S(I)$ denotes a subset $\{\bar{a}\in S^n | \forall f\in I f(\bar{a})=0\}$. When $S=\mathbf{B}$, $V_{\mathbf{B}}(I)$ is simply denoted by $V(I)$ and called a variety of I. We say I is satisfiable in S if $V_S(I)$ is not empty. When $S=\mathbf{B}$, we simply say I is satisfiable.

2.2 Boolean extension theorem

Theorem 6 (Boolean extension theorem). Let I be a finitely generated ideal in a Boolean polynomial ring $\mathbf{B}(Y_1,\ldots,Y_m,X_1,\ldots,X_n)$. For any $\bar{b}\in V(I\cap\mathbf{B}(\bar{Y}))$, there exist $\bar{c}\in\mathbf{B}^n$ such that $(\bar{b},\bar{c})\in V(I)$.

proof It suffices to show the theorem for $n=1$. Note first that any finitely generated ideal is principal in a Boolean ring, that is an ideal $\langle f_1,\ldots,f_s\rangle$ is equal to the principal ideal $\langle f_1\vee\cdots\vee f_s\rangle$. Let $I=\langle fX_1+g\rangle$ for some $f,g\in\mathbf{B}(\bar{Y})$. We claim that $I\cap\mathbf{B}(\bar{Y})=\langle fg+g\rangle$. Since $(f+1)(fX_1+g)=fg+g, fg+g\in I\cap\mathbf{B}(\bar{Y})$. Conversely, suppose that $h\in I\cap\mathbf{B}(\bar{Y})$, i.e. there exist $p,q\in\mathbf{B}(\bar{Y})$ such that $h=(pX_1+q)(fX_1+g$. Then, $h=(pf+pg+qf)X_1+qg$. Since $h\in\mathbf{B}(\bar{Y})$, we must have $pf+pg+qf=0$, from which we have $h=qg=fqg+(f+1)qg=g(pf+pg)+(f+1)qg=gp(f+1)+(f+1)qg=(p+q)(f+1)g\in\langle fg+g\rangle$. Suppose now that $\bar{b}\in V(\langle fg+g\rangle)$, that is $f(\bar{b})g(\bar{b})+g(\bar{b})=0$. Let $c=(f(\bar{b})+1)d+g(\bar{b})$ where d can be any element of $\mathbf{B}$. Then $f(\bar{b})c+g(\bar{b})=f(\bar{b})g(\bar{b})+g(\bar{b})=0$. That is $(\bar{b},c)\in V(I)$. □

2.3 Boolean Nullstellensatz

Corollary 7 (Boolean weak Nullstellensatz). For any finitely generated ideal I of a Boolean polynomial ring $\mathbf{B}(X_1,\ldots,X_n)$, the variety $V(I)(\subseteq\mathbf{B}^n)$ of I is an empty set if and only if there exists a non-zero constant element of $\mathbf{B}$ in I.

proof If $I\cap\mathbf{B}=\{0\}$, the above proof also works to show that $V(I)\neq\emptyset$. The converse is trivial. □

Theorem 8 (Boolean strong Nullstellensatz). Let I be a finitely generated ideal of a Boolean polynomial ring $\mathbf{B}(X_1,\ldots,X_n)$ such that $V(I)\neq\emptyset$. Then, for any Boolean polynomial $h(\bar{X})\in$

$\mathbf{B}(\bar{X})$,

$$h(\bar{X}) \in I \quad if\ and\ only\ if\ \ \forall(\bar{b}) \in V(I)\ h(\bar{b}) = 0. \tag{2.1}$$

proof Let $I = \langle f(\bar{X}) \rangle$ and $\mathbf{B}'$ be a Boolean subring of $\mathbf{B}$ generated by all coefficients of $f(\bar{X})$ and $h(\bar{X})$, i.e. $\mathbf{B}'$ is the smallest Boolean subring of $\mathbf{B}$ which includes all coefficients of $f(\bar{X})$ and $h(\bar{X})$. First note that I is also satisfiable in $\mathbf{B}'$ by Boolean weak Nullstellensatz. Secondly note that $\mathbf{B}'$ is finite, because each element of $\mathbf{B}$ is a sum of finite elements which have a form $a_1^{n_1}a_2^{n_2}\cdots a_l^{n_l}$ where $a_1, a_2, \ldots, a_l$ are coefficients of $f(\bar{X})$ and each n_i is either 0 or 1. By Lemma 3, $\mathbf{B}'$ has atomic elements $e_1, \ldots, e_k$ such that $e_ie_j = 0$ for any $i \neq j$ and $e_1 + \cdots + e_k = 1$. Suppose now that $\forall\bar{b} \in V(I)\ h(\bar{b}) = 0$. We certainly have the property:

$$\forall\bar{b} \in \mathbf{B}'^n(f(\bar{b}) = 0 \Rightarrow h(\bar{b}) = 0) \tag{2.2}$$

In order to show $h(\bar{X}) \in I$, we prove the following claims.
Claim 1: $f(b_1, \ldots, b_n) = 0 \Leftrightarrow e_if(e_ib_1, \ldots, e_ib_n) = 0$ for each $i = 1, \ldots, k$.
proof of Claim1 We clearly have $f(b_1, \ldots, b_n) = 0 \Leftrightarrow e_if(e_ib_1, \ldots, e_ib_n) = 0$ for each $i = 1, \ldots, k$. We also have the equation $e_if(b_1, \ldots, b_n) = e_if(e_ib_1, \ldots, e_ib_n)$. The assertion follows from them. ⊠
Claim 2: $\forall(b_1, \ldots, b_n) \in \mathbf{B}'^n(e_if(e_ib_1, \ldots, e_ib_n) = 0 \Rightarrow e_ih(e_ib_1, \ldots, e_ib_n) = 0)$ for each $i = 1, \ldots, k$.
proof of Claim2 Let i be fixed and suppose $e_if(e_ib_1, \ldots, e_ib_n) = 0$ for elements $b_1, \ldots, b_n$ in $\mathbf{B}$. Since I is satisfiable in $\mathbf{B}'$, we have elements $c_1, ..., c_n$ in $\mathbf{B}'$ such that $f(c_1, \ldots, c_n) = 0$. Let $a_j = e_ib_j + (1 + e_i)c_j$ for each $j = 1, \ldots, n$. Then, we have $e_ia_j = e_ib_j$ and $e_ta_j = e_tc_j$ for each $t \neq i$. By Claim 1, we have $f(a_1, \ldots, a_n) = 0$. By the property, we have $h(a_1, \ldots, a_n) = 0$. By Claim 1 again, we have $e_ih(e_ia_1, \ldots, e_ia_n) = 0$ which is equivalent to $e_ih(e_ib_1, \ldots, e_ib_n) = 0$. ⊠
Claim 3: The ideal $\langle e_if(\bar{X}), e_i(Uh(\bar{X}) + 1) \rangle \subseteq \mathbf{B}'(U, \bar{X})$ is unsatisfiable in $\mathbf{B}'$ for each $i = 1, \ldots, k$, where U is a new variable.
proof of Claim 3 Assume that $e_if(b_1, \ldots, b_n) = 0$ for some $(b_1, \ldots, b_n) \in \mathbf{B}'^n$. By Claim 1, we have $e_if(e_ib_1, \ldots, e_ib_n) = 0$. By Claim 2, we have $e_ih(e_ib_1, \ldots, e_ib_n) = 0$. By Claim 1 again, we have $e_ih(b_1, \ldots, b_n) = 0$. Therefore $e_i(Uh(b_1, \ldots, b_n) + 1) = e_i \neq 0$. ⊠

By the last claim and Boolean weak Nullstellensatz, we can see the ideal $\langle e_if(\bar{X}), e_i(Uh(\bar{X}) + 1) \rangle$ contains a non-zero element of $\mathbf{B}'$. Since e_i is an atomic element of $\mathbf{B}'$, it must contain e_i. So, there exist Boolean polynomials $p(U, \bar{X})$ and $q(U, \bar{X})$ of $\mathbf{B}'\ (U, \bar{X})$ such that $e_i = e_if(\bar{X})p(U, \bar{X}) + e_i(Uh(\bar{X}) + 1)q(U, \bar{X})$.

Multiplying $h(\bar{X})$ from both sides and substituting U by 1, we have $e_ih(\bar{X}) = e_if(\bar{X})p(1, \bar{X})$ $h(\bar{X})$, which shows that $e_ih(\bar{X}) \in I$. So, $h(\bar{X}) = e_1h(\bar{X}) + \cdots + e_kh(\bar{X}) \in I$.

The converse is trivial. □

Chapter 3

Boolean Gröbner Bases

A Boolean Gröbner basis is defined as a natural modification of a Gröbner basis in a polynomial ring over a Boolean ring. Though it was introduced in [19, 20] together with a computation algorithm using a special monomial reduction, the same notion was independently discovered by V.Weispfenning in a polynomial ring over a more general coefficient ring, namely, a commutative von Neumann regular ring([30]). In this section, we give a quick review of Boolean Gröbner bases. For the proofs and more detailed descriptions, refer to [30] or [23].

We concentrate on Boolean rings. In what follows, we assume that some term order on a set of power products of variables is given. For a polynomial f in a polynomial ring $\mathbf{B}[X_1, \ldots, X_n](= \mathbf{B}[\bar{X}])$ over a Boolean ring $\mathbf{B}$, we use the notations $LT(f)$, $LM(f)$ and $LC(f)$ to denote the leading power product, the leading monomial and leading coefficient of f respectively. $f - LM(f)$ is also denoted by $Rd(f)$. We also use the notations $LT(F)$ and $LM(F)$ to denote the sets $\{LT(f)|f \in F\}$ and $\{LM(f)|f \in F\}$ for a (possibly infinite) subset F of $\mathbf{B}[\bar{X}]$. $T(\bar{X})$ denotes the set of power products consisting of variables $\bar{X}$.

3.1 Definitions

Definition 9 For an ideal I of a polynomial ring $\mathbf{B}[\bar{X}]$, a finite subset G of I is called a Gröbner basis of I if $\langle LM(I)\rangle = \langle LM(G)\rangle$.

Definition 10 For a polynomial $f \in \mathbf{B}[\bar{X}]$, let $a = LC(f)$, $t = LT(f)$ and $h = Rd(f)$. A monomial reduction $\rightarrow f$ by f is defined as follows:

$$bts + p \rightarrow_f (1 - a)bts + absh + p.$$

(Note that $(bts + p) - ((1 - a)bts + absh + p) = bs(af)$.)
Where s is a term of $T(\bar{X})$, b is an element of $\mathbf{B}$ such that $ab \neq 0$ and p is any polynomial of $\mathbf{B}[\bar{X}]$. For a set $F \subseteq \mathbf{B}[\bar{X}]$, we write $g \rightarrow_F g'$ if and only if $g \rightarrow_f g'$ for some $f \in F$. A recursive closure of $\rightarrow_F$ is denoted by $\overset{*}{\rightarrow}_F$, i.e. $g \overset{*}{\rightarrow}_F g'$ if and only if $g = g'$ or there exist a sequence of monomial reductions g $\rightarrow_F g_1 \rightarrow_F \cdots \rightarrow_F g_n \rightarrow_F g'$.

Theorem 11 When F is finite, $\rightarrow F$ is noetherian, that is there is no infinite sequence of polynomials $g_1, g_2, \ldots$ such that $g_i \rightarrow F g_{i+1}$ for each $i = 1, 2, \ldots$.

Theorem 12 Let I be an ideal of a polynomial ring $\mathbf{B}[\bar{X}]$. A finite subset G of I is a Gröbner basis of I if and only if $\forall h \in I\ h \overset{*}{\rightarrow}_G 0$.

Using our monomial reductions, a reduced Gröbner basis is defined exactly same as in a polynomial ring over a field. A Gröbner basis G is reduced if each polynomial of G is not reducible by a monomial reduction of any other polynomial of G. In a polynomial ring over a field, a reduced Gröbner basis is uniquely determined. In our case, however, this property does not hold.

Example 13 Let $\mathbf{B} = \mathbb{GF}_2 \times \mathbb{GF}_2$. In a polynomial ring $\mathbf{B}[X]$, $\{(1,0)X, (0,1)X\}$ and $\{(1,1)X\}$ are both reduced Gröbner bases of the same ideal.

In order to have a unique Gröbner basis, we need one more definition.

Definition 14 A reduced Gröbner basis G is said to be stratified if G does not contain two polynomials which have the same leading power product.

Theorem 15 If G and G' are stratified Gröbner bases of the same ideal w.r.t. some term order, then $G = G'$.

In the above example, $\{(1,1)X\}$ is the stratified Gröbner basis, but the other is not.

Definition 16 For a polynomial f, $LC(f)f$ is called a Boolean closure of f, and denoted by $bc(f)$. If $f = bc(f)$, f is said to be Boolean closed.

Theorem 17 Let G be a Gröbner basis of an ideal I, then $\{bc(g) | g \in G\}\{0\}$ is also a Gröbner basis of an ideal I.

S-polynomial is also defined similarly as in a polynomial ring over a field.

Definition 18 Let $f = atr + f'$ and $g = bsr + g'$ be polynomials where $a = LC(f)$, $b = LC(g)$, $tr = LT(f)$ and $sr = LT(g)$ for some power product t, s, r such that $GCD(t,s) = 1$, i.e. t and s do not contain a common variable. The polynomial $bsf + atg = bsf' + atg'$ is called an S-polynomial of f and g and denoted by $S(f,g)$.

As in a polynomial ring over a field, the following property is crucial for the construction of Gröbner bases.

Theorem 19 Let G be a finite set of polynomials such that each element of G is Boolean closed. Then, G is a Gröbner basis if and only if $S(f,g) \overset{*}{\rightarrow}_G 0$ for any pair f, g of G.

3.2 Algorithm

For any given finite set F, using our monomial reductions, we can always construct a Gröbner basis of $\langle F\rangle$ with computing Boolean closures and S-polynomials by the following algorithms. It is also easy to construct a stratified Gröbner basis from a Gröbner basis.

```
**Algorithm: BC**
**input:** $F$ a finite subset of $\mathbf{B}[\bar{X}]$
**output:** $F'$ a set of Boolean closed polynomials such that $\langle F'\rangle = \langle F\rangle$ begin
$F' = \emptyset$
while there exists a polynomial $f \in F$ which is not Boolean closed
  $F = F \cup \{bc(f) - f\} \setminus \{f\}, F' = F' \cup \{bc(f)\}$
end.
```

```
**Algorithm: BGB**
**input:** $F$ a finite subset of $\mathbf{B}[\bar{X}]$, $>$ a term order of $T(\bar{X})$
**output:** $G$ a Gröbner basis of $\langle F\rangle$ w.r.t. $>$
begin
$G =$BC$(F)$
while there exists two polynomials $p, q \in G$ such that $S(p,q) \overset{*}{\rightarrow}_G h$ for some non-zero polynomial $h$ which is irreducible by $\rightarrow_G$
  $G = G\cup$ BC$(\{h\})$
end.
```

Since any element of a Boolean ring is idempotent, a Boolean polynomial ring is more natural to work on. We can also define Gröbner bases in Boolean polynomial rings.

A power product $X_1^{l_1}\cdots X_n^{l_n}$ is called a Boolean power product if each l_i is either 0 or 1. The set of all Boolean power products consisting of variables $\bar{X}$ is denoted by $BT(\bar{X})$. A Boolean polynomial $f(\bar{X})$ in $\mathbf{B}(\bar{X})$ is uniquely represented by $b_1t_1 + \cdots + b_kt_k$ with elements $b_1,\ldots,b_k$ of $\mathbf{B}$ and distinct Boolean power products $t_1,\ldots,t_k$. We call $b_1t_1 + \cdots + b_kt_k$ the canonical representation of $f(\bar{X})$. Since $BT(\bar{X})$ is a subset of $T(\bar{X})$, a term order $\geq$ on $T(\bar{X})$ is also defined on $BT(\bar{X})$. Given such a term order $\geq$, we use the same notations $LT(f), LM(f), LC(f)$ and $Rd(f)$ as before, which are defined by using its canonical representation. We also use the same notations $LT(F)$ and $LM(F)$ for a set F of Boolean polynomials as before.

Definition 20 For an ideal I of a Boolean polynomial ring $\mathbf{B}(\bar{X})$, a finite subset G of I is called a Boolean Gröbner basis of I if $\langle LM(I)\rangle = \langle LM(G)\rangle$ in $\mathbf{B}(\bar{X})$.

Using canonical representations of Boolean polynomials, we can also define monomial reductions for Boolean polynomials as Definition 10 and have the same property of Theorem 12. The Boolean closure of a Boolean polynomial is also similarly defined as Definition 16 and the same property of Theorem 17 holds. We can also define a stratified Boolean Gröbner basis as in Definition 14, which is unique w.r.t. a term order. Construction of a Boolean Gröbner basis is very simple. Given a finite set of Boolean polynomials $F \subseteq \mathbf{B}(\bar{X})$. Compute a Gröbner basis G of the ideal $\langle F \cup \{X_1^2 - X_1, \dots, X_n^2 - X_n\}\rangle$ in $\mathbf{B}[\bar{X}]$ w.r.t. the same term order. Then, $G \setminus \{X_1^2 - X_1, \dots, X_n^2 - X_n\}$ is a Boolean Gröbner basis of $\langle F\rangle$ in $\mathbf{B}(\bar{X})$. If G is stratified, then $G \setminus \{X_1^2 - X_1, \dots, X_n^2 - X_n\}$ is also stratified.

Example 21 The following left constraint with unknown set variables X and Y and an unknown element variable a is equivalent to the right system of equations of a Boolean polynomial ring $\mathbf{B}(X, Y, A)$, where $\mathbf{B}$ is a Boolean ring of sets and the variable A stands for the singleton $\{a\}$.

$$\begin{cases} X \cup Y \subseteq \{1,2\} \\ 1 \in X \\ a \in Y \\ X \cap Y = \emptyset \end{cases} \iff \begin{cases} (1 + \{1,2\})(XY + X + Y) = 0 \\ \{1\}X + \{1\} = 0 \\ AY + A = 0 \\ XY = 0 \end{cases}$$

The stratified Boolean Gröbner basis G of the ideal

$I = \langle (1 + \{1,2\})(XY + X + Y), \{1\}X + \{1\}, AY + A, XY\rangle$

w.r.t. a lexicographic term order $X > Y > A$ has the following form:

$G = \{\{2\}XY, \{2\}YA + \{2\}A, (1 + \{2\})Y, \{2\}XA, (1 + \{2\})X + \{1\}, (1 + \{2\})A\}$

From this we can get the elimination ideal $I \cap \mathbf{B}(A) = \langle (1+2)A\rangle$. By Boolean extension theorem, we can see that the given constraint is satisfiable if and only if the element variable a satisfies the equation $(1 + \{2\})\{a\} = 0$ that is $a = 2$.

We conclude this section with the following theorem, which is essentially a special instance of Theorem 2.3 of [30].

Definition 22 Let $\mathbf{B}$ be a Boolean ring and k be a natural number. $\mathbf{B}^k$ denotes a direct product, i.e. the set of all k-tuples of elements of $\mathbf{B}$. For an element p of $\mathbf{B}^k$, $p_i \in \mathbf{B}$ denotes the i-th element of p for each $i = 1, \dots, k$. If we define $p + q$ and $p \cdot q$ for $p, q \in \mathbf{B}^k$ by $(p + q)_i = p_i + q_i$ and $(p \cdot q)_i = p_i \cdot q_i$ for each $i = 1, \dots, k$, $\mathbf{B}^k$ also becomes a Boolean ring. For a polynomial $f(X)$ in $\mathbf{B}^k[\bar{X}] f_i (i = 1, \dots, k)$ denotes the polynomial in $B[\bar{X}]$ obtained by replacing each coefficient p of f by p_i. For a Boolean polynomial $f(\bar{X})$ in $\mathbf{B}^k(\bar{X})$, a Boolean polynomial f_i in $\mathbf{B}(\bar{X})$ is defined similarly.

Theorem 23 In a polynomial ring $\mathbf{B}^k[\bar{X}]$, let G be a finite set of Boolean closed polynomials. Then, G is a (reduced) Gröbner basis of an ideal I if and only if $G_i = \{g_i | g \in G\} \setminus \{0\}$ is a (reduced) Gröbner basis of the ideal $I_i = \{f_i | f \in I\}$ in $\mathbf{B}[\bar{X}]$ for each $i = 1, \dots, k$.

Corollary 24 In a Boolean polynomial ring $\mathbf{B}^i(\bar{X})$, let G be a finite set of Boolean closed Boolean polynomials. Then, G is a (reduced) Boolean Gröbner basis of an ideal I if and only if $G_i = \{g_i | g \in G\} \setminus \{0\}$ is a (reduced) Gröbner basis of the ideal $I_i = f_i | f \in I$ in $\mathbf{B}(\bar{X})$ for each $i = 1, \dots, k$.

Chapter 4

Comprehensive Boolean Gröbner bases

In a polynomial ring over a field, construction of a comprehensive Gröbner basis is not so simple in general. In order to get a uniform (with respect to parameters) representation of reduced Gröbner bases, we need to divide a parameter space into several partitions according to the conditions that parameters satisfy. (See [8, 9, 10, 28, 29, 31].) In our Boolean polynomial ring, however, we can always construct a stratified comprehensive Boolean Gröbner basis. We do not even need to divide a parameter space. In what follows, we use variables $\bar{A} = A_1, \cdots, A_m$ for parameters and variables $\bar{X} = X_1, \cdots, X_n$ for main variables. We also assume that some term order on $T(\bar{X})$ is given.

4.1 Definitions

Definition 25 Let $F = \{f_1(\bar{A}, \bar{X}), \dots, f_l(\bar{A}, \bar{X})\}$ be a finite subset of a Boolean polynomial ring $\mathbf{B}(\bar{A}, \bar{X})$. A finite subset $G = \{g_1(\bar{A}, \bar{X}), \dots, g_k(\bar{A}, \bar{X})\}$ of $\mathbf{B}(\bar{A}, \bar{X})$ is called a comprehensive Boolean Gröbner basis of F, if $G(\bar{a}) = \{g_1(\bar{a}, \bar{X}), \dots, g_k(\bar{a}, \bar{X})\} \setminus \{0\}$ is a Boolean Gröbner basis of the ideal $\langle F(\bar{a})\rangle = \langle f_1(\bar{a}, \bar{X}), \dots, f_l(\bar{a}, \bar{X})\rangle$in $\mathbf{B}'(\bar{X})$ for any Boolean extension $\mathbf{B}'$ of $\mathbf{B}$, i.e. a Boolean ring which includes $\mathbf{B}$ as a subring, and any $a = (a_1, \dots, a_m) \in \mathbf{B}'^m$. G is also said to be stratified if $G(\bar{a})$ is stratified for any $\bar{a} = (a_1, \dots, a_m) \in \mathbf{B}'^m$.

4.2 Naive method

In this section, we present a naive method to construct comprehensive Boolean Gröbner bases.

Theorem 26 Let $F = \{f_1(\bar{A}, \bar{X}), \dots, f_l(\bar{A}, \bar{X})\}$ be a finite subset of a Boolean polynomial ring $\mathbf{B}(\bar{A}, \bar{X})$. Considering $\mathbf{B}(\bar{A}, \bar{X})$ as a Boolean polynomial ring $\mathbf{B}((\bar{A}), \bar{X})$ with the coefficient Boolean ring $\mathbf{B}(\bar{A})$, let $G = \{g_1(\bar{A}, \bar{X}), \dots, g_k(\bar{A}, \bar{X})\}$ be a (stratified) Boolean Gröbner basis of the ideal $\langle F\rangle$ in this polynomial ring. Then G becomes a (stratified) comprehensive Boolean Gröbner basis of F.

proof Let $\bar{a} = a_1, \dots, a_m$ be an arbitrary m-tuple of elements of $\mathbf{B}'$. Note that the specialization of parameters $\bar{A}$ with $\bar{a}$ induces a homomorphism from $\mathbf{B}(\bar{A}, \bar{X})$ to $\mathbf{B}'(\bar{X})$. We clearly have $\langle F(\bar{a})\rangle$

= $\langle G(\bar{a})\rangle$ in $\mathbf{B}'(\bar{X})$.

If $f(\bar{A},\bar{X}) \rightarrow_{g(\bar{A},\bar{X})} h(\bar{A},\bar{X})$ in $(\mathbf{B}(\bar{A}))(\bar{X})$, then $f(\bar{A},\bar{X}) = p(\bar{A})ts + f'(\bar{A},\bar{X})$, $g(\bar{A},\bar{X}) = q(\bar{A})t + g'(\bar{A},\bar{X})$, and $h(\bar{A},\bar{X}) = (1 - q(\bar{A}))p(\bar{A})ts + q(\bar{A})p(\bar{A})sg'(\bar{A},\bar{X}) + f'(\bar{A},\bar{X})$ for some $t, s \in T(\bar{X})$ and $p(\bar{A})$,$q(\bar{A}) \in \mathbf{B}(\bar{A})$ and $f'(\bar{A},\bar{X})$, $g'(\bar{A},\bar{X}) \in \mathbf{B}(\bar{A},\bar{X})$, where $q(\bar{A})t$ is the Boolean leading monomial of $g(\bar{A},\bar{X})$. In case $q(\bar{a})p(\bar{a}) \neq 0$, certainly $q(\bar{a}) \neq 0$ and $p(\bar{a}) \neq 0$, so $q(\bar{a})t$ is the Boolean leading monomial of $g(\bar{a},\bar{X})$ and $p(\bar{a})ts$ is a monomial of $f(\bar{A},\bar{X})$ and $f(\bar{a},\bar{X}) \rightarrow_{g(\bar{a},\bar{X})} h(\bar{a},\bar{X})$. Otherwise, $h(\bar{a},\bar{X}) = f(\bar{a},\bar{X})$. In either case, we have $f(\bar{a},\bar{X}) \overset{*}{\rightarrow}_{g(\bar{a},\bar{X})} h(\bar{a},\bar{X})$. Therefore, if $f(\bar{A},\bar{X}) \rightarrow_{G} h(\bar{A},\bar{X})$ in $(\mathbf{B}(\bar{A}))(\bar{X})$, then we have $f(\bar{a},\bar{X}) \overset{*}{\rightarrow}_{G(\bar{a},\bar{X})} h(\bar{a},\bar{X})$ in $\mathbf{B}'(\bar{X})$. Any Boolean polynomial in the ideal $\langle F(\bar{a})\rangle$ is equal to $f(\bar{a},\bar{X})$ for some Boolean polynomial $f(\bar{A},\bar{X})$ in the ideal $\langle F\rangle$ of $(\mathbf{B}(\bar{A}))(\bar{X})$. Since G is a Boolean Gröbner basis of $\langle F\rangle$, we have $f(\bar{A},\bar{X}) \overset{*}{\rightarrow}_{G} 0$. By the above observation, we have $f(\bar{a},\bar{X}) \overset{*}{\rightarrow}_{G(\bar{a})} 0$. This shows that G is a comprehensive Boolean Gröbner basis of F.

Suppose G is stratified, then any element g of G is Boolean closed.

So, if $LC(g)(\bar{a}) = 0$, then $g(\bar{a},\bar{X})$ must be equal to 0. Therefore, unless $g(\bar{a},\bar{X}) = 0$, we have $LT(g(\bar{a},\bar{X})) = LT(g(\bar{A},\bar{X}))$. Now it is clear that $G(\bar{a})$ is stratified. □

Example 27 For the same example of Example 21, the stratified Boolean Gröbner basis of I in the Boolean polynomial ring $(\mathbf{B}(A))(X,Y)$ has the following form: $\{(\{2\}A + \{2\})XY, (1 + A + \{2\})X + \{1\}A + \{1\}, (1 + A + \{2\})Y + \{2\}A, (1 + \{2\})A\}$. From this, we can get the elimination ideal $I \cap \mathbf{B}(A) = \langle (1 + 2)A\rangle$. Moreover, if we specialize the variable A with $\{a\}$, it becomes the stratified Boolean Gröbner basis $\{X + \{1\}, Y + \{2\}\}$.

The computation of comprehensive Boolean Gröbner bases is much simpler than the computation of usual comprehensive Gröbner bases in polynomial rings over fields. Given a finite set F of Boolean polynomials in $\mathbf{B}(A_1,\ldots,A_m,X_1,\ldots,X_n)$, let G be a (stratified) Boolean Gröbner basis of the ideal $\langle F\rangle$ in the Boolean polynomial ring $(\mathbf{B}(A_1,\ldots,A_m))(X_1,\ldots,X_n)$ over the coefficient Boolean ring $\mathbf{B}(A_1,\ldots,A_m)$, then G is a (stratified) comprehensive Boolean Gröbner basis of F with parameters $A_1,\ldots,A_m$. We can also apply the above method for them, however, when m is not very small we need a huge natural number k for the isomorphism between $\mathbf{B}(A_1,\ldots,A_m)$ and $\mathbb{GF}_2^k$, namely $k \geq 2^m$. Therefor the above method is not feasible when we have many parameters. The next result recently reported in [11] enables us to apply the above method for the computation of comprehensive Boolean Gröbner bases.

4.3 Alternative Method

Theorem 28 Let $G = \{g_1(\bar{A},\bar{X}),\ldots,g_k(\bar{A},\bar{X})\}$ be a Boolean Gröbner basis of $\langle F\rangle$ for $F = \{f_1(\bar{A},\bar{X}),\ldots,f_l(\bar{A},\bar{X})\}$ in a Boolean polynomial ring $\mathbf{B}(\bar{A},\bar{X})$ w.r.t. a block term order $>$ such that $\bar{X} \gg \bar{A}$. Then G is a comprehensive Boolean Gröbner basis of F w.r.t. $>_{\bar{X}}$(restriction of $>$ on $T(\bar{X})$).

Example 29 The following left constraint is same as the previous example except that we have an another unknown variable a for an element. Using another set variable A to represent a singleton set $\{a\}$, it is equivalent to the right system of equations of a Boolean polynomial ring $\mathbf{B}(A, X, Y)$.

$$\begin{cases} X \cup Y \subseteq \{s_1, s_2\} \\ s_1 \in X \\ a \in Y \\ X \cap Y = \emptyset \end{cases} \iff \begin{cases} (1 + \{s_1, s_2\})(XY + X + Y) = 0 \\ \{s_1\}X + \{s_1\} = 0 \\ AY + A = 0 \\ XY = 0 \end{cases}$$

Let $F = \{(1 + \{s_1, s_2\})(XY + X + Y), \{s_1\}X + \{s_1\}, AY + A, XY\}$.
The stratified Boolean Gröbner basis G of F w.r.t. a lexicographic term order such that $X > Y > A$ has the following form:

$$G = \{\{s_2\}XY, \{s_2\}YA + \{s_2\}A, (1 + \{s_2\})Y, \\ \{s_2\}XA, (1 + \{s_2\})X + \{s_1\}, (1 + \{s_2\})A\}.$$

This is not reduced as a Boolean Gröbner basis in $(\mathbf{B}(A))(X, Y)$.
In fact, if we specialize A by $\{s_2\}$, $G(\{s_2\}) \setminus \{0\}$ becomes $\{\{s_2\}XY, \{s_2\}Y + \{s_2\}, (1 + \{s_2\})Y, \{s_2\}X, (1 + \{s_2\})X + \{s_1\}\}$, which is not reduced.
But, from the above comprehensive Boolean Gröbner basis we can easily construct the stratified Boolean Gröbner basis of F in $(\mathbf{B}(A))(X, Y)$:

$$\{(\{s_2\}A + \{s_2\})XY, (A + 1 + \{s_2\})X + \{s_1\}A + \{s_1\}, \\ (A + 1 + \{s_2\})Y + \{s_2\}A, (1 + \{s_2\})A\}.$$

4.4 New Algorithm for Elimination Ideals

In chapter 2, we saw the importance of ideals in Boolean polynomial rings. For a system of Boolean equations given in a form of

$$\begin{cases} f_1(X_1, X_2, \ldots, X_n) = 0 \\ \qquad\vdots \\ f_l(X_1, X_2, \ldots, X_n) = 0 \end{cases} \qquad \cdots \quad (1)$$

with Boolean polynomials $f_1(\bar{X}), f_2(\bar{X}), \ldots, f_l(\bar{X})$ of $\mathbf{B}(\bar{X})$, we can solve it by computing a stratified Boolean Gröbner basis of the ideal $I = \langle f_1(\bar{X}), f_2(\bar{X}), \ldots, f_l(\bar{X})\rangle$ w.r.t. a certain term order. We can also solve many problems concerning it. For example, if we want to decide whether a given Boolean polynomial $h(\bar{X})$ vanishes on every solutions, what we have to do is computing a Boolean Gröbner basis G of I(w.r.t. any term order) and checking the normal form of $h(\bar{X})$ by $\rightarrow_G$ is 0 or not. In any case, as long as we can compute a Boolean Gröbner basis of I, we are almost done. Unfortunately, however, Gröbner bases computations are getting heavier when the number of variables increases. If we are interested in only solutions of some restricted variables, say X_1, X_2, X_3 for example, we do not necessarily need a Boolean Gröbner basis of the whole ideal I, what we need is a Boolean Gröbner basis of the elimination ideal $\langle f_1(\bar{X}), f_2(\bar{X}), \ldots, f_l(\bar{X})\rangle \cap$

$\mathbf{B}(X_1, X_2, X_3)$. If we want to know whether a given polynomial $h(X_1, X_2, X_3)$ consisting of only three variables X_1, X_2, X_3 vanishes on every solution of (1), what we need is not a Boolean Gröbner basis of the whole ideal but a Boolean Gröbner basis of the above elimination ideal. A Gröbner basis of such an elimination ideal is usually obtained by computing a Gröbner basis of the whole ideal w.r.t. a block order $X_1, X_2, X_3 << X_4, \dots, X_n$, i.e. a term order such that each variable X_1, X_2, X_3 is lexicographically less than the other variables.

In this section, we give an algorithm to compute a Boolean Gröbner basis of an elimination ideal without computing a Boolean Gröbner basis of the whole ideal. The next lemma is an easy but important key fact for our algorithm.

Lemma 30 Let I be an ideal of a Boolean polynomial ring $\mathbf{B}(\bar{A}, \bar{X})$ with variables $\bar{A}$ and $\bar{X}$, G be a stratified Boolean Gröbner basis of I in a Boolean polynomial ring $(\mathbf{B}(\bar{A})(\bar{X}))$ w.r.t. some term order of $T(\bar{X})$. Then $G \cap \mathbf{B}(\bar{A})$ is either an empty set or a singleton $\{h(\bar{A})\}$ of a Boolean polynomial of $\mathbf{B}(\bar{A})$. In the latter case, the elimination ideal $I \cap \mathbf{B}(\bar{A})$ is equal to $\langle h(\bar{A}) \rangle$, otherwise it is equal to the trivial ideal $\{0\}$ in $\mathbf{B}(\bar{A})$.

If the computation of such a G were faster than the computation of a Boolean Gröbner basis of I in the Boolean polynomial ring $\mathbf{B}(\bar{A}, \bar{X})$ w.r.t. a block order $\bar{A} << \bar{X}$, it would give us a more efficient algorithm to compute a Boolean Gröbner basis of the elimination ideal. Unfortunately, however, a naive method described in section 4.2 is much slower than the computation of a Boolean Gröbner basis of I w.r.t. such a block order in general. Though there is a work concerning an efficient computation algorithm for such a G, it is based on the computation of a Boolean Gröbner basis of the whole ideal w.r.t. a block order([25]).

In this section, we give a new approach to compute an elimination ideal $I \cap \mathbf{B}(\bar{A})$. The key fact is the following lemma concerning the structure of a Boolean polynomial ring $\mathbf{B}(\bar{A})$.

Lemma 31 A Boolean polynomial ring $\mathbf{B}(A_1, \dots, A_m)$ is isomorphic to the direct product $\mathbf{B}^{2^m}$. An isomorphism ϕ from $\mathbf{B}(A_1, \dots, A_m)$ to $\mathbf{B}^{2^m}$ is given by $\phi(f(A_1, \dots, A_m))_i = f(c^i_1, \dots, c^i_m)$ for each $i = 1, ..., 2^m$, where $c^i_1 \dots c^i_m$ is a binary number representation of $i - 1$. The inverse of ϕ is given by $\phi^{-1}((a_1, a_2, \dots, a_{2^m})) = \sum_{i=1}^{2^m} a_i(A_1 + c^i_1 + 1)(A_2 + c^i_2 + 1) \cdots (A_m + c^i_m + 1)$. (Note that $c^i_k + 1 = 1$ if $c^i_k = 0$ and $c^i_k + 1 = 0$ if $c^i_k = 1$.)

proof Proof is by induction on m. We first show the lemma for $m = 1$. Any Boolean polynomial of $\mathbf{B}(A_1)$ has a form $aA_1 + b$ for some elements a and b of $\mathbf{B}$. By the definition, $\phi(aA_1 + b)_1 = b$ and $\phi(aA_1 + b)_2 = a + b$, that is $\phi(aA_1 + b) = (b, a + b)$. It is easy to check that ϕ is a homomorphism. It is also obvious that ϕ is a bijection. Let $m > 1$ and assume that the lemma holds for $m - 1$. Note that any element of $\mathbf{B}(A_1, \dots, A_m)$ is uniquely represented as $f(A_2, \dots, A_m)A_1 + g(A_2, \dots, A_m)$ with some elements $f(A_2, \dots, A_m)$ and $g(A_2, \dots, A_m)$ of $\mathbf{B}(A_2, \dots, A_m)$. Considering a Boolean polynomial ring $\mathbf{B}(A_1, \dots, A_m)$ as a Boolean polynomial ring $(\mathbf{B}(A_2, \dots., A_m))(A_1)$, it is isomorphic to $\mathbf{B}(A_2, \dots, A_m)^2$ with an isomorphism θ such that $\theta(f(A_2, \dots, A_m)A_1 + g(A_2, \dots, A_m)) = (g(A_2, \dots, A_m), f(A_2, \dots, A_m) + g(A_2, \dots, A_m))$ by the first assertion we have shown above. By the assumption, we have an isomorphism ψ from

$\mathbf{B}(A_2,\ldots,A_m)$ to $\mathbf{B}^{2^{m-1}}$ given by $\psi(f(A_2,\ldots,A_m))_i = f(c_2,\ldots,c_m)$ for each $i = 1,\ldots,2^{m-1}$, $c_2\cdots c_m$ is a binary number representation of $i-1$. Now, a map ϕ from $\mathbf{B}(A_1,\ldots,A_m)$ to $\mathbf{B}^{2^m}$ defined by $\phi(f(\bar{A})) = (\psi(\theta(f(\bar{A}))_1),\ \psi(\theta(f(\bar{A}))_2))$ (the concatenation of two 2^{m-1}-tuples $\psi(\theta(f(\bar{A}))_1)$ and $\psi(\theta(f(\bar{A}))_2)$ satisfies the property of the lemma. The last assertion is obvious. □

This lemma together with Corollary 24 gives us a new algorithm to compute a Boolean Gröbner basis of an elimination ideal.

```
ElimBGB
input: $F$ a finite subset of $\mathbf{B}(\bar{A},\bar{X})$,
      $\bar{A}$ parameter variables, $>$ a term order of $T(\bar{A})$
output: $G$ a Boolean Gröbner basis of the elimination ideal $\langle F\rangle \cap \mathbf{B}(\bar{A})$ w.r.t. $>$.
begin
$F' \leftarrow \{\phi(f) \mid f \in F\}$
  For $i = 1$ to $2^m$, compute the stratified Boolean Gröbner basis
    $G_i$ of the ideal generated by $F'_i = \{f_i \in \mathbf{B}(\bar{A}) \mid f \in F'\}$ w.r.t. any term order of $T(\bar{X})$.
  For $i = 1$ to $2^m$, if $G_i$ contains a non-zero element of $\mathbf{B}$
    then $b_i =$ such an element, else $b_i = 0$.
$G =$ a Boolean Gröbner basis of the ideal $\langle \phi^{-1}((b_1,b_2,\ldots,b_{2^m}))\rangle$ w.r.t. $>$.
end.
```

In the above, ϕ is an isomorphism from $(\mathbf{B}(A_1,\ldots,A_m))(\bar{X})$ to $\mathbf{B}^{2^m}(\bar{X})$ obtained as an extension of the isomorphism defined in Lemma 31. We implemented the algorithm for a Boolean ring $\mathbf{B} = \{S \subseteq St \mid S \text{ is a finite or co-finite set}\}$, where St denotes a countable set of all strings. We show how our algorithm works using an example of our computation. In the following example, we have 30 variables $X_1,\ldots,X_{30}$. $a,b,\ldots,t$ denote strings, so $\{d,p\},\{a,b\},\ldots$ are elements of our $\mathbf{B}$.

Example 32 Compute the eliminate portion $\langle F\rangle \cap \mathbf{B}(X_1,X_2,X_3)$ for
$F = \{f_1(\bar{X}), f_2(\bar{X}),\ldots,f_{18}(\bar{X})\}$ *with*
$f_1(\bar{X}) = X_1X_3X_{18} + \{d,p\}X_4X_{18}X_{20} + X_{11}X_{13} + \{a,d\}X_6X_{10} + X_5X_{18} + X_4,$
$f_2(\bar{X}) = X_2X_5 + X_4X_5 + \{k,q\}X_6X_8 + X_1X_{26}X_{27} + \{c,k\}X_4X_8 + X_{10} + X_6X_{10},$
$f_3(\bar{X}) = X_1X_2X_4 + X_3X_5X_{10} + \{d,i\}X_{11} + X_1 + X_5 + X_{12}X_{24}X_{28},$
$f_4(\bar{X}) = X_2X_4 + \{e,g\}X_3X_4 + X_1X_{12} + X_{12}X_{15} + X_5X_{10}X_{12} + X_3X_{11},$
$f_5(\bar{X}) = X_1X_3 + X_1X_5 + \{j,l\}X_2X_5X_{16} + X_{11}X_{12} + X_{11}X_{23} + X_{16} + 1,$
$f_6(\bar{X}) = X_6X_{17} + X_5X_9X_{30} + \{a,c\}X_8X_{10} + X_1X_{12} + X_{25}X_{29},$
$f_7(\bar{X}) = X_1X_{11} + X_{12} + X_2X_8 + X_3X_{11}X_{12} + X_{11}X_{12} + X_4X_6 + \{b,e\},$
$f_8(\bar{X}) = X_2 + X_4X_7 + \{c,f\}X_{12}X_{17}X_{21} + X_2X_3X_{12} + X_6X_7 + X_{12} + X_4X_{25} + X_1X_{11},$
$f_9(\bar{X}) = X_3 + X_3X_4 + \{k,m\}X_1X_3 + X_5X_6 + \{h,i\}X_7X_{24},$
$f_{10}(\bar{X}) = X_1 + \{g,i\}X_4X_5X_{11} + \{m,r\}X_1X_9X_{11} + X_2X_6 + X_{11} + 1,$
$f_{11}(\bar{X}) = X_3X_{20} + X_5 + X_5X_7 + X_{11} + \{l,o,s\}X_{13}X_{30} + X_{11}X_{18}X_{23},$
$f_{12}(\bar{X}) = X_3X_{14} + \{f,n,t\}X_1X_2 + X_2 + X_{11} + X_{11}X_{15} + X_{19}X_{22},$

$f_{13}(\bar{X}) = X_2X_7 + \{f,j\}X_{11} + X_2X_3 + X_{11}X_{12} + X_9X_{13} + X_{13},$
$f_{14}(\bar{X}) = X_3X_7 + X_8 + \{d,o\}X_8X_{13} + \{c,t\}X_2X_{23} + X_3X_{20}X_{22} + 1,$
$f_{15}(\bar{X}) = X_4X_9 + X_7X_{20} + \{b,l\}X_8X_{19} + X_{20},$
$f_{16}(\bar{X}) = \{a,e,n\}X_7X_9 + X_3X_5 + X_6X_{22} + \{e,r\}X_{18}X_{29} + X_{19}X_{21},$
$f_{17}(\bar{X}) = X_3 + X_{14} + X_{17}X_{18} + X_3X_4X_{19},$
$f_{18}(\bar{X}) = X_7 + X_7X_{21} + X_{23}X_{24}\}$

We apply the algorithm **ElimBGB** for F where X_1, X_2, X_3 are parameters and we use a purely lexicographic term order such that $X_1 < X_2 < X_3$.

The isomorphism ϕ from $\mathbf{B}(X_1, X_2, X_3)$ to $\mathbf{B}^8$ is given by $\phi(f(X_1, X_2, X_3) = (f(0,0,0), f(0,0,1), f(0,1,0), f(0,1,1), f(1,0,0), f(1,0,1), f(1,1,0), f(1,1,1))$.
$F'_1 = \{f_1(0,0,0,X_4,\dots,X_{30}),\dots,f_{18}(0,0,0,X_4,\dots,X_{30})\}$
$F'_2 = \{f_1(0,0,1,X_4,\dots,X_{30}),\dots,f_{18}(0,0,1,X_4,\dots,X_{30})\}$
$\vdots$
$F'_8 = \{f_1(1,1,1,X_4,\dots,X_{30}),\dots,f_{18}(1,1,1,X_4,\dots,X_{30})\}$
Computation of a stratified Boolean Gröbner basis for each F'_i yields
$b_1 = 0, b_2 = \{i\}, b_3 = \{k,q\}, b_4 = 0, b_5 = 0, b_6 = 0, b_7 = 0, b_8 = 0$
$\phi^{-1}((0,\{i\},\{k,q\},0,0,0,0)) = \{i\}(X_1+1)(X_2+1)X_3 + \{k,q\}(X_1+1)X_2(X_3+1)$
We finally have a desired stratified Boolean Gröbner basis

$$G = \{\{i,k,q\}X_1X_2X_3 + \{i,k,q\}X_2X_3 + \{i\}X_1X_3 + \{i\}X_3 \\ + \{k,q\}X_1X_2 + \{k,q\}X_2\}$$

Total computation time is 274 seconds by a PC with 1.7GHZ pentium-M CPU and 2GB SDRAM. Whereas, a Boolean Gröbner basis computation w.r.t. any term order did not terminate in hours, a comprehensive Gröbner basis computation with parameters X_1, X_2, X_3 did not either terminate in hours.
In the algorithm **ElimBGB**, if we use a proper termorder of $T(\bar{X})$ we can also get other eliminate portions. In the above example, we used a purely lexicographic term order such that $X_4 < X_5 < X_6 < \cdots < X_{30}$. From $G_1, G_2, \dots, G_8$, for example, if we use $G_i \cap \mathbf{B}(X_4, X_5)$ instead of using a constant part b_i, we can get an elimination ideal $\langle F\rangle \cap \mathbf{B}(X_1,\dots,X_5)$. From which, we can compute the stratified Boolean Gröbner basis G of the elimination ideal $\langle F\rangle \cap \mathbf{B}(X_3, X_4, X_5)$ w.r.t. a purely lexicographic term order $X_3 < X_4 < X_5$.

$$G = \{\{s,b,i,k,c,e,f,t,m,l\}X_3X_4X_5 + \{i\}X_4X_5 + \{s,b,k,c,e,f,t,m,l\}X_3X_5 \\ + \{s,b,i,k,c,e,f,t,m,l\}X_3X_4 + \{i\}X_4 + \{s,b,i,k,c,e,f,t,m,l\}, \\ \{o\}X_5X_4 + \{o\}X_3X_5 + \{o\}X_4 + \{o\}X_3, \\ (1+\{s,b,i,k,c,h,e,f,t,m,l\})X_3X_4 + (1+\{s,b,i,k,c,h,e,f,t,m,l\})X_3\}$$

What the algorithm **ElimBG** computes is essentially a comprehensive Boolean Gröbner basis of $\langle F\rangle$ with parameters $\bar{A}$. The possible values for each parameter are not only 0 and 1 but also all elements of $\mathbf{B}$. In the example of the last section, possible values for each parameter X_1, X_2, X_3

are all the subsets of $a, b, c, \ldots, s, t$ and their complements. So, there are $(2^{21})^3 = 2^{63}$-many possible cases for all specializations, where as there are only 8 cases for 0 and 1 specializations. In this sense, our algorithm is efficient. For m-many parameters, however, our algorithm needs computations of 2^m-many Boolean Gröbner bases. This is of course infeasible when m is big, our method is effective only when m is small. Complexity of Boolean Gröbner bases computation is exponential in the number of variables in the worst case for both time and spaces. (So, the method based on computations of Boolean Gröbner bases is not quite unreasonable, since it is an NP-hard problem to solve Boolean equations.) Therefore, the method based on the computation of Boolean Gröbner bases w.r.t. some block order or the naive method to compute comprehensive Boolean Gröbner bases described in section 4.2 should be more efficient than our method at least from the theoretical point of view. (In case we have a parallel computation environment with enough computer resources, it does not apply.) Nevertheless, our (sequential) computation experiments show the efficiency of our method. In the experiments, we used randomly generated 32 sets of Boolean polynomials with 20 to 30 variables with 5 parameters. For 15 examples, either a Boolean Gröbner basis computation w.r.t. a block order or a naive comprehensive Boolean Gröbner basis computation did not terminate in hours, whereas we successfully computed the elimination ideals by our method for all examples.

Chapter 5

Implementation on general computer algebra systems

5.1 Computation of Boolean Gröbner bases

Definition 33 Let St denote an enumerable set of strings (of some computer language). $\mathcal{P}_{FC}(St)$ denotes the set of all finite or co-finite subsets of St, i.e. $\mathcal{P}_{FC}(St) = \{S \subset St \mid S$ is finite or $St \setminus S$ is finite$\}$.

In the rest of this chapter we consider $\mathcal{P}_{FC}(St)$ as a Boolean ring and we concentrate on this Boolean ring for the computation of Boolean Gröbner bases. Note that an atomic element in this Boolean ring is nothing but a singleton of a string.

Let $f_1(\bar{X}), f_2(\bar{X}), \ldots, f_l(\bar{X}) \in \mathcal{P}_{FC}(St)(\bar{X})$. When we compute a Boolean Gröbner basis of the ideal $\langle f_1(\bar{X}), f_2(\bar{X}), \ldots, f_l(\bar{X})\rangle$, we do not need to use whole $\mathcal{P}_{FC}(St)$ (which is infinite). Let $\{s_1, s_2, \ldots, s_k\}$ be the set of all string that is contained in a coefficient of some f_i and $e1 = \{s_1\}; e_2 = \{s_2\}, \ldots, e_k = \{s_k\}$. Then, the finite Boolean subring $\mathbf{B}$ generate by $e_1, \ldots, e_k$ is enough to work. By Stone's representation theorem $\mathbf{B}$ is isomorphic to some direct product of $\mathbb{GF}_2$, more precisely it is isomorphic to $\mathbb{GF}_2^{k+1}$. Note that we have an extra atomic element $1 + e_1 + e_2 + \cdots + e_k$ of $\mathbf{B}$.

Definition 34 For any element $b \in \mathbf{B}$, we can express b in the form $b = b_1e_1 + b_2e_2 + \cdots + b_ke_k + b_{k+1}(1 + e_1 + e_2 + \cdots + e_k)$, where $b_i \in \{0, 1\}, 1 \leq i \leq k + 1$. Let θ be an isomorphism from $\mathbf{B}$ to $\mathbb{GF}_2^{k+1}$ defined by $\theta(b) = (b_1, \ldots, b_{k+1})$. For each $i = 1, \ldots, k + 1$, a projection π_i is an epimorphism from $\mathbf{B}$ to $\mathbb{GF}_2$ defined by $\pi_i(b) = \theta(b)_i$(the i-th component of $\theta(b)$). We also define a monomorphism π_i^{-1} from $\mathbb{GF}_2$ to $\mathbf{B}$ by $\pi_i^{-1}(0) = 0$ and $\pi_i^{-1}(1) = e_i$ for each $i = 1, \ldots, k$ and $\pi_{k+1}^{-1}(1) = 1 + e_1 + \cdots + e_k$. θ, π_i and π_i^{-1} are naturally extended to an isomorphisms from $\mathbf{B}(\bar{X})$ to $\mathbb{GF}_2^{k+1}(\bar{X})$, an epimorphism from $\mathbf{B}(\bar{X})$ to $\mathbb{GF}_2(\bar{X})$ and a monomorphism from $\mathbb{GF}_2(\bar{X})$ to $\mathbf{B}(\bar{X})$ respectively.

We can easily prove the following lemma.

Lemma 35 For each $f \in \mathbf{B}(\bar{X})$ $f = \pi_1^{-1}(\pi_1(f)) + \cdots + \pi_{k+1}^{-1}(\pi_{k+1}(f))$.

Since $\mathbb{GF}_2$ is a field, a Boolean Gröbner basis in the Boolean polynomial ring $\mathbb{GF}_2(\bar{X})$ can be computed by a usual Buchberger algorithm and we can compute it in most computer algebra system which has a facility to compute Gröbner bases in a polynomial ring over $\mathbb{GF}_2$.
For a finite set of polynomials $F = \{f_1, \ldots, f_l\} \subset \mathcal{P}_{FC}(St)(\bar{X})$, let $e_1, \ldots, e_k$ and $\mathbf{B}$ be as above. We abuse the notations π_i and π_i^{-1} for a set, i.e. $\pi_i(P) = \{\pi_i(f) \mid f \in P\}$ and $\pi_i^{-1}(Q) = \{\pi_i^{-1}(f) \mid f \in Q\}$.

Given a finite set F of Boolean polynomials in $\mathbf{B}(X_1, \ldots, X_n)$, let $\mathbf{B}'$ be its smallest Boolean subring that contains all coefficients of polynomials in F. Obviously, any Boolean Gröbner basis G of the ideal $\langle F \rangle$ in $\mathbf{B}'(X_1, \ldots, X_n)$ is also a Boolean Gröbner basis of the ideal $\langle F \rangle$ in $\mathbf{B}(X_1, \ldots, X_n)$. Note that $\mathbf{B}'$ is a finite Boolean ring, so it is isomorphic to a direct product $\mathbb{GF}_2^k$ of the Galois field $\mathbb{GF}_2$ for some natural number k.

Computation of Boolean Gröbner bases in $\mathbb{GF}_2(\bar{X})$ is quite easy.

Theorem 36 For a finite set $\{f_1, \ldots, f_l\}$ of Boolean polynomials in $\mathbb{GF}_2(\bar{X})$, let G be a (reduced) Gröbner basis of the ideal $\langle f_1, \ldots, f_l, X_1^2 - X_1, \ldots, X_n^2 - X_n \rangle$ in the polynomial ring $\mathbb{GF}_2[\bar{X}]$ over the field $\mathbb{GF}_2$ w.r.t. some term order. Then $G \setminus \{X_1^2 - X_1, \ldots, X_n^2 - X_n\}$ is a (reduced) Boolean Gröbner basis of the ideal $\langle f_1, \ldots, f_l \rangle$ in $\mathbb{GF}_2(\bar{X})$ w.r.t. the same term order.

Now we are ready to describe our implementation method.

Algorithm: Boolean GB for $\mathcal{P}_{\mathbf{FC}}(\mathbf{St})$
input: F a finite subset of $\mathcal{P}_{FC}(St)(\bar{X})$ and a term order $>$ on $T(\bar{X})$
output: G a reduced Boolean Gröbner basis of $\langle F \rangle$ w.r.t. $>$
For each $i = 1, \ldots, k+1$ compute the reduced Boolean Gröbner basis G_i of the ideal $\langle \pi_i(F) \rangle$ in $\mathbb{GF}_2(\bar{X})$. Set $G = \cup_{i=1}^{k+1} \pi_i^{-1}(G_i)$.

In order to get a stratified Boolean Gröbner basis, we further need the following manipulation.

Algorithm: Stratification for $\mathcal{P}_{\mathbf{FC}}(\mathbf{St})$
input: G a reduced Boolean Gröbner basis in $\mathcal{P}_{FC}(St)(\bar{X})$
output: G' a stratified Boolean Gröbner basis
Let $\{t_1, \ldots, t_s\}$ be the set of all leading terms(i.e. initials) of some polynomial in G. For each $i = 1 \ldots, s$, let $g_i = \sum_{LT(g)=t_i, g \in G} g$. Set $G' = \{g_1, \ldots, g_s\}$.

We conclude this section with a simple example of our method.

Example 37 We show a calculation process of the Boolean Gröbner basis of the following F.

$$F = \begin{cases} (\{e_1, e_2\} + 1) * X * Y + \{e_1\} * X + Y + \{e_2\} \\ X * Y + \{e_1\} * Y + X + \{e_1, e_2\} \end{cases}$$

We compute $\pi_i(F)$ as follows.

$$\pi_1(F) = \begin{cases} X + Y \\ X * Y + Y + X + 1 \end{cases} \quad \pi_2(F) = \begin{cases} Y + 1 \\ X * Y + X + 1 \end{cases} \quad \pi_3(F) = \begin{cases} X * Y + Y \\ X * Y + X \end{cases}$$

We compute the reduced Boolean Gröbner basis G_i of the ideal $\langle \pi_i(F) \rangle$ in $\mathbb{GF}_2(X, Y)$.

$$G_1 = \begin{cases} X + 1 \\ Y + 1 \end{cases} \quad G_2 = \{1\} \quad G_3 = \{X + Y\}$$

We compute $\pi_i^{-1}(G_i)$ as follows.

$$\pi_1^{-1}(G_1) = \begin{cases} \{e_1\} * X + \{e_1\} \\ \{e_1\} * Y + \{e_1\} \end{cases} \quad \pi_2^{-1}(G_2) = \{\{e_2\}\} \quad \pi_3^{-1}(G_3) = \{(\{e_1, e_2\} + 1) * (X + Y)\}$$

Finally, we do Stratification in order to get a stratified Boolean Gröbner basis.

$$G' = \begin{cases} (\{e_2\} + 1) * X + (\{e_1, e_2\} + 1) * Y + \{e_1\} \\ \{e_1\} * Y + \{e_1\} \\ \{e_2\} \end{cases}$$

5.2 Implementation method

We show how we can implement the computation method of Boolean Gröbner bases described in the last section using the only manipulations of polynomial rings over the Galois field $\mathbb{GF}_2$.

In our implementation, an element of $\mathcal{P}_{FC}(S)$ is represented as a polynomial over $\mathbb{GF}_2$. For example, an element $1 + \{s_1, s_2\}$ of $\mathcal{P}_{FC}(S)$ is represented as a polynomial $1 + s_1 + s_2$ of $\mathbb{GF}_2[s_1, s_2]$. Using this representation, a polynomial f of $\mathcal{P}_{FC}(S)[\bar{X}]$ is translated into a polynomial in $\mathbb{GF}_2[s_1, \ldots, s_t, \bar{X}]$, where $s_1, \ldots, s_t$ are all the strings which occurs in some coefficient of f. The smallest Boolean subring of $\mathcal{P}_{FC}(S)$ that contains $s_1, \ldots, s_t$ is isomorphic to the direct product $\mathbb{GF}_2^{t+1}$. Using an isomorphism ψ such that $\psi(\{s_i\})$ is the $(t+1)$-tuple of $0, 1$ such that only the i-th component is 1 and the others are all 0 and $\psi(1 + \{s_1, \ldots, s_t\})$ is the $(t+1)$-tuple of $0, 1$ such that only the last component is 1, we can consider f as a polynomial of $\mathbb{GF}_2^{t+1}[\bar{X}]$. Under this isomorphism, f_i can be computed by simply specializing s_i with 1 and other s_j with 0 for $i = 1, \ldots, t$, for $i = t + 1$ by specializing all variables $s_1, \ldots, s_t$ with 0.

Example 38 $f = (1 + \{s_1, s_2\})XY$ is translated into $XY + s_1 s_2 XY$.
$g = \{s_1, s_2\}X(\{s_1\}Y)$ is translated into $(s_1 + s_2)X s_1 Y = s_1^2 XY + s_1 s_2 XY$.

Note that the second polynomial g could be further simplified to $s_1 XY$, however we do not employ this simplification since it does not affect the above specializations. By this rather lazy strategy together with the computation technique of Boolean Gröbner bases described in Theorem 36, we can construct most part of Boolean Gröbner bases computations by only using facilities of

Risa/Asir. Our codes for the computations of both Boolean Gröbner bases and comprehensive Boolean Gröbner bases consists less than 300 lines.

The following are computation examples by Risa/Asir.

```
[1378] G=cbgb([(1+(s1+s2))*(x*y+x+y),s1*x+s1,s2*y+s2,x*y],
                                [x,y],[],[s1,s2],2,1,1,1)$
[1379] bp_str(G,[x,y],[]);
[1*y+[s2],1*x+[s1]]

[1380] G=cbgb([(1+(s1+s2))*(x*y+x+y),s1*x+s1,a*y+a,x*y],
                                [x,y],[a],[s1,s2],2,1,1,1)$
[1382] bp_str(G,[x,y],[a]);
[([s2]+1)*a,1*a*y+([s2]+1)*y+[s2]*a,
  1*a*x+([s2]+1)*x+[s1]*a+[s1],[s2]*a*y*x+[s2]*y*x]
```

5.3 Efficient Boolean Gröbner bases software

In this section, we give the computation data obtained by SageMath, Risa/Asir, Mathemathica, Maple, and Singlar, in order to consider suitable software to compute Boolean Gröbner bases. In the experiments, we used randomly generated 20 examples which include polynomials over $\mathbb{GF}_2$ with 100 variables by using PolyBoRi command "random_element". Most combinatorial problems are consisted of polynomials which have total degree 1 or 2 because the Boolean operations $\vee, \wedge, \neg$ are defined by $a \vee b = a + b + a \cdot b, a \wedge b = a \cdot b, \neg a = 1 + a$. In fact, polynomials of Example 5 and 6 are constructed of a linear combination of monomials have total degree 1 or 2. In our experiments, we therefore randomly generated polyomials have total degree 1 or 2 like Example 39, 40 and 41. All the computations are done by the same computer with the following spec:

OS: Ubuntu 14.04 LTS 64bit, CPU: Intel(R) Core(TM) i7-3970X, Clock: 3.50GHz, Number of Cores: 6, Memory: 64GB.

We compared the following system:

SageMath Version 6.7: PolyBoRi package with heuristic option False. Risa/Asir Version 20140224: "nd_gr" command. Mathemathica Version 10: Modulus 2 options. Maple: Gröbner package with default options. Singular 3-1-6: "std" command. With the exception of SageMath, we add the following polynomials $\{x_1^2 + x_1,\ x_2^2 + x_2,\ \cdots,\ x_n^2 + x_n\}$ to polynomials F in Examples.

Table 5.1 contains computation time of Gröbner bases (in seconds) of F in examples 39, 40 and 41.

	SageMath	Risa/Asir	Singlar	Mathematica	Maple
Example 39	0.73	0.99	0.27	1723.74	>1 hour
Example 40	3.66	40.49	385.65	>1 hour	>1 hour
Example 41	500.90	> 2 hours	> 2 hours	>2 hours	>2 hours

Table 5.1: Computation time of Gröbner bases (Sec)

For other 8 examples, a Gröbner basis computation by Risa/Asir, Mathematica, Maple and Singlar did not terminate in hours, whereas SageMath successfully computed.

Example 39 $F = \{x_{17}x_{77} + x_{60}x_{85},\ x_4x_{96} + x_{96}x_{99},\ x_{35}x_{84} + x_{39}x_{59},\ x_{23}x_{58} + x_{61}x_{83},\ x_{35}x_{45} + x_{43}x_{76},\ x_{17}x_{51} + x_{75}x_{85},\ x_{49}x_{73} + x_{70},\ x_{28}x_{50} + x_{35}x_{80},\ x_8x_{30} + x_{14}x_{49},\ x_{35}x_{41} + x_{52}x_{54},\ x_{13}x_{29} + x_{17}x_{28},\ x_{21}x_{72} + x_{39}x_{49},\ x_{22}x_{92} + x_{37}x_{38},\ x_{17}x_{55} + x_{57}x_{98},\ x_{14}x_{72} + x_{32}x_{67},\ x_{25}x_{42} + x_{58}x_{80},\ x_1x_{24} + x_{78}x_{96},\ x_{20}x_{41} + x_{58}x_{84},\ x_{20}x_{47} + x_{36}x_{41},\ x_{46}x_{56} + x_{66}x_{75}, x_{26}x_{85} + x_{46}x_{100},\ x_{43}x_{60} + x_{44}x_{69},\ x_5x_{82} + x_6x_{27},\ x_{26}x_{94} + x_{30}x_{65},\ x_1x_{88} + x_{54}x_{90}\}$.

Example 40 $F = \{x_{39} + x_{40}x_{76} + x_{45} + x_{52} + x_{93} + x_{94},\ x_{24} + x_{25}x_{62} + x_{25} + x_{51} + x_{72} + x_{80},\ x_{25}x_{71} + x_{25} + x_{32} + x_{38} + x_{69} + x_{90},\ x_5 + x_{16}x_{30} + x_{22} + x_{35} + x_{65} + x_{96},\ x_7x_{54} + x_{11} + x_{49} + x_{67} + x_{87} + x_{92},\ x_5x_{53} + x_{17} + x_{37} + x_{74} + x_{76} + x_{90},\ x_{17} + x_{44}x_{66} + x_{61} + x_{65} + x_{74} + x_{89},\ x_{16} + x_{33} + x_{34} + x_{57}x_{69} + x_{59} + x_{73},\ x_{10} + x_{35}x_{41} + x_{51} + x_{59} + x_{100} + 1,\ x_2 + x_5x_{47} + x_{56} + x_{91} + x_{92} + x_{95},\ x_{15} + x_{30}x_{96} + x_{30} + x_{37} + x_{84} + x_{86},\ x_1 + x_{53} + x_{76}x_{88} + x_{76} + x_{85} + 1,\ x_{10} + x_{19}x_{22} + x_{35} + x_{50} + x_{58} + x_{92},\ x_{18} + x_{37} + x_{44} + x_{51}x_{59} + x_{59} + x_{77},\ x_{30} + x_{31}x_{73} + x_{38} + x_{45} + x_{78} + x_{89},\ x_{14} + x_{22} + x_{24} + x_{81}x_{98} + x_{92} + x_{98},\ x_{10}x_{31} + x_{19} + x_{39} + x_{40} + x_{41} + x_{44}\}$.

Example 41 $F = \{x_4x_{84} + x_{69}x_{92},\ x_{49}x_{82} + x_{75}x_{89},\ x_{41}x_{73} + x_{58}x_{75},\ x_1x_{43} + x_9x_{50},\ x_2x_{86} + x_{15}x_{79},\ x_{24}x_{34} + x_{45}x_{52},\ x_4x_{48} + x_{22}x_{98},\ x_2x_{23} + x_{51}x_{81},\ x_8x_{77} + x_{10}x_{79},\ x_3x_8 + x_{53}x_{95},\ x_3x_{73} + x_{18}x_{95},\ x_8 + x_{18} + x_{27} + x_{29} + x_{30} + x_{66},\ x_2 + x_9 + x_{63} + x_{73} + x_{79} + x_{97},\ x_{36} + x_{46} + x_{59} + x_{63} + x_{70} + x_{74},\ x_{31} + x_{34} + x_{36} + x_{41} + x_{60} + x_{66},\ x_9 + x_{30} + x_{48} + x_{79} + x_{83} + x_{88},\ x_{26} + x_{47} + x_{67} + x_{85} + x_{88} + x_{100},\ x_1 + x_{42} + x_{53} + x_{55} + x_{86} + x_{98},\ x_{33} + x_{57} + x_{69} + x_{70} + x_{84} + x_{90},\ x_3 + x_{12} + x_{14} + x_{59} + x_{61} + x_{72},\ x_2 + x_{41} + x_{43} + x_{63} + x_{91} + x_{97},\ x_{12} + x_{19} + x_{38} + x_{62} + x_{65} + x_{77},\ x_{20} + x_{32} + x_{36} + x_{50} + x_{98} + 1,\ x_2 + x_4 + x_{17} + x_{40} + x_{85} + x_{92},\ x_{11} + x_{29} + x_{31} + x_{46} + x_{79} + x_{98},\ x_{10} + x_{51} + x_{58} + x_{59} + x_{89} + x_{90},\ x_2 + x_8 + x_{17} + x_{42} + x_{50} + x_{93},\ x_6 + x_{24} + x_{27} + x_{64} + x_{86} + x_{90},\ x_1 + x_6 + x_{47} + x_{67} + x_{74} + x_{85},\ x_{26} + x_{40} + x_{54} + x_{57} + x_{68} + x_{89},\ x_7 + x_{51} + x_{53} + x_{92} + x_{94} + x_{98}\}$.

5.4 Coding of Boolean Gröbner basis computation in SageMath

Our program to compute Boolean Gröbner bases of $\mathcal{P}_{FC}(\{s_1, \dots, s_k\})$ has the following rather simple shape.

```
def bgb(Polys,Vars,Eles):
  B=BooleanPolynomialRing(len(Vars)+len(Eles),
    Eles+Vars,order='lex')
  BPolys=(B.ideal(Polys)).gens()
  BEles=(B.ideal(Eles)).gens()
  Polys_set=divide(BPolys,Eles)
  Bgb_Set=bgb_comp(Polys_set,Vars,Eles)
  Ele_Polys=mulatom(Bgb_Set,BEles)
  Bgb=stratify(Ele_Polys,Eles)
  return Bgb
```

A Boolean polynomial of $\mathcal{P}_{FC}(\{s_1,\ldots,s_k\})(\bar{X})$ is represented by a Boolean polynomial of $\mathbb{GF}_2(s_1,\ldots,s_k,\bar{X})$ considering $s_1,\ldots,s_k$ as indeterminates. For example, a Boolean polynomial $\{green, red\}X_1 + \{blue\}X_2$ is represented by a polynomial $(green + red) * X_1 + blue * X_2$. We input a list of such represented Boolean polynomials in `Polys`, a list of variables, i.e., $\bar{X}$ in `Vars` and a list of elements, i.e., $s_1,\ldots,s_k$ in `Eles`. `BooleanPolynomialRing` is a PolyBoRi command which defines a Boolean polynomial ring $\mathbb{GF}_2(\bar{X}, s_1,\ldots,s_k)$. For the input F of `Polys`, `divide` computes $\pi_i(F)$ for each $i = 1,\ldots,k+1$. `bgb_comp` computes a reduced Gröbner basis G_i of the ideal $\langle \pi_i(F)\rangle$ in $\mathbb{GF}_2(\bar{X})$ for each $i = 1,\ldots,k+1$, which uses the PolyBoRi program `groebner_basis` to compute Gröbner bases of $\mathbb{GF}_2$. `mulatom` is a program to compute $\pi_i^{-1}(G_i)$. Finally `stratify` compute the stratified Boolean Gröbner basis G'.

The following is a computation example of a Boolean Gröbner basis by our program. It compute the stratified Boolean Gröbner basis $\{x + \{s_1\}, y + \{s_2\}\}$ of the ideal $\langle(1 + \{s_1, s_2\})(XY + X + Y), \{s_1\}X + \{s_1\}, \{s_2\}Y + \{s_2\}, XY\rangle$ in a Boolean polynomial ring $\mathcal{P}_{FC}(\{s_1, s_2\})(x, y)$ w.r.t. a lex order such that $x > y$.

```
% sage
sage: load("bgb.sage")
sage: var("x,y,s1,s2")
(x, y, s1, s2)
sage: bgb([(1+s1+s2)*(x*y+x+y),s1*x+s1,s2*y+s2,x*y],[x,y],[s1,s2])
[x + s1, y + s2]
```

5.5 Parallel Boolean Gröbner bases computation

In this section we describe a parallelization method for Boolean Gröbner basis computation and its implementation. In section 5.1 we see that each computation of $\langle \pi_i(F)\rangle$ for $i = 1,\ldots,k$ is done independently. Hence, the computation of the reduced Gröbner basis G_i of the ideal $\langle \pi_i(F)\rangle$ in $\mathbb{GF}_2(\bar{X})$ is also done independently. Therefore we can easily have an algorithm for parallel Boolean Gröbner basis computation. It is also easy to implement a parallel Boolean Gröbner basis computation by using a decorator which gives a function of a parallel interface. This decorator represented by "@parallel" is supported in SageMath. Our program to compute a Boolean Gröbner basis for $\mathcal{P}_{FC}(\{s_1, \ldots, s_k\})$ has the following rather simple shape.

```
@parallel()
def parallel_gb(In):
  I = ideal(In[1])
  BG=I.groebner_basis(heuristic=False)
  return [In[0],BG]

def parallel_bgb(Polys,Vars,Eles):
  B = BooleanPolynomialRing(len(Vars)+len(Eles), Vars+Eles,
    order='lex')
  BPolys=(B.ideal(Polys)).gens()
  BVars=(B.ideal(Vars)).gens()
  BEles=(B.ideal(Eles)).gens()
  Polys_set=divide(BPolys,BVars,BEles)
  Input=[[i,Polys_set[i]] for i in range(len(Polys_set))]
  Output=list(parallel_gb(Input))
  Bgb_set=[]
  Bgb_set=[Bgb_set+Output[j][1][1] for i in
    range(len(Output))
    for j in range(len(Output))if Output[j][1][0]==i]
  Ele_BPolys=mulatom(Bgb_set,BEles)
  Bgb=stratify(Ele_BPolys,BVars)
  return Bgb
```

`BooleanPolynomialRing` and `divide` are the same functions as those of `bgb` in the previous section. `parallel_gb` computes a reduced Gröbner basis G_i of the ideal $\langle \pi_i(F) \rangle$ in $\mathbb{GF}_2(\bar{X})$ for each $i = 1, \ldots, k$ in parallel. After that, an output of `parallel_gb` stored in an array is sorted. `mulatom` and `stratify` are the same functions as those of `bgb`.

We give a snapshot of timing data of the following example of parallel Boolean Gröbner basis computation. In the example, we have 40 variables $x_1, \cdots, x_{40}$. The symbols $e_1, e_2, \cdots, e_{10}$ denote strings, so $\{e_1\}$, $\{e_2\}$, $\cdots$ are elements of our $\mathbf{B}$.

Example 42 $F = \{x_8x_{40} + x_{11}x_{15} + x_{13}x_{30} + x_{23}x_{27}, x_{19} + x_{26}x_{38},\ \{e_1\}x_{14}x_{40} + x_{15}x_2 + x_1,\ \{e_7\}x_{26} + \{e_{10}\}x_{37} + \{e_7\},\ x_8x_{23} + x_{11}x_{39} + x_{16}x_{18},\ x_{25}x_{27} + \{e_4\}x_{35},\ \{e_5\}x_{12} + x_{33}x_4 + x_{17}x_6 + x_{33},\ \{e_4\}x_{10} + x_{22}x_{24} + x_{12}x_3,\ \{e_1\}x_{27} + \{e_6\}x_{34},\ \{e_6\}x_{14}x_2 + x_{20}x_{28} + x_{27}x_{38} + x_{31}x_{38} + e_9,\ \{e_1\}x_{18}x_{37} + x_{12}x_{16} + x_1x_{20} + x_{22},\ x_4x_{13} + x_5,\ x_{10}x_{14}x_{32},\ \{e_6\}x_{16}x_{22}, \{e_1\}x_{22}x_{37} + \{e_8\}x_{25},\ \{e_3\}x_{10} + x_{12}x_{30},\ \{e_4\}x_{15} + \{e_9\}x_{22} + x_{25},\ \{e_5\}x_{35} + x_{12} + x_7,\ \{e_2\}x_2x_{29} + \{e_7\}x_{21}x_{33} + \{e_8\}x_{36}x_9,\ x_1 + x_8 + x_{15} + x_{16} + x_{22},\ \{e_3\}x_7 + \{e_3\},\ \{e_8\}x_{28} + \{e_8\}\}$

The computation is done by the computer with OS: Ubuntu 14.04 LTS 64bit, Software: SageMath 7.1, CPU: Intel(R) Core(TM) i7-3970X, Clock: 3.50GHz, Number of Cores: 6, Memory: 64GB. Total computation time is 16.7 seconds using parallel Boolean Gröbner basis computation. Whereas, sequential computation time is 70 seconds.

```
sage: load("bgb.sage")
sage: %time B=parallel_bgb(Polys_ex_para,Vars_ex_para,Eles_ex_para)
For the element 3 GB Computation time  2.95557999611
For the element 4 GB Computation time  4.66137695312
For the element 8 GB Computation time  4.94551992416
For the element 1 GB Computation time  7.18508601189
For the element 9 GB Computation time  9.36309504509
For the element 10 GB Computation time  10.1585040092
For the element 2 GB Computation time  11.1377859116
For the element 7 GB Computation time  11.1130139828
For the element 11 GB Computation time  11.1987161636
For the element 5 GB Computation time  13.8662071228
For the element 6 GB Computation time  14.5633881092
CPU times: user 4.82 s, sys: 153 ms, total: 4.97 s
Wall time: 16.7 s
```

This parallel computation is essentially same as the parallel computation introduced in [22, 24]. Though we have satisfactory speed-up for this example, there are two problems for this type of parallel computation. One is that we can expect only $n + 1$-times speed-up, where n is the number of elements. The another one is that, unless each computation has almost same computation time, we can not expect enough speed-up. For example, if two computations need 5 minutes and other need only a few seconds then the speed-up can be only double.

As is described in the introduction, we need to compute at most 10 Boolean Gröbner bases for solving one Sudoku puzzle. Since each Boolean Gröbner basis computation is done within 1 seconds, we do not need parallel computation for the solver of a Sudoku puzzle. For the computation of the s-rank of a Sudoku puzzle, we need to compute much more Boolean Gröbner bases. Since they are divided into many independent computations, parallel and distributed computation of Boolean Gröbner bases is effective for its computation.

5.6 Distributed Boolean Gröbner bases computation

In this section we describe how we implemented distributed Boolean Gröbner basis computation. We have implemented distributed computation using "multiprocessing" module in Python which provide the remote manager. Client code is as follows.

```
from multiprocessing.managers import BaseManager
class QueueManager(BaseManager):pass
QueueManager.register('bgb_remote')
QueueManager.register('parallel_bgb_remote')

def distributed_bgb(Polys,Vars,Eles,IP,Port,Parallel=False):
  B = BooleanPolynomialRing(len(Vars)+len(Eles), Vars+Eles,
    order='lex')
  m = QueueManager(address=(IP,Port),authkey='abracadabra')
  m.connect()
  if Parallel==True:
    Bgb = (m.parallel_bgb_remote(Polys,Vars,Eles)).
      _getvalue()
  else:
    Bgb = (m.bgb_remote(Polys,Vars,Eles))._getvalue()
  return Bgb
```

Clients setup IP address and Port number of a distributed computing server. Basically, clients set any password to “authkey” for the authentication when “BaseManager” object is used. We fix “authkey” here for simplicity. Next, server code is as follows.

```
from multiprocessing.managers import BaseManager
class QueueManager(BaseManager):pass

def _bgb(polys,vars,eles):
  print("bgb call ")
  w=walltime()
  Bgb=bgb(polys,vars,eles)
  wtime=walltime(w)
  print '\033[94m'+"BGB Computation time "+'\033[0m',wtime
  return Bgb

def _parallel_bgb(polys,vars,eles):
  print("parallel bgb call ")
  return parallel_bgb(polys,vars,eles)

if __name__ == '__main__':
  load("bgb.sage")
  QueueManager.register('bgb_remote', callable=_bgb)
  QueueManager.register('parallel_bgb_remote', callable=
    _parallel_bgb)
  m = QueueManager(address=("127.0.0.1",50000),authkey=
    'abracadabra')
  s = m.get_server()
  s.serve_forever()
```

When client calls `bgb_remote` function, server executes `_bgb` function. Although there are other implementation methods to distribute, this implementation method is very simple and easy to deal with.

The following computation example shows how to use our program. It compute the stratified Boolean Gröbner basis $\{x + \{s_1\}, y + \{s_2\}\}$ of the ideal $\langle (1 + \{s_1, s_2\})(XY + X + Y), \{s_1\}X + \{s_1\}, \{s_2\}Y + \{s_2\}, XY \rangle$ in a Boolean polynomial ring $\mathcal{P}(\{s_1, s_2\})(x, y)$ w.r.t. a lex order such that $x > y$.

```
sage: load("bgb.sage")
sage: var("x,y,s1,s2")
(x, y, s1, s2)
sage: distributed_bgb([(1+s1+s2)*(x*y+x+y),s1*x+s1,s2*y+s2,x*y],
....: [x,y],[s1,s2],"127.0.0.1",50000,Parallel=True)
[x + s1, y + s2]
```

Parallel computation is available as an option. In this way, we can easily use distributed Boolean Gröbner basis computation.

5.7 Coding of Comprehensive Boolean Gröbner bases

In this section we introduce implement methods of Comprehensive Boolean Gröbner bases in SageMath. In a polynomial ring over a field, construction of a comprehensive Gröbner basis is not so simple in general. Whereas, construction of comprehensive BooleanGröbner bases is very simple.

There are three methods to compute a comprehensive Boolean Gröbner basis. The first method, let G be a Boolean Gröbner basis of I in $\mathbf{B}^{2^m}(\bar{X})$, then $\varphi^{-1}(G)$ becomes a comprehensive Boolean Gröbner basis of I with parameters $\bar{A}$, since a Boolean polynomial ring $\mathbf{B}(A_1, \ldots, A_m)$ is isomorphic to the direct product $\mathbf{B}^{2^m}$, there exists an isomorphism φ from $(\mathbf{B}(A_1, \ldots, A_m))(\bar{X})$ to $\mathbf{B}^{2^m}(\bar{X})$. The second method, a comprehensive Boolean Gröbner basis of I in $\mathbf{B}(\bar{A}, \bar{X})$ with main variables $\bar{X}$ and parameters $\bar{A}$ can be obtained by simply computing a usual Boolean Gröbner basis of I regarding both $\bar{X}$ and $\bar{A}$ as variables with a certain block term order such that $\bar{X} \gg \bar{A}$(see [11]). The third method, let G be a Boolean Gröbner basis of I in $(\mathbf{B}(\bar{A}))(\bar{X})$, then G becomes a comprehensive Gröbner basis of I with parameters $\bar{A}$ (see [26]). For m-many parameters, the first method needs computations of 2^m-many Boolean Gröbner bases, so the first method is effective empirically when m is small. We have implemented the first method and the second method in SageMath. Our comprehensive BooleanGröbner bases program run both the first and the second methods in parallel and return output of a faster one. Using only each method is available as an option.

First of all, we review a calculation process of the first method and the second method through the following example.

Example 43 Let $F = \{(1+\{s_1,s_2\})(X_1X_2+X_1+X_2), \{s_1\}X_1+\{s_1\}, AX_2+A, X_1X_2\}$.

When using the first method, we obtain the following Boolean Gröbner bases in Boolean polynomial ring $\mathbf{B}^2(X_1,X_2)$.

$$\{G_1 = \{\{s_2\}X_1X_2, (\{s_2\}+1)X_1+\{s_1\}, (\{s_2\}+1)X_2\}, G_2 = \{\{s_2\}X_1, \{s_2\}X_2+\{s_2\}, \{s_2\}+1\}\}$$

After φ^{-1}, we obtain the following Boolean Gröbner bases in Boolean polynomial ring $B(A)(X_1,X_2)$.

$$\{\varphi^{-1}(G_1) = \{\{s_2\}AX_1X_2+\{s_2\}X_1X_2, (\{s_2\}+1)AX_1+\{s_1\}+(\{s_2\}+1)X_1+\{s_1\},$$
$$(\{s_2\}+1)AX_2+(\{s_2\}+1)X_2\},$$
$$\varphi^{-1}(G_2) = \{\{s_2\}AX_1, \{s_2\}AX_2+\{s_2\}A, (\{s_2\}+1)A\}\}$$

Finally, we obtain the following stratified Comprehensive Boolean Gröbner basis.
$G = \{\{s_2\}AX_1X_2+\{s_2\}X_1X_2, (A+\{s_2\}+1)X_1+\{s_1\}A+\{s_1\}, (A+\{s_2\}+1)X_2+\{s_2\}A, (\{s_2\}+1)A\}$
Next, we describe the second method. We obtain the following Comprehensive Boolean Gröbner basis of F by computing a Boolean Gröbner basis of F w.r.t. a lexicographic term order such that $X > Y > A$.

$$G = \{\{s_2\}X_1X_2, \{s_2\}AX_1, (\{s_2\}+1)X_1+\{s_1\}, \{s_2\}AX_2+\{s_2\}A, (\{s_2\}+1)X_2, (\{s_2\}+1)A\}$$

We can implement above two methods easily. Regarding the first method, we need to implement φ for the input F but this function which perform an operation from $(\mathbf{B}(A_1,\dots,A_m))(\bar{X})$ to $\mathbf{B}^{2^m}(\bar{X})$ can be made in the same way as `divide` function in our program for a Boolean Gröbner basis computation. Regarding the second method, we run the program of Boolean Gröbner basis w.r.t. a lexicographic term order such that $\bar{X} > \bar{A}$.

The following computation example shows how to use our program. It compute the comprehensive Boolean Gröbner basis of above example $F = \{(1+\{s_1,s_2\})(X_1X_2+X_1+X_2), \{s_1\}X_1+\{s_1\}, AX_2+A, X_1X_2\}$

```
sage: load("cbgb.sage")
sage: var("X1,X2,A,s1,s2")
(X1, X2, A, s1, s2)
sage: F=[(1+s1+s2)*(X1*X2+X1+X2),s1*X1+s1,A*X2+A,X1*X2]
sage: cbgb(F,[X1,X2],[s1,s2],[A])
Using BGB method with X>A
[X1*X2*s2, X1*A*s2, X1*s2 + X1 + s1, X2*A*s2 + A*s2, X2*s2 + X2,
A*s2 + A]
```

It is easy to implement the first method and the second method, but it is not easy to kill the other one after a faster one finished when two methods are run in parallel. In general, a parent process can not produce grandchild processes to prevent creating zombie processes. So we create children processes which perform the first method and the second method on a same parent process and kill the other one after a faster one finished. We show how to create processes for a computation of a Boolean Gröbner basis in the following program.

```
queue = multiprocessing.Queue()
B = BooleanPolynomialRing(len(Vars)+len(Eles), Vars+Eles,
  order='lex')
BPolys=(B.ideal(Polys)).gens()
BVars=(B.ideal(Vars)).gens()
BEles=(B.ideal(Eles)).gens()
Polys_set=divide(BPolys,BVars,BEles)
In=[[i,Polys_set[i],Vars,Eles] for i in range(
  len(Polys_set))]
jobs=[]
for i in range(len(In)):
  p = Process(target=bgb_comp_using_process, args=(
  i,In[i][1],In[i][2],In[i][3],queue))
  jobs.append(p)

for p in jobs:
  p.start()

Output=[]
for p in range(len(jobs)):
  Output.append(queue.get())
return Output

def bgb_comp_using_process(i,Polys,Vars,Eles,queue):
  B = BooleanPolynomialRing(len(Vars)+len(Eles), Vars+Eles,
    order='lex')
  if Polys != [B.zero()] :
    I = ideal(Polys)
    BG=I.groebner_basis(heuristic=False)
  else:
    BG=[B.zero()]
  queue.put([i,BG])
```

`Process` is a API offered in the multiprocessing package, “Process(target=bgb_comp_using _process, args=(i,In[i][1],In[i][2],In[i][3],queue))” create a process object of calculating a Boolean Gröbner basis. Then, `p.start()` start processes and `queue.put` insert a result into a queue. We obtain a result of a Boolean Gröbner basis by `queue.get`. In this way, we create process objects of the first method and the second method, and compute them in parallel. In the above snapshot, the second method finished faster than the first method. We give an efficient example of the first method.

Example 44 $F = \{f_1(\bar{X}), f_2(\bar{X}), \ldots, f_{18}(\bar{X})\}$ have 27 variables $X_4, \cdots, X_{30}$ and 3 parameters A_1, A_2, A_3.
$f_1(\bar{X}) = A_1A_3X_{18} + \{d,p\}X_4X_{18}X_{20} + X_{11}X_{13} + \{a,d\}X_6X_{10} + X_5X_{18} + X_4,$
$f_2(\bar{X}) = A_2X_5 + X_4X_5 + \{k,q\}X_6X_8 + A_1X_{26}X_{27} + \{c,k\}X_4X_8 + X_{10} + X_6X_{10},$
$f_3(\bar{X}) = A_1A_2X_4 + A_3X_5X_{10} + \{d,i\}X_{11} + A_1 + X_5 + X_{12}X_{24}X_{28},$
$f_4(\bar{X}) = A_2X_4 + \{e,g\}A_3X_4 + A_1X_{12} + X_{12}X_{15} + X_5X_{10}X_{12} + A_3X_{11},$
$f_5(\bar{X}) = A_1A_3 + A_1X_5 + \{j,l\}A_2X_5X_{16} + X_{11}X_{12} + X_{11}X_{23} + X_{16} + 1,$
$f_6(\bar{X}) = X_6X_{17} + X_5X_9X_{30} + \{a,c\}X_8X_{10} + A_1X_{12} + X_{25}X_{29},$
$f_7(\bar{X}) = A_1X_{11} + X_{12} + A_2X_8 + A_3X_{11}X_{12} + X_{11}X_{12} + X_4X_6 + \{b,e\},$
$f_8(\bar{X}) = A_2 + X_4X_7 + \{c,f\}X_{12}X_{17}X_{21} + A_2A_3X_{12} + X_6X_7 + X_{12} + X_4X_{25} + A_1X_{11},$
$f_9(\bar{X}) = A_3 + A_3X_4 + \{k,m\}A_1A_3 + X_5X_6 + \{h,i\}X_7X_{24},$
$f_{10}(\bar{X}) = A_1 + \{g,i\}X_4X_5X_{11} + \{m,r\}A_1X_9X_{11} + A_2X_6 + X_{11} + 1,$
$f_{11}(\bar{X}) = A_3X_{20} + X_5 + X_5X_7 + X_{11} + \{l,o,s\}X_{13}X_{30} + X_{11}X_{18}X_{23},$
$f_{12}(\bar{X}) = A_3X_{14} + \{f,n,t\}A_1A_2 + A_2 + X_{11} + X_{11}X_{15} + X_{19}X_{22},$
$f_{13}(\bar{X}) = A_2X_7 + \{f,j\}X_{11} + A_2A_3 + X_{11}X_{12} + X_9X_{13} + X_{13},$
$f_{14}(\bar{X}) = A_3X_7 + X_8 + \{d,o\}X_8X_{13} + \{c,t\}A_2X_{23} + A_3X_{20}X_{22} + 1,$
$f_{15}(\bar{X}) = X_4X_9 + X_7X_{20} + \{b,l\}X_8X_{19} + X_{20},$
$f_{16}(\bar{X}) = \{a,e,n\}X_7X_9 + A_3X_5 + X_6X_{22} + \{e,r\}X_{18}X_{29} + X_{19}X_{21},$
$f_{17}(\bar{X}) = A_3 + X_{14} + X_{17}X_{18} + A_3X_4X_{19},$
$f_{18}(\bar{X}) = X_7 + X_7X_{21} + X_{23}X_{24}\}$

The computation is done by the computer with OS: Ubuntu 14.04 LTS 64bit, Software: SageMath 7.1, CPU: Intel(R) Core(TM) i7-3970X, Clock: 3.50GHz, Number of Cores: 6, Memory: 64GB. Total computation time is 4.9 seconds using the first method. This example is same as Example 4 in [26], it took 274 seconds with 1.7GHZ pentium-M CPU and 2GB SDRAM at that time. We use a richer computer but we obtain farther more speed-up. Whereas, the second method did not terminate within 1 hour.

Chapter 6

Applications

6.1 Boolean Gröbner bases computation of a finite powerset algebra

Definition 45 Let S be an arbitrary set and $\mathcal{P}(S)$ be its power set, i.e., the family of all subsets of S. Then, $(\mathcal{P}(S), \vee, \wedge, \neg)$ becomes a Boolean algebra with the operations $\vee, \wedge, \neg$ as union, intersection and the complement of S respectively. It is called a *powerset algebra* of S.

Let k be its cardinality. Then the Boolean ring $\mathbf{B}$ of the powerset algebra $\mathcal{P}(S)$ is isomorphic to the direct product $\mathbb{GF}_2^k$. More precisely, let $S = \{a_1, a_2, \dots, a_k\}$ then the isomorphism θ from $\mathcal{P}(S)$ to $\mathbb{GF}_2^k$ is defined by $\theta(A) = (e_1, e_2, \dots, e_k)$ for each $A \subseteq S$, where $e_i = 1$ if $a_i \in A$ and $e_i = 0$ if $a_i \notin A$ for each $i = 1, \dots, k$.

For an element $v \in \mathbb{GF}_2^k$, $\pi_i(v)$ denotes the i-th component of v. This projection is naturally extended to a Boolean polynomial of $\mathbb{GF}_2^k(\bar{X})$. The following theorem reduces the computation of a Boolean Gröbner basis of a Boolean polynomial ring $\mathbb{GF}_2^k(\bar{X})$ to the computation of Boolean Gröbner bases of $\mathbb{GF}_2(\bar{X})$.

Theorem 46 In a Boolean polynomial ring $\mathbb{GF}_2^k(\bar{X})$, let G be a finite set of Boolean closed polynomials. Then, G is a (reduced) Boolean Gröbner basis of an ideal I in $\mathbb{GF}_2^k(\bar{X})$ if and only if $\pi_i(G) = \{\pi_i(g) | g \in G\} \setminus \{0\}$ is a (reduced) Gröbner basis of the ideal $\pi_i(I) = \{\pi_i(f) | f \in I\}$ in $\mathbb{GF}_2(\bar{X})$ for each $i = 1, \dots, k$.

For each $i = 1, \dots, k$, define a map ϕ_i from $\mathbb{GF}_2$ to $\mathbb{GF}_2^k$ by $\phi_i(0) = (0, \dots, 0)$ and $\phi_i(1) = (e_1, \dots, e_k)$ where $e_i = 1$ and $e_j = 0$ for any j such that $j \neq i$. It is also naturally extended to a map from $\mathbb{GF}_2(\bar{X})$ to $\mathbb{GF}_2^k(\bar{X})$.

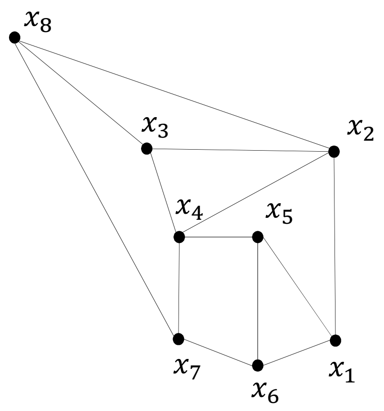
Algorithm: Boolean GB of power set algebra
input: F a finite subset of $\mathbb{GF}_2^k(\bar{X})$ and a term order $>$ on $T(\bar{X})$
output: G a reduced Boolean Gröbner basis of $\langle F \rangle$ w.r.t. $>$
For each $i = 1, \dots, k$ compute the reduced Boolean Gröbner basis G_i of the ideal $\langle \pi_i(F) \rangle$ in

$\mathbb{GF}_2(\bar{X})$. Set $G = \cup_{i=1}^{k} \phi_i(G_i)$.

Algorithm: Stratification of power set algebra
input: G a reduced Boolean Gröbner basis in $\mathbb{GF}_2^k(\bar{X})$
output: G' a stratified Boolean Gröbner basis
Let $\{t_1, \ldots, t_s\}$ be the set of all leading terms of some polynomial in G. For each $i = 1 \ldots, s$, let $g_i = \sum_{LT(g)=t_i, g \in G} g$. Set $G' = \{g_1, \ldots, g_s\}$.

Definition 47 Let I be an ideal of $\mathbf{B}(\bar{X})$. For a subset A of $\mathbf{B}^n$, $V_A(I)$ denotes a subset $\{\bar{a} \in A | \forall f \in I f(\bar{a}) = 0\}$. When A =$\mathbf{B}^n$, $V_A(I)$ is simply denoted by $V(I)$ and called a *variety* of I. We say I is *satisfiable* in A if $V_A(I)$ is not empty. When $A = \mathbf{B}^n$, we simply say I is *satisfiable*.

Example 48 Consider the coloring problem of the following graph by three colors, *green*, *blue* and *red*. Let S be a finite set $\{green, blue, red\}$ and $\mathbf{B}$ be the powerset algebra of S. Let $Sing$ denote the subset of $\mathbf{B}^8$ defined by $Sing = \{(s_1, \ldots, s_8) \in \mathbf{B}^8 |$ each s_i is a singleton$\}$. Without loss of generality we can assume x_1 is assigned to *green* and x_2 is to *blue*. Then the problem is equivalent to computing the variety $V_{Sing}(I)$ for the ideal $I = \langle x_1 + \{green\}, x_2 + \{blue\}, x_1x_2, x_1x_5, x_1x_6, x_2x_3, x_2x_4, x_2x_8, \ldots, x_7x_8\rangle$ of $\mathbf{B}(x_1, x_2, \ldots, x_8)$.



The graph of Example 48

4				9				
3						1	8	
			5					
		5	8					
	2	9						
					1	7		
		6					5	
					7			
			2					9

Example of a Sudoku puzzle

Example 49 Consider the above Sudoku puzzle. We associate a variable X_{ij} for each grid at the i-th row and the j-th column. This puzzle can be considered as a set constraint where each variable should be assigned a singleton from 9 candidates $\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{7\},\{8\}$ and $\{9\}$ so that any distinct two variables which lie on a same row, column or block must be assigned different singletons. 17 variables are assigned singletons $X_{11}=\{4\},X_{15}=\{9\},\ldots,X_{99}=\{9\}$ as the initial conditions. Let $\mathbf{B}$ be a powerset algebra of the finite set $\{1,2,\ldots,9\}$. This constraint is translated into a system of equations of the Boolean polynomial ring $\mathbf{B}(X_{11},X_{12},\ldots,X_{99})=\mathbf{B}(\bar{X})$ as follows:

(1) $X_{11}+\{4\}=0, X_{15}+\{9\}=0,\ldots,X_{99}+\{9\}=0.$
(2) $X_{ij}X_{i'j'}=0$ for each pair of distinct variables $X_{ij},X_{i'j'}$ which lie on a same row, column or block.
(3) $\sum_{(i,j)\in A}X_{ij}+1=0$ where A is a set of indices lying on a same row, column or block. (There are 27 such A's. Remember that $1=\{1,2,\ldots,9\}$.)

Let I be the ideal of $\mathbf{B}(\bar{X})$ generated by the corresponding polynomials of (1),(2) and (3). Let $Sing$ denote the subset of $\mathbf{B}^{81}$ defined by $Sing=\{(s_1,s_2,\ldots,s_{81})\in\mathbf{B}^{81}|$ each s_i is a singleton $\}$. Then the puzzle is equivalent to computing the variety $V_{Sing}(I)$.

The above examples are so-called *singleton set constraints*. We can handle such a constraint by the computation of Boolean Gröbner bases of a finite powerset algebra. See [13] for more details.

6.2 Sudoku Solver

Firstly, we give the computation data obtained by our program of SageMath and the previous program of Risa/Asir [12]. We also optimized programs of both Risa/Asir and SageMath for Sudoku puzzle. Next, we give some computation data of the s-rank of Sudoku puzzles obtained by our parallel and distributed Boolean Gröbner basis program in SageMath in order to show its effect. The reader is referred to [13] for s-rank.

Table 6.1 contains average time (in seconds) of 10 puzzles in the Sudoku book High and UltraHard

[7] for obtaining a solution of a puzzle. These puzzles have a property of $basicsolvable$ introduced in [13]. Each computation is done serially by computer 1 in Table 6.2.

We put snapshots of the computations of the following puzzle by SageMath and Risa/Asir programs. `S50_1` is basic solvable.

	SageMath (Sequential)	Risa/Asir (Sequential)
High	0.64	9.44
UltraHard	0.77	11.28

Table 6.1: The calculating time of a Sudoku puzzle

		9	3	8	4	2		
	8			1			3	
3				5				8
9				2				3
1				7				6
8			4		5			2
4		3				1		7
	2						9	
		8	6	4	1	3		

S50_1

```
sage: S=sudoku_solve(S50_1)
This is solvable.
   1  2  3  4  5  6  7  8  9
1 [6, 7, 9, 3, 8, 4, 2, 5, 1]
2 [2, 8, 5, 7, 1, 6, 4, 3, 9]
3 [3, 1, 4, 2, 5, 9, 7, 6, 8]
4 [9, 4, 6, 1, 2, 8, 5, 7, 3]
5 [1, 5, 2, 9, 7, 3, 8, 4, 6]
6 [8, 3, 7, 4, 6, 5, 9, 1, 2]
7 [4, 6, 3, 5, 9, 2, 1, 8, 7]
8 [5, 2, 1, 8, 3, 7, 6, 9, 4]
9 [7, 9, 8, 6, 4, 1, 3, 2, 5]
S50_1: Comp time  0.671790838242
```

```
[1903] load("./S50_1")$

 _ _ _ _ _ _ _ _ _ _
|6|7|9|3|8|4|2|5|1|
|2|8|5|7|1|6|4|3|9|
|3|1|4|2|5|9|7|6|8|
|9|4|6|1|2|8|5|7|3|
|1|5|2|9|7|3|8|4|6|
|8|3|7|4|6|5|9|1|2|
|4|6|3|5|9|2|1|8|7|
|5|2|1|8|3|7|6|9|4|
|7|9|8|6|4|1|3|2|5|
7.956sec + gc : 0.2046sec(8.165sec)
```

Next, we give some computation data of the s-rank of Sudoku puzzles obtained by our parallel and distributed Boolean Gröbner basis program. The computing environment used for our experiments is summarized below. For the computation of s-ranks, we empirically compute more than 100 Boolean Gröbner bases. These Boolean Gröbner basis computations are independent, so we

	Computer 1 and 2	Computer 3
OS	Ubuntu 14.04 LTS 64bit	Ubuntu 14.04 LTS 64bit
SageMath	7.1	7.1
CPU Clock	Intel(R) Core(TM) i7-3970X 3.50GHz	Intel(R) Core(TM) i7-4960X 3.60GHz
Num of Cores	6	6
Memory	64GB	64GB

Table 6.2: The computing environment

can expect reasonable speed-up by using parallel and distributed Boolean Gröbner basis computation.

The following Table 6.3 contains computation time (in seconds) obtained by three types of computation. The first one, "Sequential" is a serial computing, which means all Boolean Gröbner basis computations are executed sequentially. The second one, "Parallel" means Boolean Gröbner basis computations executed simultaneously by a single computer with multiple processors/cores. We run a Boolean Gröbner bases program introduced in [15] as regards "Sequential" and "Parallel". The third one, "Parallel and Distributed" means "Parallel" computations by three computers in Table 6.2. The first row in Table 6.3 is the average time (in seconds) of 10 puzzles in the Sudoku book UltraHard [7] for obtaining s-rank of a puzzle. Example A and Example B contains the data for two puzzles among them.

	Sequential	Parallel	Parallel and Distributed
Average	74.13	15.55	6.49
Example A	184.9	37.32	14.39
Example B	64.12	14.04	5.66

Table 6.3: Computation time of Srank (Sec)

2						9		
		5	1				6	
	6			3				4
	1			9				2
		4	5					7
					8		4	
7						4		
	5				6		2	
		1	2	7				3

Example A

2						9	8	
	3				1			6
		9		5				4
			6				3	
		3				1		
	9				4			
9				7		8		
4			5				9	
	6	2						5

Example B

The computation time of Example A and B shows that our parallel and distributed computation is efficient for s-rank. When a given Sudoku puzzle is not basic solvable (see [13]), for computing its s-rank, we need computations of many Boolean Gröbner bases. For such computations, our computing method is practical and useful.

6.3 Hierarchy of Sudoku puzzles

In this section, we propose a new hierarchy for the data reported in [13]. For their data, we also found errors and corrected by our program. We use the same notations given in Example 49. The reader is referred to [13] for an operation Ψ.

In [13], s-ranks of 525 Sudoku puzzles contained in the series of Sudoku books (named High, SuperHigh, Hard, SuperHard and UltraHard) [7] are reported as in the following table.

s-rank	0	1	2	3	4	5	∞
High	84	3	10	7	1	0	0
SuperHigh	58	9	22	12	4	0	0
Hard	39	15	21	17	8	4	0
SuperHard	17	13	32	24	19	1	0
UltraHard	11	15	22	21	21	9	6

S-rank of Sudoku puzzles

In the following example, we show an error of [13].

Example 50 In [13], the following puzzle have a s-rank 3 but it is not correct. Let J be an ideal of polynomials which represent this puzzle.

	6				5	3	4	
3		8		1				5
4			2				6	
5						6		
	9			7			1	
		6						9
	3				1			4
9				6		7		1
	8	7	3				2	

Example of s-rank 2 in [13]

	6	1	9	8	5	3	4	
3		8		1				5
4	5	9	2	3	7	1	6	8
5		3?				6		
	9	3?		7			1	
		6			3			9
6	3	2	7		1			4
9	4	5	8	6	2	7	3	1
1	8	7	3				2	6

$\Psi_0^*(J)$

	6	1	9	8	5	3	4	2
3	2	8		1				5
4	5	9	2	3	7	1	6	8
5	2	(3)				6		7
	9	4		7		2	1	3
		6		2	3	4		9
6	3	2	7		1			4
9	4	5	8	6	2	7	3	1
1	8	7	3				2	6

$\{2\} \in \Psi_1^1(\Psi_0^*(J) + \langle X_{43} + \{3\}\rangle)$

2	6	1	9	8	5	3	4	7
3	7	8	6	1	4	2	9	5
4	5	9	2	3	7	1	6	8
5	2	4	1	9	8	6	7	3
8	9	3	5	7	6	4	1	2
7	1	6	4	2	3	8	5	9
6	3	2	7	5	1	9	8	4
9	4	5	8	6	2	7	3	1
1	8	7	3	4	9	5	2	6

$\Psi_0^*(\Psi_0^*(J)$ $+\langle\{3\}X_{43}\rangle)$

Once we compute $\Psi_0^*(J)$; in other words, we use naked and hidden singles, then we can see that this puzzle have $X_{43} = \{3\}$ or $X_{53} = \{3\}$. If we compute $\Psi_0^*(\Psi^0{*}_0(J)$ $+\langle X_{43} + \{3\}\rangle)$, then this ideal contains a contradiction polynomial $\{2\}$. So, we get $\{6\}X_{43}$ and compute $\Psi_0^*(\Psi_0^*(J)$ $+\langle\{6\}X_{43}\rangle)$. In this puzzle, we have a correct answer by adding only $\{3\}X_{43}$. Therefore s-rank is 1 NOT 2.

We have recomputed 735 Sudoku puzzles which are contained in the series of Sudoku books [7] by our program. The results are as follows.

s-rank	0	1	2	∞
High	84	21	0	0
SuperHigh	58	47	0	0
Hard	40	65	0	0
SuperHard	17	86	2	0
UltraHard	13	90	2	0

S-rank of Sudoku puzzles (revised)

Our new data shows that we cannot categorize Sudoku puzzles in terms of their s-ranks. In order to give a finer hierarchy, we define two numbers 'A' and 'B'. 'A' means the number of figures such that we can obtain s-rank by using the only elements of 'A' figures or more. For example, in example 14, the number of figures we firstly have for each element are 4, 2, 4, 3, 3, 5, 3, 2, 3. After we compute $\Psi_0^*(J)$, the number of figures are 7, 4, 7, 4, 5, 7, 5, 5, 5. We can obtain a contradiction polynomial by computing Gröbner bases for elements 1, 3, and 6 which have 7 figures. Then we can obtain an answer of example 14 by $\Psi_0^*(\Psi_0^*(J)$ $+\langle\{3\}X_{43}\rangle)$. Therefore, example 14 have s-rank 1 and A=6. When A = 0 and we cannot construct $BR_k(J)$, then we up s-rank. 'B' means the number of computations of Boolean Gröbner bases for getting a maximal ideal such that $\Psi_0^B(\Psi_0^*(J)$ $+\langle BR_k(J)\rangle)$. The following table contains an obtained data by our program. In the table, puzzles are ordered from left to right according to our mathematical levels of difficulty.

s-rank	0	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2
A	-	6	6	4	4	4	4	4	2	2	2	2	2	4	4	2
B	-	1	2	1	2	3	4	5	1	2	3	4	5	3	4	2
High	84	1	1	10	4				2	2						
SuperHigh	58	17		22	2	2			3	1						
Hard	40	10		28	5	1		1	13	6			1			
SuperHard	17	4	1	38	11	3			15	13			1	1		1
UltraHard	13	2		29	9	4	2		21	14	6	3			1	1

S-rank of Sudoku puzzles (Proposed)

Chapter 7

Conclusions

There are many sophisticated implementations to compute Boolean Gröbner bases such as [2, 3, 5]. Though any of them deals with only Boolean polynomials over $\mathbb{GF}_2$, we can implement our method on those systems.

We have introduced parallel computation of Boolean Gröbner bases and comprehensive Boolean Gröbner bases of an arbitrary finite power set algebra. It is implemented in the computer algebra system SageMath using the PolyBoRi library. A parallel Boolean Gröbner basis computation program is implemented with "@parallel" decorator. We do not use any sophisticated technique of parallel programming. Nevertheless, our method is sufficiently effective as we saw in chapter 5. We have also introduced the distributed computation of Boolean Gröbner bases. Data of our experiments presented in chapter 6 show that our programs work properly and effectively. Our parallel and distributed program in SageMath(PolyBoRi) can obtain s-rank within 10 seconds. When a given Sudoku puzzle is not basic solvable, for computing its s-rank we need computations of many Boolean Gröbner bases. For such a computation, parallel computation is efficient.

For only solving combinatorial problems such as Sudoku puzzles, symbolic computation of Boolean Gröbner bases is too heavy. In fact, a Sudoku puzzle can be formulated in a Boolean polynomial ring of $\mathbb{GF}_2$ using 729 variables. This approach is hired for Sudoku solvers by SAT. They can solve any Sudoku puzzle in a second, while symbolic computation for such a formulation is too heavy even for PolyBoRi. However, this formulation cannot decide the level of difficulty of a Sudoku puzzle. Our approach by symbolic computation is an ideal tool for deciding the s-rank of a Sudoku puzzle.

As is described in the introduction, our parallel Boolean Gröbner basis computation does not have an enough effect on just solving one Sudoku puzzle. For solving a Sudoku puzzle which has a size 16×16, however, we think parallel and distributed Boolean Gröbner basis computations are useful. For solving a normal Sudoku puzzle of the size 9×9, we need about 800 Boolean polynomials with 81 variables, whereas for solving a Sudoku puzzle of the size 16×16, we need about 3700 Boolean polynomials with 196 variables.

We can also parallelize the computation of a comprehensive Boolean Gröbner basis by using the same parallel computation, we have implemented it and show its efficiency. A comprehensive Boolean Gröbner basis is an ideal tool for obtaining an elimination of an ideal of a Boolean polynomial. Computation of such an elimination is very important for many types of combinatorial problems. We expect that our parallel and distributed computation method of Boolean Gröbner bases also contributes to speed-up solving those problems.

We also have some problems besides Sudoku puzzles for which our BGB algorithm is superior to standard algorithms for our future work.

Bibliography

[1] Arnold, E., Lucas, S. and Taalman, L. (2010). Gröbner basis representations of Sudoku, The College Mathematics Journal, 41(2), pp.101-112.

[2] Bosma, W., Cannon, J., and Playoust, C. (1997). The Magma Algebra System I: The user language. Journal of Symbolic Computation, 24 (3-4), pp.235-265. http://magma.maths.usyd.edu.au/magma/.

[3] Brickenstein, M., Dreyer, A. (2009). A framework for Gröbner basis computations with Boolean polynomials. Journal of Symbolic Computation, 44 (9), pp.1326-1345. PolyBoRi Polynomials over Boolean Rings. http://polybori.sourceforge.net/.

[4] Buchberger, B. (1965). Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal. Doctoral Dissertation Math. Inst.University of Innsbruck, Austria.

[5] Decker, W., Greuel, G.-M., Pfister, G., Schönemann, H. (2010). Singular 3-1-2–A computer algebra system for polynomial computations. http://www.singular.unikl.de/.

[6] Gago-Vargas, J., Hartillo, I., Martín-Morales, J., Ucha, J.M. (2006). Sudokus and Grobner bases: not only a divertimento. Computer algebra in scientific computing (CASC 2006), Springer LNCS 4194, pp.155-165.

[7] Gohnai, K., and Cross Word editorial desk (2008). Number Placement Puzzles(Basic, Middle, High, SuperHigh, Hard, SuperHard, UltraHard), (In Japanese) Kosaido Publishing Co.

[8] Kapur, D. (1995). An Approach for Solving Systems of Parametric Polynomial Equations. Principles and Practices of Constraint Programming. (eds. Saraswat and Van Hentenryck), MIT Press, pp.217-244.

[9] Manubens, M. and Montes, A. (2006). Improving DISPGB algorithm using the discriminant ideal. Journal of Symbolic Computation, 41, pp.1245-1263.

[10] Montes, A. (2002). A new algorithm for discussing Gröbner bases with parameters. Journal of Symbolic Computation, 33(2), pp.183-208.

[11] Inoue, S. (2009). On the Computation of Comprehensive Boolean Gröbner Bases. Proceedings of CASC2009, Springer LNCS 5743, pp.130-141.

[12] Inoue, S. (2009). BGSet Boolean Groebner bases for Sets.
http://www.mi.kagu.tus.ac.jp/˜inoue/BGSet/.

[13] Inoue, S. and Sato, Y. (2014). A Mathematical Hierarchy of Sudoku Puzzles and its Computation by Boolean Gröbner Bases. Proceedings of 12th International Conference, AISC2014, Springer LNCS 8884, pp.88-98.

[14] Nagai, A. and Inoue, S. (2014). An Implementation Method of Boolean Gröbner Bases and Comprehensive Boolean Gröbner Bases on General Computer Algebra Systems. Proceedings of ICMS2014, Springer LNCS 8592, pp.531-536.

[15] Nagai, A. and Sato, Y. (2015). An efficient implementation of Boolean Gröbner Bases of a power set algebra. Proceedings of ATCM2015, Mathematics & Technology, LLC, pp.326-335.

[16] Nagai, A. and Sato, Y. (2016). Parallel and Distributed Boolean Gröbner Bases Computation in SageMath. Proceedings of ATCM2016, Mathematics & Technology, LLC, pp.269-278.

[17] Noro, M. et al. (2009). A Computer Algebra System Risa/Asir.
http://www.math.kobe-u.ac.jp/Asir/asir.html.

[18] Rudeanu, S. (1974). Boolean functions and equations. North-Holland, Amsterdam.

[19] Sakai,K. and Sato, Y. (1988). Boolean Gröbner bases. ICOT Technical Momorandum 488, http://www.icot.or.jp/ARCHIVE/Museum/TRTM/tm-list-E.html

[20] Sakai,K. and Sato, Y. and Menju, S. (1991). Boolean Gröbner bases(revised). ICOT Technical Report 613, http://www.icot.or.jp/ARCHIVE/Museum/TRTM/tr-list-E.html

[21] Sato,Y. (1996). Set Constraint Solvers(Prolog Version).
http://www.jipdec.or.jp/archives/icot/ARCHIVE/Museum/FUNDING/funding-95-E.html

[22] Sato,Y. (1998). Set Constraint Solvers(KLIC Version).
http://www.jipdec.or.jp/archives/icot/ARCHIVE/Museum/FUNDING/funding-98-E.html

[23] Sato,Y. (1998). A new type of canonical Gröbner bases in polynomial rings over Von Neumann regular rings. Proceedings of ISSAC 1998, ACM Press, pp.317-32.

[24] Sato, Y. and Suzuki, A. (1999). Parallel Computation of Boolean Gröbner Bases. Electronic Proceedings of ATCM1999.
http://epatcm.any2any.us/10thAnniversaryCD/EP/1999/contributed_papers.html

[25] Sato, Y. and Inoue, S. (2005). On the Construction of Comprehensive Boolean Gröbner Bases. Proceedings of the Seventh Asian Symposium on Computer Mathematics(ASCM 2005), pp.145-148.

[26] Sato, Y., Nagai, A. and Inoue, S. (2008). On the computation of elimination ideals of Boolean polynomial rings. Springer LNCS 5081, pp.334-348.

[27] Sato, Y. et al. (2011). Boolean Gröbner bases. Journal of Symbolic Computation, 46, pp.622-632.

[28] Suzuki, A. and Sato, Y. (2003). An Alternative approach to Comprehensive Gröbner Bases. Journal of Symbolic Computation, 36(3-4), pp.649-667.

[29] Suzuki, A. and Sato, Y. (2006). A Simple Algorithm to Compute Comprehensive Gröbner Bases Using Gröbner Bases. International Symposium on Symbolic and Algebraic Computation(ISSAC 2006), Proceedings, pp.326-331.

[30] Weispfenning, V. (1989). Gröbner Bases in polynomial ideals over commmutative regular rings. EUROCAL'87, Springer LNCS 378, pp.336-347.

[31] Weispfenning, V. (1992). Comorehensive Gröbner Bases. Journal of Symbolic Computation, 14, pp.1-29.