数理解析研究所講究録
918巻 1995年 277-291

Gröbner Bases for Set Constraints

Yosuke Sato
Dept.of Computer Science Ritsumeikan University
1916 Nojicho,Kusatu,Siga,525-77, Japan
E-mail: ysato@theory.cs.ritsumei.ac.jp

Abstract

This paper proposes the application of Gröbner bases to solve set constraints given in terms of symbols such as $\cap, \cup, \subseteq, \in$ and $\notin$. Set constraints can be represented in the form of polynomial equations of a certain Boolean ring, hence we can apply Boolean Gröbner bases we introduced in order to handle polynomial ideals of Boolean polynomial rings. In this paper we study Boolean Gröbner bases in more detail and show they have several nice properties which do not necessarily hold for standard Gröbner bases. Using these properties we describe how we can apply Boolean Gröbner bases to solve set constraints.

1 Introduction

In constraint programming, there are often applications in which we want to solve constraints written in terms of membership or inclusion of sets such as $\in$ and $\subseteq$. Since a family of sets is naturally interpreted as a Boolean ring, many of such constraints, called set constraints in this paper, can be represented by polynomial equations over Boolean rings of sets. For example, the constraints $a \in X, b \notin Y$ and $X \subseteq Y$, where a, b are constant symbols of elements and X, Y are variables for sets, are represented by the equations $\{a\}X = \{a\}$, $\{b\}Y = 0$ and $XY = X$ respectively. Hence there arises an interest if we can apply Gröbner bases to solve them.

Gröbner bases introduced in [Buchberger 65] are extremely useful tools to decide many problems of polynomial ideals. When a coefficient domain is not a field, however, it generally is not so simple to define or calculate Gröbner bases since we can not induce a reduction from a polynomial straightforwardly. General framework of constructing Gröbner bases of polynomial rings over commutative Noetherian rings with some conditions about computability were first introduced in [Trinks 78] and [Zacharias 78]. It, however, is not very efficient since the reductions are not defined so simple that calculation of Gröbner bases is very heavy. For more limited coefficient domain, [Weispfenning 89] introduced Gröbner bases for polynomial rings over commutative regular rings based on special reductions. The same notion of Gröbner bases for Boolean polynomial rings was independently introduced by us and called Boolean Gröbner bases in [Sakai 88], and more detailed study was given in [Sakai 90].

In this paper, we prove two nice properties of Boolean Gröbner bases. One is about the extendability of special solutions presented in Theorem 2.5. When we use an admissible total order $>$ of power products of variables $\bar{X} = X_1, X_2, \ldots, X_n$ and $\bar{Y} = Y_1, Y_2, \ldots, Y_m$ such that $Y_i > X_1^{s_1} X_2^{s_2} \cdots X_n^{s_n}$ for each variable Y_i and power product $X_1^{s_1} X_2^{s_2} \cdots X_n^{s_n}$, Boolean Gröbner bases in general have the form $\{g_1(\bar{X}, \bar{Y}), g_2(\bar{X}, \bar{Y}), \ldots, g_l(\bar{X}, \bar{Y}), h_1(\bar{X}),$

$h_2(\bar{X}), \ldots, h_k(\bar{X})\}$. Theorem 2.5 ensures us that we can extend each solution of the equations $\{h_1(\bar{X}) = 0, h_2(\bar{X}) = 0, \ldots, h_k(\bar{X}) = 0\}$ to a solution of the whole equations $\{g_1(\bar{X}, \bar{Y}) = 0, g_2(\bar{X}, \bar{Y}) = 0, \ldots, g_l(\bar{X}, \bar{Y}) = 0, h_1(\bar{X}) = 0, h_2(\bar{X}) = 0, \ldots, h_k(\bar{X}) = 0\}$. The another one presented in Theorem 2.6 shows the existence of parametric Boolean Gröbner bases. For a given set of polynomials $\{f_1(\bar{X}, \bar{Y}), f_2(\bar{X}, \bar{Y}), \ldots, f_k(\bar{X}, \bar{Y})\}$, we can construct a parametric Boolean Gröbner base $G(\bar{X}) = \{g_1(\bar{X}, \bar{Y}), g_2(\bar{X}, \bar{Y}), \ldots, g_l(\bar{X}, \bar{Y})\}$, that is $G(\bar{a}) = \{g_1(\bar{a}, \bar{Y}), g_2(\bar{a}, \bar{Y}), \ldots, g_l(\bar{a}, \bar{Y})\}$ becomes a Boolean Gröbner basis of $\{f_1(\bar{a}, \bar{Y}), f_2(\bar{a}, \bar{Y}), \ldots, f_k(\bar{a}, \bar{Y})\}$ for each instantiation $\bar{a}$ for the variables $\bar{X}$.
These properties play especially important rolls in our application of Boolean Gröbner bases to solve set constraints.

In section 2, we first give brief review of Boolean Gröbner bases together with several classical results of polynomial ideals of Boolean rings, then we show our main results concerning Boolean Gröbner bases. In section 3, we describe several methods to solve set constraints using Boolean Gröbner bases. In section 4, we give some examples of our methods from our implementation. We also compare our methods with other solvers of Boolean equations in the last section.

2 Boolean Gröbner bases

A *Boolean ring* B is a commutative ring with identity such that every element of B is idempotent, i.e.

$$a^2 = a \quad \text{for all } a \in B.$$

It has the following important property:

$$a + a = 0 \quad \text{for all } a \in B.$$

In this section we fix such a computable Boolean ring B, and describe our Gröbner bases method to solve polynomial equations over B. Since each element of a Boolean ring is idempotent, a quotient ring $B[X_1, X_2, \ldots, X_n]/I$ is more convenient to work on, rather than a polynomial ring $B[X_1, X_2, \ldots, X_n]$ itself, where I is the ideal generated by the set of polynomials $\{X_1^2 + X_1, X_2^2 + X_2, \ldots, X_n^2 + X_n\}$. This quotient ring is called a *Boolean polynomial ring* and denoted by $B(X_1, X_2, \ldots, X_n)$. We also call its element a *Boolean polynomial*. The important property of this quotient ring is that it also becomes a Boolean ring.
A *power product* of variables $X_1, X_2, \ldots, X_n$ is a term $X_1^{s_1} X_2^{s_2} \cdots X_n^{s_n}$ for some non-negative integers $s_1, s_2, \ldots, s_n$. When every s_i is 0 it is denoted by 1. The set of power products naturally forms a commutative monoid. We express elements of B by lowercase letters a, b, c, ..., power products by lowercase Greek letters α, β, γ, ... (possibly with suffix). A power product $X_1^{s_1} X_2^{s_2} \cdots X_n^{s_n}$ is called a *Boolean power product* if $s_i \leq 1$ for each i. Note that each equivalent class of the quotient ring $B(X_1, X_2, \ldots, X_n)$ is uniquely represented by a form $\sum_{i=1}^{l} a_i \alpha_i$, where $\alpha_1, \ldots, \alpha_l$ are different Boolean power products. In this section we regard Boolean polynomials as such representation forms. Using this representation the Boolean polynomial ring becomes computable using the rewriting rules $\{X_1^2 \to X_1, X_2^2 \to X_2, \ldots, X_n^2 \to X_n\}$.

Let us first present the following classical result of Boolean polynomial rings. It enables us to deal with semantic properties of equations concerning their solutions by handling ideals only syntactically.

Theorem 2.1
Let I be an finitely generated ideal in a Boolean polynomial ring $B(X_1, X_2, \ldots, X_n)$. Any n-tuple $\bar{a} = a_1, a_2, \ldots, a_n$ of elements of B is called a solution of I if $f(\bar{a}) = 0$ for every $f \in I$. We say a Boolean polynomial $h(\bar{X})$ is *valid* under I if $h(\bar{a}) = 0$ for any solution $\bar{a}$ of I. Then we have the following properties.

(1) A finitely generated ideal I has a solution if and only if there is not any non-zero element of B in I.

(2) When I has a solution, $h(\bar{X})$ is valid under I if and only if $h(\bar{X}) \in I$, for each Boolean polynomial $h(\bar{X})$. □

A total order $>$ on the commutative monoid of power products is called *admissible* if it satisfies the following properties.

1. If $\alpha > \beta$, then $\alpha\gamma > \beta\gamma$ for any γ.
2. $\alpha > 1$, for any power product $\alpha(\neq 1)$.

Let us fix such a total admissible order $>$. Note the trivial fact that the restriction of $>$ on the set of Boolean power products is also a total order. For a Boolean polynomial $f = \sum_{i=1}^{l} a_i\alpha_i$, the greatest Boolean power product among $\alpha_1, \ldots, \alpha_l$ is called the *leading Boolean power product* of f(denoted by $lpp(f)$), its coefficient is called the *leading coefficient*(denoted by $lc(f)$). The rest part of f is also denoted by $res(f)$. The notation $a\alpha \triangleright h$ denotes a Boolean polynomial but also indicates $lc(a\alpha \triangleright h) = a$, $lpp(a\alpha \triangleright h) = \alpha$ and $res(a\alpha \triangleright h) = h$. A Boolean polynomial f is called a *rule* if $lc(f)res(f) = res(f)$.

For a rule $f = a\alpha \triangleright h$, we define a reduction $\Rightarrow_f$ on the set of Boolean polynomials. It reduces a Boolean polynomial $b\alpha\gamma + g$ such that $ab \neq 0$ as follows:

$$b\alpha\gamma + g \Rightarrow_f (1 + a)b\alpha\gamma + b\gamma h + g$$

For a set F of rules and Boolean polynomials h, h', we say h is *reduced* to h' by F (denoted $h \Rightarrow_F h'$) if $h \Rightarrow_f h'$ for some $f \in F$. The transitive reflexive closure and symmetric transitive reflexive closure of $\Rightarrow_F$ are denoted by $\overset{*}{\Rightarrow}_F$ and $\overset{*}{\Leftrightarrow}_F$ respectively. For any finite set F of rules, we can show the reduction $\Rightarrow_F$ has a termination property, i.e. there is no infinite reduction sequence of Boolean polynomials such that $f_0 \Rightarrow_F f_1 \Rightarrow_F f_2 \Rightarrow_F \cdots$. We abuse the notation $h\downarrow_F$ to denote one of Boolean polynomials h' such that $h \overset{*}{\Rightarrow}_F h'$ and h' is not reducible by $\Rightarrow_F$. Rules are induced because of the following reason. Let F be an arbitrary set of rules, then for each Boolean polynomial f and g we have $f + g \in I$ if and only if $f \overset{*}{\Leftrightarrow}_F g$, where I is the ideal generated by F. This property does not generally holds unless F is a set of rules.

Definition 2.1
Let I be a finitely generated ideal of $B(X_1, X_2, \ldots, X_n)$. A *Boolean Gröbner basis* of I is a finite set G of rules satisfying the following.

(BG 1) G generates I.

(BG 2) $g + g' \in I$ if and only if there is a Boolean polynomial h such that $g \overset{*}{\Rightarrow}_G h$ and $g' \overset{*}{\Rightarrow}_G h$. In particular, $g \in I$ if and only if $g \overset{*}{\Rightarrow}_G 0$.

In addition if it has the following two properties, it is called a *normal* Boolean Gröbner basis.

(BG 3) Each $g \in G$ is not reducible by $\Rightarrow_{g'}$ for any $g' \in G$ distinct from g.

(BG 4) The leading Boolean power product of each Boolean polynomial of G is distinct each other.

In the definition of standard Gröbner bases of polynomial rings over fields, the property (BG 4) is not included. It is a direct conclusion from the property (BG 3). We require it in order to have the following property (P 2).
(Normal) Boolean Gröbner bases have the following important properties.

(P 1) The constraint given by F, that is a set of equations $\{f = 0 | f \in F\}$, is unsatisfiable if and only if the Boolean Gröbner bases of (the ideal generated by) F includes a non-zero constant element of B.

(P 2) For any finitely generated ideal, there uniquely(w.r.t. $>$) exists its normal Boolean Gröbner basis.

The property (P 2) enables us to consider the normal Boolean Gröbner basis of F as a canonical solution of the constraint given by F.

In order to calculate Boolean Gröbner bases, we need to define several notations.
For a pair of rules f and g, its *critical polynomial*(denoted $\mathrm{cp}(f, g)$) is the following Boolean polynomial:

$$lc(g)\frac{lpp(g)}{\mathrm{GCD}(lpp(f), lpp(g))}f + lc(f)\frac{lpp(f)}{\mathrm{GCD}(lpp(f), lpp(g))}g$$

where $\mathrm{GCD}(lpp(f), lpp(g))$ denotes the greatest common divisor of $lpp(f)$ and $lpp(g)$.
For a rule f, its *variable critical polynomial* is the following Boolean polynomial:

$$(1 + X)f$$

where X is a variable included in $lpp(f)$. The set of variable critical polynomials of h is denoted by $\mathrm{vcp}(h)$.

Critical polynomials are generally called S-polynomials in calculation of standard Gröbner bases. Variable critical polynomials are induced since we are working on quotient rings. In calculation of a Boolean Gröbner basis, a variable critical polynomial $(1 + X)f$ plays the same roll as the S-polynomial between f and $X^2 + X$.

We can describe Boolean Gröbner bases as follows. Hence, we can construct them using these polynomials by a similar completion method of Buchberger's algorithm to construct standard Gröbner bases.

Theorem 2.2
A finite set G of rules is a Boolean Gröbner basis if and only if every critical polynomial and variable critical polynomial constructed by rules of G is reduced to 0 by $\overset{*}{\Rightarrow}_G$. □

Throughout the rest of the paper whenever we use Boolean Gröbner bases, they are always

supposed to be normal. Let us present an important classical result before proving several nice properties of Boolean Gröbner bases.

Let $B(\bar{X},\bar{Y})$ be a Boolean polynomial ring with variables $\bar{X} = X_1, X_2, \ldots, X_n$ and $\bar{Y} = Y_1, Y_2, \ldots, Y_m$. For each ideal I of $B(\bar{X},\bar{Y})$, $I \cap B(\bar{X})$ (denoted by $I_{B(X)}$) forms also an ideal of the Boolean polynomial ring $B(\bar{X})$. Any n-tuple $\bar{a} = a_1, a_2, \ldots, a_n$ of elements of B is called an $(\bar{X})$-*solution projection* of I if there exists an m-tuple $\bar{b} = b_1, b_2, \ldots, b_m$ of elements of B such that $\bar{a}, \bar{b}$ is a solution of I. Clearly any $(\bar{X})$-solution projection of I is a solution of $I_{B(X)}$. The converse does not generally hold for arbitrary polynomial rings. But in Boolean polynomial rings it fortunately follows from the following classical result.

Lemma 2.3
Let I be a finitely generated ideal of the Boolean polynomial ring $B(\bar{X},\bar{Y})$. Suppose $I_{B(X)}$ has a solution, then each solution can be extended to a whole solution of I.

proof: We give a brief sketch. We assume there is only one variable $\bar{Y} = Y$. General case easily follows inductively. Recall that any finitely generated ideal in a Boolean ring is a principal ideal. Hence there is a Boolean polynomial f such that $I = (f)$. Note that we can express $f = Yg(\bar{X}) + h(\bar{X})$. Then we can show that $I_{B(X)} = ((g(\bar{X}) + 1)h(\bar{X}))$. Let $\bar{a}$ be a solution of $I_{B(X)}$, i.e. $(g(\bar{a}) + 1)h(\bar{a}) = 0$. Then the equation $Yg(\bar{a}) + h(\bar{a}) = 0$ has a solution $c(g(\bar{a}) + 1) + h(\bar{a})$ where c is any element of B.

□

Let us consider the following two problems.

Problem 1. How can we find solution projections ?

Problem 2. How can we extend a given solution projection to a whole solution ?

We first consider the first problem.
The next lemma is a standard technique of Gröbner bases to calculate $I_{B(X)}$ which immediately follows from the definition of Boolean Gröbner bases.

Lemma 2.4
Let $>$ be an admissible total order on the set of power products of variables $Y_1, Y_2, \ldots, Y_m, X_1, X_2, \ldots, X_n$ such that $Y_i > X_1^{s_1} X_2^{s_2} \cdots X_n^{s_n}$ for any Y_i and power product $X_1^{s_1} X_2^{s_2} \cdots X_n^{s_n}$. Under this order let G be the Boolean Gröbner basis of a finitely generated ideal I of $B(\bar{X},\bar{Y})$. Then $G \cap B(\bar{X})$ is a Boolean Gröbner basis of $I_{B(X)}$. □

By the above Lemma 2.3-2.4, we can immediately conclude the following property of Boolean Gröbner bases.

Theorem 2.5
Using the same notations as in the above lemmas, $\bar{a}$ is an $(\bar{X})$-solution projection of I if and only if $g(\bar{a}) = 0$ for every Boolean polynomial $g(\bar{X}) \in G \cap B(\bar{X})$. □

This theorem provides us a nice own property of Boolean Gröbner bases in the following sense. In a general polynomial ring, after we find a candidate $\bar{a}$ of $(\bar{X})$-solution projections of I by solving the equations $\{g(\bar{X}) = 0 | g(\bar{X}) \in G \cap B(\bar{X})\}$, we have to check if there exists $\bar{b}$ such that $\bar{a}, \bar{b}$ is a solution of I. This check, in general, depends on each $\bar{a}$. But, in a Boolean polynomial ring, it does not depend. In fact, Theorem 2.5 guarantees us that this check is not even necessary.

Let us now consider the second problem. That is we want to find how we can extend a $(\bar{X})$-solution projection $\bar{a}$ of I to a whole solution $\bar{a},\bar{b}$ of I. Since $\bar{b}$ is a solution of the instantiated ideal $I(\bar{a}) = \{p(\bar{a},\bar{Y})|p(\bar{X},\bar{Y}) \in I\}$ of $B(\bar{Y})$ and it is generated by $G(\bar{a}) = \{p(\bar{a},\bar{Y})|p(\bar{X},\bar{Y}) \in G\}$, we can solve it by calculating its Boolean Gröbner basis in $B(\bar{Y})$. This calculation, however, strongly depends on the values $\bar{a}$.
Fortunately there is an another approach which allows us extend $(\bar{X})$-solution projections simultaneously.

Let I be an ideal of a Boolean polynomial ring $B(\bar{X},\bar{Y})$. For each n-tuple $\bar{a}$ of elements of B, let $I(\bar{a}) = \{p(\bar{a},\bar{Y})|p(\bar{X},\bar{Y}) \in I\}$. Then it is easy to check that $I(\bar{a})$ is an ideal of a Boolean polynomial ring $B(\bar{Y})$.
Since a Boolean polynomial ring is also a Boolean ring, $B(\bar{X},\bar{Y})$ can be regarded as a Boolean polynomial ring $(B(\bar{X}))(\bar{Y})$ over a Boolean ring $B(\bar{X})$ with variables $\bar{Y}$. In this Boolean polynomial ring we can also calculate a Boolean Gröbner basis of I. Let it be denoted by $G(\bar{X})$. For each n-tuple $\bar{a}$ of elements of B, define

$$G(\bar{a}) = \{g(\bar{a},\bar{Y}) \mid g(\bar{X},\bar{Y}) \in G(\bar{X}) \text{ and } g(\bar{a},\bar{Y}) \neq 0\}.$$

We call $G(\bar{X})$ a *parametric Boolean Gröbner basis* because of the following theorem.

Theorem 2.6
For each n-tuple $\bar{a}$ of elements of B,

(1) $G(\bar{a})$ is the Boolean Gröbner basis of the ideal $I(\bar{a})$ of $B(\bar{Y})$.

Moreover for each Boolean polynomial $f(\bar{X},\bar{Y})$, we have

(2) $f(\bar{a},\bar{Y})\downarrow_{G(a)} = (f(\bar{X},\bar{Y})\downarrow_{G(X)})(\bar{a},\bar{Y})$.

proof: Note that a substitution $\bar{X} \leftarrow \bar{a}$ naturally induce a homomorphism from $B(\bar{X})$ into B. Hence we can easily show the following lemmas and corollary. Using them together with Theorem 2.2, at first (1) follows, then (2) follows immediately. □

Lemma 2.7
If $f(\bar{X},\bar{Y}) \Rightarrow_{g(X,Y)} f'(\bar{X},\bar{Y})$ in $(B(\bar{X}))(\bar{Y})$,
then $f(\bar{a},\bar{Y}) = f'(\bar{a},\bar{Y})$ or $f(\bar{a},\bar{Y}) \Rightarrow_{g(a,Y)} f'(\bar{a},\bar{Y})$ in $B(\bar{Y})$ for each $\bar{a}$. □

Corollary 2.8
If $f(\bar{X},\bar{Y}) \stackrel{*}{\Rightarrow}_{G(X)} f'(\bar{X},\bar{Y})$ in $(B(\bar{X}))(\bar{Y})$,
then $f(\bar{a},\bar{Y}) \stackrel{*}{\Rightarrow}_{G(a)} f'(\bar{a},\bar{Y})$ in $B(\bar{Y})$ for each $\bar{a}$. □

Lemma 2.9
If $f(\bar{X},\bar{Y})$ is not reducible by $\Rightarrow_{G(X)}$ in $(B(\bar{X}))(\bar{Y})$,
then $f(\bar{a},\bar{Y})$ is not either reducible by $\Rightarrow_{G(a)}$ in $B(\bar{Y})$ for each $\bar{a}$. □

By (1) of Theorem 2.6, we can give an alternative method to answer the above problems 1 and 2.
In general $G(\bar{X})$ has the following form, where $\alpha_1,\alpha_2,\ldots,\alpha_k$ are power products consisting of only variables $Y_1,Y_2,\ldots,Y_m$.

$$G(\bar{X}) = \{h_1(\bar{X})\alpha_1 \triangleright g_1(\bar{X},\bar{Y}), h_2(\bar{X})\alpha_2 \triangleright g_2(\bar{X},\bar{Y}),\ldots,h_k(\bar{X})\alpha_k \triangleright g_k(\bar{X},\bar{Y}), h(\bar{X})\}$$

Then we have:

1. $\bar{a}$ is an $(\bar{X})$-solution projection of I if and only if $h(\bar{a}) = 0$.

2. If $\bar{a}$ is an $(\bar{X})$-solution projection of I, $G(\bar{a})$ can be regarded as a canonical solution for the variables $\bar{Y}$ of the whole extension of $\bar{a}$, by the property (P 2) of Boolean Gröbner bases. Hence, we can consider $G(\bar{X})$ as a functional which assigns the canonical solution for the variables $\bar{Y}$ of the whole extension for each $(\bar{X})$-solution projection of I.

Note that this approach is better than the previous one for the problem 2 since it provides us simultaneous extensions with parameter $\bar{X}$. For the problem 1, however, it is not complete. We have to check if there exists $\bar{a}$ such that $h(\bar{a}) = 0$.

We conclude this section, by showing the following type of problems concerning validity defined in Theorem 2.1 can be solved also using Boolean Gröbner bases.

Problem 3.
Let $f_1(\bar{X},\bar{Y}), f_2(\bar{X},\bar{Y}),\ldots, f_l(\bar{X},\bar{Y})$ and $f(\bar{X},\bar{Y})$ be Boolean polynomials of a Boolean polynomial ring $B(\bar{X},\bar{Y}) = B(X_1,X_2,\ldots,X_n,Y_1,Y_2,\ldots,Y_m)$. Then our problem is described as follows.
Find an n-tuple $\bar{a}$ of elements of B such that the set of equations $\{f_1(\bar{a},\bar{Y}) = 0, f_2(\bar{a},\bar{Y}) = 0,\ldots, f_l(\bar{a},\bar{Y}) = 0\}$ has a solution and the equation $f(\bar{a},\bar{b}) = 0$ holds for each solution $\bar{b}$ of it.

By theorem 2.1 it is equivalent to finding $\bar{a}$ such that the ideal $I(\bar{a})$ of $B(\bar{Y})$ generated by $\{f_1(\bar{a},\bar{Y}) = 0, f_2(\bar{a},\bar{Y}) = 0,\ldots, f_l(\bar{a},\bar{Y}) = 0\}$ does not include non-zero elements of B and $f(\bar{a},\bar{Y})$ is included in $I(\bar{a})$.
Let $G(\bar{X}) = \{h_1(\bar{X})\alpha_1 \triangleright g_1(\bar{X},\bar{Y}), h_2(\bar{X})\alpha_2 \triangleright g_2(\bar{X},\bar{Y}),\ldots, h_k(\bar{X})\alpha_k \triangleright g_k(\bar{X},\bar{Y}), h(\bar{X})\}$ be the Boolean Gröbner basis of $\{f_1(\bar{X},\bar{Y}), f_2(\bar{X},\bar{Y}),\ldots, f_l(\bar{X},\bar{Y})\}$ in $(B(\bar{X}))(\bar{Y})$ and let $f'(\bar{X},\bar{Y}) = (f(\bar{X},\bar{Y})\downarrow_{G(X)})(\bar{X},\bar{Y})$.
By Theorem 2.6, it is equivalent to finding $\bar{a}$ such that $h(\bar{a}) = 0$ and $f'(\bar{a},\bar{Y}) \equiv 0$. Let $f'(\bar{X},\bar{Y}) = p_1(\bar{X})\beta_1 + p_2(\bar{X})\beta_2 + \cdots + p_k(\bar{X})\beta_k$ be the representation of f' in $(B(\bar{X}))(\bar{Y})$. Then $f'(\bar{a},\bar{Y}) \equiv 0$ is equivalent to that $\bar{a}$ satisfies the equations $p_1(\bar{a}) = 0, p_2(\bar{a}) = 0,\ldots, p_k(\bar{a}) = 0$. Hence, we can solve $\bar{a}$ by calculating the Boolean Gröbner basis of $\{h(\bar{X}), p_1(\bar{X}), p_2(\bar{X}),\ldots, p_k(\bar{X})\}$.

3 Set Constraints

In this section we describe how we can apply Boolean Gröbner bases to solve set constraints. In order to describe set constraints, let us first give a language.

A language of set constraints is a second order language given by the following symbols.

$a, b, c, \ldots$: first-order constant symbols for elements

$x, y, z, \ldots$: first-order variables for elements

$\emptyset$: second-order constant symbols for an empty set

$X, Y, Z, \ldots$: second-order variables for sets

$\in, \notin, \subseteq$: predicate symbols for elements and sets

$\{\cdot\}, \{\cdot,\cdot\}, \{\cdot,\cdot,\cdot\}, \ldots$: function symbols for functions from elements to sets

$\cap, \cup, ^{\sim}$: function symbols for functions from sets to sets
($^{\sim}X$ denotes the complement of X)

$\vee, \wedge, \neg, \rightarrow$: logical symbols

Let us first note that any constraint given by the above language can be represented as the following form.

$$\bigvee_{i=1}^{k} H^i, \quad \text{where } H^i \equiv \bigwedge_{j=1}^{l_i} H^i_j$$

Each H^i_j is either an atomic formula or a negation of an atomic formula.
We will concentrate on solving each component H^i. (The set of solutions of $\bigvee_{i=1}^{k} H^i$ is given as a union of each set of solutions of H^i.)

Let us first consider the following constraint H with only atomic formulas H_j.

$$H \equiv \bigwedge_{j=1}^{l} H_j$$

Suppose H includes first order variables $x_1, x_2, \ldots, x_p$. Let $X_1, X_2, \ldots, X_p$ be some second order variables which do not appear in H. We translate H_j into H'_j by eliminating $x_1, x_2, \ldots, x_p$ using $X_1, X_2, \ldots, X_p$ as follows.
If H_j has a form such as $x_i \in T$ or $x_i \notin T$ for some first-order variable x_i, it is translated into $\{x_i\} \cap T = \{x_i\}$ or $\{x_i\} \cap T = \emptyset$ respectively. After this translation any first-order variable x_i occurs only in a term such as $\{\cdots, x_i, \cdots\}$. Note that any term such as $\{\cdots\}$ can be represented as a union of singletons of first-order variables and a finite set of constant symbols. For example $\{a, x, b, y\}$ is represented as $\{x\} \cup \{y\} \cup \{a, b\}$. Replacing each singleton $\{x_i\}$ by X_i, any term is translated into a term with no first-order variables, and we get H'_j which does not include any first-order variables.
By the construction, the following should be clear.

$$H \Longleftrightarrow \exists X_1 \exists X_2 \cdots \exists X_p (X_1 = \{x_1\} \wedge X_2 = \{x_2\} \wedge \cdots \wedge X_p = \{x_p\} \wedge \bigwedge_{j=1}^{l} H'_j)$$

Let U be the set of all constant symbols $a, b, c, \ldots$. Then the set of all subsets of U naturally forms a Boolean ring by defining $X + Y = (X \cap {}^{\sim}Y) \cup ({}^{\sim}X \cap Y)$ and $XY = X \cap Y$ for each $X, Y \subseteq U$ with an empty set as 0 and a whole set U as 1.
We denote this Boolean ring by B in the rest of this section. Note that the atomic formula $a \in X, a \notin X$ and $X \subseteq Y$ are represented by the equations $\{a\}X = \{a\}, \{a\}X = 0$ and $XY = X$ respectively. Hence, if a constraint in the form of an atomic formula has no first-order variables for elements, it can be represented by a polynomial equation of a Boolean polynomial ring $B(Y_1, Y_2, \ldots, Y_n)$, where $Y_1, Y_2, \ldots, Y_n$ are second-order variables included in the constraint.
Therefore we can translate $\bigwedge_{j=1}^{l} H'_j$ into the form $\bigwedge_{j=1}^{l} f_j = 0$ with some Boolean polynomials $f_1, \ldots, f_l$ in $B(X_1, X_2, \ldots, X_p, Y_1, Y_2, \ldots, Y_n)$ for some second-order variables $Y_1, Y_2, \ldots, Y_n$ and $X_1, X_2, \ldots, X_p$.
Now we have the following representation.

$$H \Longleftrightarrow \exists X_1 \exists X_2 \cdots \exists X_p (X_1 = \{x_1\} \wedge X_2 = \{x_2\} \wedge \cdots \wedge X_p = \{x_p\} \wedge \bigwedge_{j=1}^{l} f_j = 0)$$

We will describe two methods to solve this constraint by using Boolean Gröbner bases. The first one is appropriate when we are interested in the first-order variables $x_1, x_2, \ldots, x_p$, the second one is appropriate when we are interested in the second-order variables $Y_1, Y_2, \ldots, Y_n$.

Solution method 1
Let $>$ be a total admissible order such that $Y_i > X_1^{s_1} X_2^{s_2} \cdots X_p^{s_p}$ for each Y_i and power product $X_1^{s_1} X_2^{s_2} \cdots X_p^{s_p}$. Under this order, calculate the Boolean Gröbner basis G of $\{f_1(\bar{X}, \bar{Y}), f_2(\bar{X}, \bar{Y}), \ldots, f_l(\bar{X}, \bar{Y})\}$. In general it has the following form.

$$G = \{g_1(\bar{X}, \bar{Y}), g_2(\bar{X}, \bar{Y}), \ldots, g_m(\bar{X}, \bar{Y}), h_1(\bar{X}), h_2(\bar{X}), \ldots, h_r(\bar{X})\}$$

By Theorem 2.5, we can conclude that any solution for the first-order variables $x_1, x_2, \ldots, x_p$ of the constraint H is a solution of the equations $\{h_1(\{\bar{x}\}) = 0, h_2(\{\bar{x}\}) = 0, \ldots, h_r(\{\bar{x}\}) = 0\}$, and vice versa.
In order to solve second-order variables $\bar{Y}$, we have to calculate the Boolean Gröbner basis of $\{g_1(\{\bar{a}\}, \bar{Y}), g_2(\{\bar{a}\}, \bar{Y}), \ldots, g_m(\{\bar{a}\}, \bar{Y})\}$ for each solution $\bar{a} = a_1, a_2, \ldots, a_p$ of the above equations. Where $\{x\}$ and $\{a\}$ denote $\{x_1\}, \{x_2\}, \ldots, \{x_p\}$ and $\{a_1\}, \{a_2\}, \ldots, \{a_p\}$ respectively.

Solution method 2
Calculate the parametric Boolean Gröbner basis $G(\bar{X})$ of $\{f_1(\bar{X}, \bar{Y}), f_2(\bar{X}, \bar{Y}), \ldots, f_l(\bar{X}, \bar{Y})\}$. In general it has the following form.

$$G(\bar{X}) = \{g_1(\bar{X}, \bar{Y}), g_2(\bar{X}, \bar{Y}), \ldots, g_m(\bar{X}, \bar{Y}), h(\bar{X})\}$$

By Theorem 2.6 (1), we can conclude the following.
The constraint H has a solution if and only if $h(\{\bar{x}\}) = 0$ has a solution. Furthermore, for each solution $\bar{a}$ of it, $G(\{\bar{a}\})$ can be considered as a canonical solution for the second-order variables $Y_1, Y_2, \ldots, Y_n$.

Remark 1
In case there are no first-order variables in H, these methods are same.

Remark 2
Note that we can also solve first-order variables by solving $h(\{\bar{x}\}) = 0$. In general, however, it is more complex than solving $\{h_1(\{\bar{x}\}) = 0, h_2(\{\bar{x}\}) = 0, \ldots, h_r(\{\bar{x}\}) = 0\}$, since $\{h_1(\bar{X}), h_2(\bar{X}), \ldots, h_r(\bar{X})\}$ is the Boolean Gröbner basis of $\{h(\bar{X})\}$.
One might think that we can get $h_1, h_2, \ldots, h_r$ by calculating the Boolean Gröbner basis of $\{h\}$ after calculating h by the second method. Our experimental data show that calculation of Boolean Gröbner bases of the Boolean polynomial ring $(B(\bar{X}))(\bar{Y})$ is much heavier than calculation of Boolean Gröbner bases of the Boolean polynomial ring $B(\bar{X}, \bar{Y})$. The essential reason is that calculation of Boolean polynomials of $B(\bar{X})$ are generally much heavier than calculation of elements of B.

We next consider a constraint $H \equiv \bigwedge_{j=1}^{l} H_j$ where each H_j can be not only an atomic formula but also a negation of an atomic formula.
If the inside of the negation is either of a form $\in$ or $\notin$, it can be represented by an atomic formula. Otherwise representing the inside as an equation $f = 0$ as before, it is represented by $f \neq 0$. Note that this is equivalent to $\exists z\{z\}f = \{z\}$(i.e. $\exists z z \in f$) for some first-order variable z which does not occur in f.

Therefore by adding some new first-order variables we can translate H into a form without negations.

Remark 3
In the above solutions, if we are not interested in all second-order variables $Y_1, Y_2, \ldots, Y_n$ but only some variables $Y_{i_1}, Y_{i_2}, \ldots, Y_{i_m}$, we can solve the constraint only for these variables using a total admissible order such that any power product consisting of only $Y_{i_1}, Y_{i_2}, \ldots, Y_{i_m}$ is less than any of other variable Y_k.

We conclude this section with a solution method for a special type of set constraints with quantifiers.
Special type of set constraints
Given Boolean polynomials $f_1(\bar{X}, \bar{Y}), f_2(\bar{X}, \bar{Y}), \ldots, f_l(\bar{X}, \bar{Y})$ and $f(\bar{X}, \bar{Y})$ of $B(\bar{X}, \bar{Y})$. Find constant symbols $\bar{a}$ such that the set of equations $\{f_1(\bar{X}, \{\bar{a}\}) = 0, f_2(\bar{X}, \{\bar{a}\}) = 0, \ldots, f_l(\bar{X}, \{\bar{a}\}) = 0\}$ has a solution for the second-order variables $\bar{X}$ and $f(\bar{S}, \{\bar{a}\}) = 0$ holds for each solution $\bar{S}$ of it.

It can be solved by the method given at the end of section 2. Using the same notations there, it is equivalent to finding $\bar{a}$ such that $\bar{a}$ satisfies the equations $h(\{\bar{a}\}) = 0, p_1(\{\bar{a}\}) = 0, p_2(\{\bar{a}\}) = 0, \ldots, p_k(\{\bar{a}\}) = 0$.
Since our domain is not all subsets of B but only singleton sets of B, we can not solve it by simply calculating the Boolean Gröbner basis H of $\{h(\bar{X}), p_1(\bar{X}), p_2(\bar{X}), \ldots, p_k(\bar{X})\}$. We have to solve the equations in the domain of singleton sets of B. In general it is easier to solve $\{q(\{\bar{x}\}) = 0 | q(\bar{X}) \in H\}$ than solving the equations $h(\{\bar{x}\}) = 0, p_1(\{\bar{x}\}) = 0, p_2(\{\bar{x}\}) = 0, \ldots, p_k(\{\bar{x}\}) = 0$.

4 Examples

We give several examples of our implementation of set constraint solvers based on the methods described in section 3.

In the following query "`?- solve_set_element(Cons,SetVar,AtomVar).`", we input a constraint in a form of a list of equations for `Cons`, a list of second-order variables in which we are interested for `SetVar` and a list of first-order variables for `AtomVar`. The symbols `/\` and `\/` represent $\cap$ and $\cup$ respectively. The symbols `a1,a2, ...` in `{ }` represent different constant symbols of elements unless they are included in the list `AtomVar`. Symbols included in `AtomVar` represent first-order variables.

```
?- solve_set_element([
{a1,a9}/\(X2\/ ~X1)= {a2,a5,a7}/\S5,
S1/\(S2/\ ~{a5}\/(S3/\(S4/\ ~(X1\/{a6}))))=(S5/\S6)\/(S4/\S7/\S2)\/S8,
S5/\S4/\S9/\S6/\S2/\(S10\/X3\/{a7})/\({a9}\/X2)=0,S5/\S8/\ ~(X3\/{a5,a8})=0,
S2/\(~{a6,a8})=S4\/S8,S2/\(~{a9})=S11\/S10\/X2\/{a5,a7}\/S7,S11/\S3=0,
S10/\S3=0,S12=S9\/S7,S9/\S7/\S15/\ ~(X1\/X2\/X3\/{a4,a5,a6,a7,a8,a9,a10})=0,
{a4}/\S6={a4},X1/\S9=X1,X3/\S7=X3,S1\/S10\/X1\/X2=(S12\/S13)/\{a4}\/X2\/S10,
(S1\/S12\/S2)/\X1\/X2\/S6=(S10\/S9)\/(S7/\X3)\/S2,
(S5\/S3\/{a4})/\(S2\/S4\/S3\/S6)=(S1\/(X1\/X2/\(S3\/S10)))/\S1\/S14
],[X1,X2,X3],[]).
```

```
contradiction deduced as follows
{a9} = 0
```

In this query, there is no first-order variables. Hence two methods described in section 3 are same. Our program calculated the Boolean Gröbner basis of the first list to solve X1,X2 and X3 and detected it includes the constant element {a9}. Hence it returned the above answer.
Since we got a contradiction by {a9} = 0, in the next query we made a9 a variable in order to get some conditions to make the constraint satisfiable. The symbol C1 denotes the equations in the first list of the above query.

```
?- solve_set_element([C1],[],[a9]).

constraint is satisfiable when
(~{a2,a4})*{a9} = 0
```

Our program proceeds the method 1 when we input no second-order variables. Hence, it returned the answer (~{a2,a4})*{a9} = 0. We can see that it holds if and only if a9 = a2 or a9 = a4.
In the following third and forth queries, we replaced a9 by a2 and a4 respectively, that is C2 (C3) is given from C1 replacing a9 by a2 (a4).

```
?- solve_set_element([C2],[X1,X2,X3],[]).

constraint is always satisfiable
--------------------------------
under the above condition [X1,X2,X3] has the following form
{a5}*X3*X1 = 0
{a5,a7}*X3*X2 = {a5,a7}*X3
{a2}*X3 = 0
{a1,a2}*X1 = {a1}
{a1,a2}*X2 = 0

?- solve_set_element([C3],[X1,X2,X3].[]).

constraint is always satisfiable
--------------------------------
under the above condition [X1,X2,X3] has the following form
{a5}*X3*X1 = 0
{a2,a5,a7}*X3*X2 = {a2,a5,a7}*X3
{a4}*X3 = 0
{a1,a4}*X1 = {a1,a4}
{a1,a4}*X2 = 0
```

In the next query, each list of variables is non-empty. In this case, our program proceed the method 2.

```
?- solve_set_element([C1],[X1,X2,X3],[a9]).

constraint is satisfiable when
(~{a2,a4})*{a9} = 0
--------------------------------
under the above condition [X1,X2,X3] has the following form
({a5}*{a9}+{a5})*X3*X1 = 0
({a2,a5,a7}*{a9}+{a2,a5,a7})*X3*X2 = ({a2,a5,a7}*{a9}+{a2,a5,a7})*X3
({a2,a4}*{a9})*X3 = 0
({a1,a2,a4}*{a9}+{a1})*X1 = {a1,a4}*{a9}+{a1}
({a1,a2,a4}*{a9}+{a1})*X2 = 0
```

Note that the solution of X1,X2,X3 is a general form of the above two solutions, i.e. when a9 = a2 (a9 = a4) it is same as the solution of the third(forth) query.

We next show an example of the solvers discussed at the end of section 3.
In the following query "?- get_gb(Cons,AtomVar,GB).", we input a list of equations for Cons and a list of first-order variables for AtomVar as before. Our program returns its Boolean Gröbner basis into the output variable GB.
In the query "?- make_valid_under_gb(Eq,GB).", we input an equation for Eq and a Boolean Gröbner basis given by the above query. Our program returns conditions of first-order variables such that Eq becomes valid under the constraints given by GB.

Using the same list of equations [C1] as above, let us execute the following query.

```
?- get_gb([C1],[a9],GB).
```

Our program returned a Boolean Gröbner basis into the variable GB. For arbitrary equation, we can ask conditions to make it valid, using this Boolean Gröbner basis.
Let us execute the following query.

```
?- make_valid_under_gb({a9}/\(S1\/S2)=0,GB).

equation is valid when
(~{a2})*{a9} = 0
```

It means {a9}/\(S1\/S2)=0 is valid under the constraints if and only if
(~{a2})*{a9} = 0, i.e. a9 = a2.
We can also ask other questions using the same GB.

```
?- make_valid_under_gb({a9}/\(S11\/S12)={a9},GB).

equation is valid when
(~{a4})*{a9} = 0
```

It means `{a9}/\(S11\/S12)={a9}` is valid under the constraints if and only if `(~{a4})*{a9} = 0`, i.e. `a9 = a4`.

5 Conclusion

Our research of this paper was originally motivated by a desire to install a set constraint solver in CAL that is a constraint logic programming language we have developed in ICOT([Aiba 88]). In constraint logic programming, we have to keep a constraint in a form of its *canonical representation.* The following two properties are indispensable for this representation.

1. Canonical representation is unique, that is if two constraints are equivalent(usually defined as they have same solutions), their canonical representation forms are identical.

2. We can easily check satisfiability of a constraint using its canonical representation form.

In general, simple canonical representation is preferable in order to get simple solutions. It is also preferred that we can handle more problems relating to a constraint besides checking only satisfiability.
In our set constraint solver, we did not want to use any parameter to represent a canonical form in order to make solutions easy to read. Moreover, in order to manage many problems relating to a constraint, we wanted to make its canonical solution have powers to decide problems concerning validity.

Boolean Gröbner bases enable us to have such canonical solution forms. In fact, they have the following further nice property.

> Boolean Gröbner bases provide many types of canonical solution forms according to the employed admissible total orders of power products.

Admissible total orders of power products heavily affect the shapes of Gröbner bases. When we use an admissible total order $>$ of power products of variables $\bar{X} = X_1, X_2, \ldots, X_n$ and $\bar{Y} = Y_1, Y_2, \ldots, Y_m$ such that $Y_i > X_1^{s_1} X_2^{s_2} \cdots X_n^{s_n}$ for each variable Y_i and power product $X_1^{s_1} X_2^{s_2} \cdots X_n^{s_n}$, Gröbner bases generally have the following form:

$$\{g_1(\bar{X}, \bar{Y}), g_2(\bar{X}, \bar{Y}), \ldots, g_l(\bar{X}, \bar{Y}), h_1(\bar{X}), h_2(\bar{X}), \ldots, h_k(\bar{X})\}$$

Hence, if we want to solve equations with priority for the variable $\bar{X}$ over the other variables $\bar{Y}$, that is we are more interested in $\bar{X}$ than $\bar{Y}$, such orders are suitable. Especially, if we want to solve equations with a strict order of variables for priority, a purely lexicographic order is suitable.
On the other hand, if we do not want to give priority to any variable, a total degree order (or a total degree reverse lexicographic order to get complete influence) is suitable.
In our Boolean polynomial rings of sets, Boolean Gröbner bases generally have simple(i.e. easy to read) forms when we use a total degree order. In the first example of our programs in section 4, the power products of the variables in `SetVar` are compared by a total degree order to get simple answers.

Of course there are many other methods to solve Boolean equations. As far as we know, they are essential divided into two types of methods. One is based on Boole's classical

variable elimination method, another is based on Löwenheim's formula. In order to compare our method with them, let us see how the answer of the third example of the section 4 looks like when we solve it by the above two classical methods.

If we use Boole's classical variable elimination method, we can have a canonical solution form with parameters U_1, U_2 and U_3 as follows.

$$X_1 = (\tilde{}\{a_1, a_2\}) * U_1 + \{a_1\}$$
$$X_2 = (\tilde{}\{a_1, a_2\}) * U_2$$
$$X_3 = \{a_5\} * U_1 * U_2 * U_3 + \{a_5, a_7\} * U_2 * U_3 + (\tilde{}\{a_2, a_5, a_7\}) * U_3$$

It is possible to eliminate parameters to get the following canonical solution form, although we can not simply use it to decide problems concerning validity.

$$\{a_1\} \leq X_1 \leq \tilde{}\{a_2\}$$
$$0 \leq X_2 \leq \tilde{}\{a_1, a_2\}$$
$$0 \leq X_3 \leq \{a_5\} * X_1 * X_2 + \{a_5, a_7\} * X_2 + \tilde{}\{a_2, a_5, a_7\}$$

Neither of then seem to look simpler than the solution of our method. By Boole's method, each set of equations is solved with a strict order of variables for priority, that is variables are solved step by step. In the example, X_3 is solved by using the solutions of X_1 and X_2, hence the solution does not look so simple for X_3.

Löwenheim's formula constructs a general solution form of given equations using as many parameters as the variables included in the equations from a special instance of solution of them. (It does not offer a unique canonical solution form since a general solution form strongly depends on an instance of solution we choose. In general, it is not either possible to get a canonical solution form without parameters by simply eliminating its parameters.) By this formula, we can get a general solution with parameters U_1, U_2 and U_3 as follows. We used $X_1 = \{a_1\}, X_2 = 0, X_3 = 0$ as an instance of solutions.

$$X_1 = \{a_7\} * U_3 * U_2 * U_1 + \{a_5, a_7\} * U_3 * U_1 + \tilde{}\{a_1, a_2\} * U_1 + \{a_1\}$$
$$X_2 = \{a_5\} * U_3 * U_2 * U_1 + \tilde{}\{a_1, a_2\} * U_2$$
$$X_3 = \{a_1, a_5\} * U_3 * U_2 * U_1 + \{a_5, a_7\} * U_3 * U_2 + \{a_1\} * U_3 * U_1 + \tilde{}\{a_1, a_2, a_5, a_7\} * U_3$$

This, again, does not seem to look simpler than the solution of our method.

We did not describe our solvers for constraints among first-order variables such as the equation `(~{a2,a4})*{a9} = 0` of the examples of section 4, since solving these equations belong to discrete combinatorial problems and very few mathematical (Boolean) structures seem to be embedded.

References

[Aiba 88] Akira Aiba, Ko Sakai, Yosuke Sato, David J.Hawley, Ryuzo Hasegawa (1988). Constraint Logic Programming Language CAL Proceedings of The International Conference on Fifth Generation Computer, 263 276

[Buchberger 65] Buchberger, B. (1965). Ein Algorithmus zum Auffinden der Basiselemente des Restklassenrings nach einem nulldimensionalen Polynomideal. PhD thesis, Universität Innsbruck.

[Buchberger 85] Buchberger, B. (1985). Gröbner bases: An algorithmic method in polynomial ideal theory, chap 6 in Recent Trends in Multidimentional System Theory, N. K. Bose Ed., Reidel Publ. Comp.

[Martin 79] Martin, U., Nipkow, T. (1990). Boolean Unification - The Story So Far. Unification, ed. C.Kirchner, Academic Press, 437 455.

[Robbiano 85] Robbiano, L. (1985). Term orderings on the polynomial ring, EUROCAL '85, Springer LNCS Vol 204, 513 517.

[Rudeanu 74] Rudeanu, S. (1974). Boolean Functions and Equations, North-Holland

[Sakai 88] Sakai, K., Sato, Y. (1988). Boolean Gröbner bases. Proceeding of LA-Symposium in winter, RIMS, Kyoto Univ., 29 40

[Sakai 90] Sakai, K., Sato, Y., Menju, S. (1990). Boolean Gröbner bases(revised). ICOT Technical Report 613, also submitted for publication.

[Trinks 78] Trinks, W. (1978). Über B.Buchbergers Verfahren, Systeme algebraischer Gleichungen zu lösen. J.Number Theory 10, 475 488.

[Weispfenning] Weispfenning, V. (1989). Gröbner bases in polynomial rings over commutative regular rings, EUROCAL '87, J.H. Davenport Ed., Springer LNCS Vol 378, 336 347.

[Zacharias 78] Zacharias, G. (1978). Generalized Gröbner bases in commutative polynomial rings. Thesis at M.I.T., Dept. Comp. Sci.