Parallel computation of Boolean Gröbner bases

Y. Sato[†] and A. Suzuki[‡]
Department of Computer Science
Ritsumeikan University
[†]ysato@theory.cs.ritsumei.ac.jp
[‡]sakira@theory.cs.ritsumei.ac.jp

Abstract

Boolean rings are isomorphic to direct products of a Galois field $GF(2)$. Using these structures, boolean Gröbner bases can be constructed by a parallel computation algorithm. We implemented this method using a parallel logic programming language KLIC. In this paper, we report on this work.

1 Introduction

In polynomial rings over commutative Von Neumann regular rings, we can construct Gröbner bases using special kinds of monomial reductions([W 89]). This method is implemented as a Gröbner bases computation algorithm in "SET CONSTRAINT SOLVER Version 1.0"([Sa 97, Sb 97]).
Meanwhile any commutative Von Neumann regular ring is known to be isomorphic to a sub-ring of a direct product of fields ([SW 75]), and a Gröbner basis of its polynomial ring can be characterized in terms of a Gröbner basis of a polynomial ring over each field. (See Theorem 4.1 for more detailed description.) With this property, we can also construct a Gröbner basis of a polynomial ring over a commutative Von Neumann regular ring by computing a Gröbner basis in a polynomial ring over each field independently. When the structures of direct products are simple, this method seems more efficient, especially under the environment we can use parallel computations.
Finite boolean rings are among the simplest examples of commutative Von Neumann regular rings. They are isomorphic to finite direct products of a Galois field $GF(2)$. Polynomial rings we are handling in [Sa 97] are essentially polynomial rings over finite boolean rings. So, the above method is expected to be more efficient than the original one.
We implemented this method as a parallel computation algorithm of Gröbner bases in "SET CONSTRAINT SOLVER Version 2.0" ([S 99]).
In this paper, we introduce this parallel computation algorithm, and report

on experimental results we got through the actual parallel computations.
In section 2, we give a short description of the theory of Gröbner bases in polynomial rings over commutative Von Neumann regular rings with minimum results for understanding our work. The reader is referred to [W 89] or [S 98] for more detailed comprehensive description.
In section 3, we give a quick review for our set constraint solver based on Gröbner bases. In section 4, we describe our parallel computation algorithm. In section 5, we give several data we got through our experiments.

2 Gröbner bases

A commutative ring R with identity 1 is called a *Von Neumann regular ring* if it has the following property:

$$\forall a \in R\, \exists b \in R\, a^2b = a.$$

For such b, $a^* = ab$ and $a^{-1} = ab^2$ are uniquely determined and satisfy $aa^* = a$, $aa^{-1} = a^*$ and $(a^*)^2 = a^*$.

Note that every direct product of fields is Von Neumann regular ring. Conversely, any Von Neumann regular ring is known to be isomorphic to a subring of a direct product of fields([SW 75]).

We assume R is a Von Neumann regular ring in this section. Note that any sub-ring of R generated by a finite subset of R becomes a Noetherean ring, although R is not generally a Noetherean ring. Therefore there exists a Gröbner basis for any finitely generated ideal in polynomial rings over R. We can even have an algorithm to construct a Gröbner basis of the ideal generated by a given finite set of polynomials. This algorithm is essentially same as Buchberger's algorithm except that we have to define special monomial reductions in our polynomial rings.

In the following unless mentioned, Greek letters α, β, γ are used for terms, Roman letters a, b, c for elements of R and f, g, h for polynomials over R. Throughout this section we work in a polynomial ring over R and assume that some total admissible order on the set of terms is given. The largest term of f is denoted by $lt(f)$ and its coefficient by $lc(f)$. The largest monomial of f i.e. $lc(f)lt(f)$ is denoted by $lm(f)$, the rest of f i.e. $f - lm(f)$ is denoted by $rm(f)$.
Definition 2.1
For a polynomial $f = a\alpha + g$ with $lm(f) = a\alpha$, a monomial reduction $\rightarrow_f$ is defined as follows:

$$b\alpha\beta + h \rightarrow_f b\alpha\beta + h - ba^{-1}\beta(a\alpha + g)$$

where $ab \neq 0$ and $b\alpha\beta$ need not be the largest monomial of $b\alpha\beta + h$.
A monomial reduction $\rightarrow_F$ by a set F of polynomials is also defined naturally.

Using this monomial reduction, we can construct a Gröbner basis of the ideal generated by a given finite set of polynomials. Though the algorithm is almost same as the Buchberger's algorithm for polynomial rings over fields, we have to take account of the following observation.

In a polynomial ring over a field, the equivalence relation $\overset{*}{\leftrightarrow}_F$ induced by the monomial reduction $\rightarrow_F$ for a set F of polynomials coincides with the equivalence relation induced by the ideal (F) generated by F. In our polynomial ring, however, these two equivalence relations are in general not equal.

Definition 2.2

A polynomial f is called *boolean closed* if $(lc(f))^* f = f$. $(lc(f))^* f$ is called a *boolean closure* of f and denoted by $bc(f)$. Note that the boolean closure of any polynomial is boolean closed.

Theorem 2.1

Let F be a set of boolean closed polynomials. Then the equivalence relation $\overset{*}{\leftrightarrow}_F$ coincides with the equivalence relation induced by the ideal (F).

Since $lm(f) = lm(bc(f))$, we can construct a set of boolean closed polynomials H from any given set of polynomials F such that $(F) = (H)$. H is also called a *boolean closure* of F. Though such H is not determined uniquely, we abuse the notation $bc(F)$ to denote one of such H.

Using our monomial reductions, Gröbner bases are defined as follows.

Definition 2.3

A finite set G of polynomials is called a Gröbner basis, if it satisfies the following two properties.

· $f \overset{*}{\leftrightarrow}_G g$ iff $f - g \in (G)$ for each polynomial f and g.

· $\rightarrow_G$ has a Church Rosser property,
  i.e. for each polynomial f and g, $f \overset{*}{\leftrightarrow}_G g$ iff there exists
  a polynomial h such that $f \overset{*}{\rightarrow}_G h$ and $g \overset{*}{\rightarrow}_G h$.

Definition 2.4

For each pair of polynomials $f = a\alpha\gamma + f'$ and $g = b\beta\gamma + g'$, where $lm(f) = a\alpha\gamma$, $lm(g) = b\beta\gamma$ and $GCD(\alpha, \beta) = 1$, the polynomial $b\beta f - a\alpha g = b\beta f' - a\alpha g'$ is called the S-polynomial of f and g and denoted by $SP(f, g)$.

We can also characterize Gröbner bases in terms of S-polynomials as in polynomial rings over fields.

Theorem 2.2

Let G be a finite set of boolean closed polynomials. Then

G is a Gröbner basis iff $SP(f, g) \overset{*}{\rightarrow}_G 0$ for any pair f and g of polynomials in G.

It is crucial that each polynomial of G is boolean closed. The theorem does not hold otherwise. This theorem enables us to construct a Gröbner basis G

for a given finite set F of polynomials such that $(G) = (F)$.

Algorithm

```
Let $F_0 = bc(F)$.
    while there exist $f, g \in F_i$ such that $SP(f,g) \downarrow_{F_i} \neq 0$
        $F_{i+1} = bc(F_i \cup \{SP(f,g) \downarrow_{F_i}\})$
    otherwise $G = F_i$
```

Reduced Gröbner bases are defined naturally and can be constructed from any Gröbner bases immediately. We have the following property.

Theorem 2.3

Let G be a reduced Gröbner basis; then any element of G is boolean closed.

In polynomial rings over fields, reduced Gröbner bases serve us as canonical forms of Gröbner bases, however we have to be careful in our polynomial rings.

Definition 2.5

A polynomial f is called monic if it satisfies $lc(f) = (lc(f))^*$.

Definition 2.6

A reduced Gröbner basis G is called a *stratified* Gröbner basis, when it satisfies the following two properties.

· Ever element of G is monic.

· $lt(f) \neq lt(g)$ for any distinct element f and g of G.

Theorem 2.4

A stratified Gröbner basis is determined uniquely. That is two stratified Gröbner bases G and G' such that $(G) = (G')$ must be identical.

Let us conclude this section with showing that there exist another types of canonical forms of Gröbner bases.

Definition 2.7

A Gröbner basis G each element of which is monic and boolean closed is called an *optimal* Gröbner basis if it has the following three properties.

(1) G is a minimal Gröbner basis, i.e. $G - \{f\}$ is not a
 Gröbner basis of the ideal (G) for any element f of G.

(2) For any element f of G, $rm(f)$ is not reducible by $\rightarrow_G$.

(3) For each element f and g of G, if $lt(f)|lt(g)$ i.e.
 $lt(f)$ divides $lt(g)$,then $lc(g)lc(f) = lc(f)$.

Theorem 2.5

Optimal Gröbner basis is determined uniquely, that is optimal Gröbner bases G and G' such that $(G) = (G')$ must be identical.

Note that once we got a reduced Gröbner basis, it is immediate to construct a stratified and an optimal Gröbner basis from it.

3 Set constraint solver

Set constraints are a calculus for reasoning about relationships between sets and elements. Expressions in this calculus are built from set variables, element variables, the set operations such as union, intersection and complement. Set constraints consist of containment and equality relationships between these expressions. Set constraints can be represented in terms of polynomial equations of certain boolean rings. In order to manage finite domain constraints mathematically, we developed a set constraint solver in [Sa 97], where Gröbner bases computation methods described in the section 2 is employed for solving such polynomial equations.

Our solver can handle two types of objects, one is set and another is element. We can solve constraints consisting of these two types of variables with set operations such as $\cup$(union), $\cap$(intersection), ˜(complement), and relations such as $\subseteq$, $\subset$, $=$, $\neq$, $\in$, $\notin$.
We describe how the Gröbner bases computation method is applied for such set constraints through the following simple example of set constraints.
(For more complicated constraint such as including element variables, we have to use some delicate technique concerning admissible term orders; however, the essential part of our solver is also computations of Gröbner bases.)
Example

$$\{X \cup Y \subseteq \{a, b\}, \quad a \in X, \quad b \in Y, \quad X \cap Y = \emptyset\}$$

In this constraint, X and Y are variables for sets which we want to solve. a and b are constant symbols for elements.
The constraint is translated into polynomial equations of a boolean ring $P^{FC}(A)$.

$$\{(X*Y+X+Y)*\{a,b\}=X*Y+X+Y, \{a\}*X=\{a\}, \{b\}*Y=\{b\}, X*Y=0\}$$

Where A is a countable set of all constant symbols of elements, $P^{FC}(A)$ is a boolean ring consisting of all finite or co-finite subsets of A. Since every element of boolean ring is idempotent, the residue ring $P^{FC}(A)[X,Y]/(X^2 + X, Y^2 + Y)$ is more suitable for us to work on than the polynomial ring $P^{FC}(A)[X,Y]$ itself. Gröbner bases of such residue rings are naturally defined and all results described in section 2 also hold in residue rings. We call Gröbner bases of such residue rings *boolean Gröbner bases.* In order to solve the above polynomial equations, we compute the optimal boolean Gröbner basis of

$$\{(X*Y+X+Y)*\{a,b\}+X*Y+X+Y, \{a\}*X+\{a\}, \{b\}*Y+\{b\}, X*Y\}$$

and get the following

$$\{X + \{a\}, \quad Y + \{b\}\}.$$

This is the canonical form of the given constraint.
¿From this we can easily see that $X = \{a\}$ and $Y = \{b\}$.

The reason we work in $P^{FC}(A)$ instead of $P(\{a,b\})$(a boolean ring consisting of all subsets of $\{a,b\}$) is explained as follows.
If we subtract $X \cup Y \subseteq \{a,b\}$ from the above constraint, the canonical form is $\{X*Y, \{a,b\}*X+\{a\}, \{a,b\}*Y+\{b\}\}$. Therefore there are many(actually infinite) instance of solutions such as $X=\{a,c\}, Y=\{b,d\}$ besides $X=\{a\}$ and $Y=\{b\}$. On the other hand, the canonical form does not change as long as we are working in $P(\{a,b\})$.

In the computation of boolean Gröbner bases, however, we do not use whole portion of $P^{FC}(A)$. The only portion we need is a boolean sub-ring of $P^{FC}(A)$ that is generated by only the elements of $P^{FC}(A)$ that appear in the constraint. In the above example of constraint, this portion is a boolean ring generated by $\{a\}$ and $\{b\}$. The atomic elements of this boolean ring are $\{a\}$, $\{b\}$ and $\tilde{}\{a,b\}$, so this portion is isomorphic to the direct product $GF(2)^3$ of 3 Galois fields $GF(2)$. For a finite subset S of A, in general, a boolean ring generated by $\{\{e\}|e \in S\}$ is isomorphic to the direct product $GF(2)^{N+1}$ of $N+1$-many Galois fields $GF(2)$ where N is the cardinality of S. We denote this boolean ring by $P^+(S)$. This observation will play an important role in the parallel computation of boolean Gröbner bases discussed in the next section.

4 Parallel computation

We first describe a key theorem which enables us to have a parallel computation algorithm for our Gröbner bases. It is first mentioned as Theorem 2.3 of [W 89], although the proof is easy.

Let R be a sub-ring of the direct product $\prod\limits_{i \in S} K_i$ of fields K_i, $i \in S$.
For a polynomial f over R, f_i denotes a polynomial over K_i given from f by replacing every coefficient r of f with $r(i)$. For a set F of polynomials over R, we put $F_i = \{f_i | f \in F\}$.
Theorem 4.1
For a set F of polynomials and a set G of boolean closed polynomials of a polynomial ring $R[\bar{X}]$, the following two conditions are equivalent.

· G is a reduced Gröbner basis of the ideal (F) in $R[\bar{X}]$.
· G_i is a reduced Gröbner basis of the ideal (F_i) in $K_i[\bar{X}]$
for each element i of S.

For a given Von Neumann regular ring R, if we know its structure, that is, if we know a ring isomorphism Φ from R to a sub-ring of the direct product $\prod_{i \in S} K_i$, and S is a finite set, we can construct a reduced Gröbner basis G by computing G_i for every $i \in S$. When the cardinality of S is not so large and we can use parallel computation environment, this method (which will be called *the new method* in what follows) seems more efficient than the method based on our monomial reductions described in section 2 (which will be called *the old method* in what follows), because the computation of G_i is independent for each i and the construction of G from G_i $i \in S$ is immediate. In case the computation of K_i is much simpler than the computation of R for every $i \in S$, the new method is expected to be efficient even for sequential computations.

The boolean ring $P^+(S)$ used in our set constraint solver is isomorphic to the direct product $GF(2)^{N+1}$ of $N+1$-many Galois fields $GF(2)$, where N is the cardinality of S a set of constant symbols of elements appearing in a given set constraint. Since the computation of $GF(2)$ is much simpler than the computation of $P^+(S)$, the new method is expected to be more efficient than the old method.

We conclude this section with a simple example of the new method.

Example
Computation of the optimal boolean Gröbner basis of

$$\left\{ \begin{array}{l} (\tilde{}\{a,b\}) * X * Y + \{a\} * X + Y + \{b\} \\ X * Y + \{a\} * Y + X + \{a,b\} \end{array} \right.$$

by the new method is proceeded as follows.
The boolean ring $P^+(\{a,b\})$ is isomorphic to the direct product $GF(2)^3$. Let Φ be a ring isomorphism from $P^+(\{a,b\})$ to $GF(2)^3$ such that
$\Phi(\{a\}) = (1,0,0)$, $\Phi(\{b\}) = (0,1,0)$ and $\Phi(\tilde{}\{a,b\}) = (0,0,1)$. Φ is naturally extended to a ring isomorphism from $P^+(\{a,b\})[X,Y]$ to $GF(2)^3[X,Y]$.
Then $\Phi((\tilde{}\{a,b\})*X*Y+\{a\}*X+Y+\{b\}) = (0,0,1)*X*Y+(1,0,0)*X+Y+(0,1,0)$ and $\Phi(X*Y+\{a\}*Y+X+\{a,b\}) = X*Y+(1,0,0)*Y+X+(1,1,0)$.
(Since $(1,1,1)$ is an identity, we simply write Y instead of $(1,1,1)*Y$ for example.)
In order to get the optimal boolean Gröbner basis of

$$F = \left\{ \begin{array}{l} (0,0,1) * X * Y + (1,0,0) * X + Y + (0,1,0) \\ X * Y + (1,0,0) * Y + X + (1,1,0) \end{array} \right.$$

we have to compute a reduced boolean Gröbner basis G of F.
We compute a reduced boolean Gröbner basis G_i of F_i in a residue ring $GF(2)[X,Y](X^2+X, Y^2+Y)$ for each $i = 1,2,3$, and get
$G_1 = \{Y+1,\ X+1\}$, $G_2 = \{1\}$ and $G_3 = \{Y+X\}$.
¿From this we can construct a reduced boolean Gröbner basis

$$G = \left\{ \begin{array}{l} (1,0,0) * (Y+1) \\ (1,0,0) * (X+1) \\ (0,1,0) \\ (0,0,1) * (Y+X). \end{array} \right.$$

¿From this we can construct the stratified boolean Gröbner basis

$$\begin{cases} (1,0,1)*Y+(0,0,1)*X+(1,0,0) \\ (1,0,0)*(X+1) \\ (0,1,0) \end{cases}$$

and the optimal boolean Gröbner basis

$$\begin{cases} Y+(0,0,1)*X+(1,0,0) \\ (1,1,0)*X+(1,0,0) \\ (0,1,0). \end{cases}$$

Converting back to the expression of the boolean ring $P^+(\{a,b\})$ we finally get the desired optimal boolean Gröbner basis

$$\begin{cases} Y+\tilde{}\{a,b\}*X+\{a\} \\ \{a,b\}*X+\{a\} \\ \{b\}. \end{cases}$$

5 Experimental results

We implemented the new method as a parallel computation algorithm of boolean Gröbner bases in [S 99]. Our program is written in KLIC which is a parallel logic programming language developed at ICOT(Institute for new generation computer technology) and released as a free software([KLIC]). The KLIC compiler enables us to use actual parallel computations through PVM([PVM]). We had an amount of computation experiments using our program. The experiments gave us enough evidences that the new method is more efficient than the old one as we expected, that is (i) the new method is much faster than the old one even when the computation is sequential, (ii) the new method needs less memory than the old one and (iii) we can get reasonable speed up under the actual parallel computations.
We give three data of Gröbner bases computations in the following tables.

Data 1(14 variables 10 elements)						
	s-poly	bc	time	memory	process	CPU
sequential	4769	0	0:24	7116	1	1
parallel†	4769	0	0:05	7452	11	1
old	3848	495	2:08	25576	1	1

Data 2(20 variables 15 elements)						
	s-poly	bc	time	memory	process	CPU
sequential	54280	0	10:45	7128	1	1
parallel†	54280	0	1:16	6984	16	1
parallel	54280	0	5:10	6988	3	3
parallel	54280	0	3:03	6984	7	7
old	31313	2126	35:16	50216	1	1

Data 3(25 variables 20 elements)						
	s-poly	bc	time	memory	process	CPU
sequential	302973	0	698:59	155000	1	1
parallel	302973	0	217:03	99800	7	7
old	-	-	∞	∞	1	1

The number of *variables* is the number of indeterminates of a polynomial ring, which generally affects the size of the Gröbner basis computation. The number of *elements* is the cardinality of constant symbols of elements, which determines how many independent computations we need. The data are of three kinds of computations, *sequential* for sequential computations of the new method, *parallel* for parallel computations of the new method and *old* for computations of the old method. The computation time is measured by minutes:seconds. The computation memory is measured by kilobytes. The process number is the number of processes running through PVM. The CPU number is the number how many CPU's are actually used. We also give the number of S-polynomials(s-poly) and boolean closures(bc) created through the computations. For the computations of the new method, the number of S-polynomials is the total sum of the numbers of S-polynomials created through each independent computation. The computation of *parallel*† is a simulated parallel computation with only one CPU, although many processes are running through PVM. For this computation, the computation time given in the table is the maximum of the computation time of all processes. At the computation of the old method in data 3, the memory was exceeded.
We used 7 computers(PentiumII 400MHZ x 4, PentiumII 333MHZ x 3). Each of them has 512 Megabytes memory. The operating system is FreeBSD 3.1.

6 Conclusion and remarks

Our implementation is rather naive, we do not use any sophisticated technique of parallel programming. Nevertheless, our method is sufficiently effective on the actual parallel computations as we have showed in the previous section. The main reason is that our algorithm is a simple distributive memory parallel algorithm. In case we use enough number of processes, there are only two communications between the master process and each slave process. The first is for giving a set F_i of polynomials from the master to each slave, the second is for giving a reduced Gröbner basis G_i from each slave back to the master. Since the construction of G from G_i's is immediate as we mentioned, the computation time of the Gröbner basis G is almost same as the longest computation time of G_i's in case we can use ideal parallel computation. Therefore, when the computation time of one Gröbner basis G_i is enormously longer than the rest, we can not expect any speed up by our parallel computation. This is the worst case. On the other hand, when the computation time of each Gröbner basis is almost same, we can expect ideal speed up. The data of simulated parallel computations with only one CPU give us almost precise information for such situations. The ratio of the time of sequential to the time of parallel† in the table of previous section is the possible speed up by the parallel computation. It is 4.8 in the data 1 with 11 processes, 8.5 in the data 2 with 16 processes. In our whole experiment, the numbers of elements range from 2 to 35 and we got reasonable speed up in most cases. Because of the shortage of computers we can use, however, we were not able to try bigger scale of computations. Though we can expect that the new method is more efficient than the old one as in the data 3, we have to have parallel computation experiments to check efficiency of speed up for bigger scale of computations.

For the computation of each Gröbner basis G_i, we can also use a standard direct parallel computation based on distributed reductions of S-polynomials. Since this parallel computation is clearly independent of our parallel computation, we can easily combine them to get more speed up.

References

[KLIC] KLIC Version 3.002.
http://www.icot.or.jp/AITEC/COLUMN/KLIC/klic.html

[PVM] Parallel Virtual Machine http://www.epm.ornl.gov/pvm/

[Sa 97] Sato, Y. (1997). SET CONSTRAINT SOLVER Version 1.0.
http://www.icot.or.jp/AITEC/FGCS/funding/itaku-H8-index-E.html

[Sb 97] Sato, Y. (1997). Set Constraint Solver - Groebner bases for non-numerical domains -. International Symposium on Symbolic and Algebraic Computation(ISSAC 97), Poster Abstracts pp 13-14.

[S 98] Sato, Y. (1998). A new type of canonical Gröbner bases in polynomial rings over Von Neumann regular rings. International Symposium on Symbolic and Algebraic Computation(ISSAC 98), Proceedings pp 317-321.

[S 99] Sato, Y. (1999). SET CONSTRAINT SOLVER Version 2.0.
http://www.icot.or.jp/AITEC/FGCS/funding/itaku-H10-index-E.html

[SW 75] Saracino, D. and Weispfenning, V. (1975). On algebraic curves over commutative regular rings, Model Theory and Algebra, a memorial tribute to A.Robinson, Springer LNM Vol 498, 307–387.

[W 89] Weispfenning, V. (1989). Gröbner bases in polynomial ideals over commutative regular rings, EUROCAL '87, J.H. Davenport Ed., Springer LNCS Vol 378, 336–347.