Письма в ЭЧАЯ. 2009. Т. 6, № 7(156). С. 25–31

AN ALGORITHMIC APPROACH TO SOLVING POLYNOMIAL EQUATIONS ASSOCIATED WITH QUANTUM CIRCUITS[1]

V. P. Gerdt,[2] *M. V. Zinin*[3]

Joint Institute for Nuclear Research, Dubna

In this paper we present two algorithms for reducing systems of multivariate polynomial equations over the finite field F_2 to the canonical triangular form called lexicographical Gröbner basis. This triangular form is the most appropriate for finding solutions of the system. On the other hand, the system of polynomials over F_2 whose variables also take values in F_2 (Boolean polynomials) completely describes the unitary matrix generated by a quantum circuit. In particular, the matrix itself can be computed by counting the number of solutions (roots) of the associated polynomial system. Thereby, efficient construction of the lexicographical Gröbner bases over F_2 associated with quantum circuits gives a method for computing their circuit matrices that is alternative to the direct numerical method based on linear algebra. We compare our implementation of both algorithms with some other software packages available for computing Gröbner bases over F_2.

В работе представлены два алгоритма приведения полиномиальных уравнений со многими переменными над конечным полем F_2 к канонической треугольной форме, называемой лексикографическим базисом Гребнера. Данная треугольная форма наиболее удобна для нахождения решений полиномиальных уравнений. С другой стороны, системы многочленов над F_2 с переменными, также принимающими значение в F_2 (булевы многочлены), дают полное описание унитарной матрицы, задаваемой квантовой схемой. В частности, сама эта матрица определяется числом решений (числом общих корней многочленов) системы. Тем самым эффективное построение лексикографических базисов Гребнера над F_2 для полиномиальной системы, описывающей квантовую схему, дает метод вычисления унитарной матрицы этой схемы, альтернативный прямому использованию методов линейной алгебры. Дано сравнение эффективности программной реализации описанных алгоритмов с рядом других программ вычисления базисов Гребнера над F_2.

PACS: 03.67.Ac, 01.30.Cc, 03.67.-a

1. INTRODUCTION

1.1. Motivation. Constructing unitary matrices from quantum circuit [1–3], built from the Hadamard and Toffoli gates, is reduced to counting a number of solutions in F_2 of multivariate polynomial systems over F_2, associated with the circuits. Since these gates form

[1]The research presented in this paper was partially supported by grant 07-01-00660 from the Russian Foundation for Basic Research and by grant 5362.2006.2 from the Ministry of Education and Science of the Russian Federation.

[2]E-mail: gerdt@jinr.ru

[3]E-mail: mzinin@gmail.com

a universal gate basis [4], this construction may also be applied to circuits containing other quantum gates [5] due to the famous Solovay–Kitaev algorithm [6, 7]. The last algorithm is based on the Solovay–Kitaev theorem [8] which, when specified to the Hadamard and Toffoli gates, asserts that any circuit matrix can be approximated with arbitrary precision by a sequence of these gates. Given a quantum circuit containing the Hadamard and Toffoli gates only, construction of the multivariate polynomial system associated with the circuit is an easy task from the algorithmic and computational point of view [2, 3], which was observed first in [9]. Much more difficult problem is to count the number of common roots in the ground field for polynomials in the system. For this purpose we apply here the most universal algorithmic tool of investigating and solving a multivariate polynomial system of equations — Gröbner bases [10, 11] and involutive bases [12] which are Gröbner bases of special form. Both of these bases give the canonical form of the system which is uniquely defined by the initial system and the monomial ordering which, to be algorithmically admissible, must satisfy certain conditions [10]. The algorithms for constructing Gröbner bases designed in [10] and involutive bases designed in [12, 13] are called the Buchberger algorithm and Involutive algorithm, respectively.

Actually, Involutive algorithm constructs not the reduced Gröbner basis, but the involutive basis, which is usually redundant as a Gröbner basis and possesses some extra features. However, to solve systems for quantum circuits Gröbner basis it suffices to use Involutive algorithm just to output the corresponding reduced Gröbner basis which is a well-defined subset of the involutive basis [13].

1.2. Basic Notations and Definitions. Throughout the paper we use the following notations and definitions. $\mathbb{X} = \{x_1, \dots, x_n\}$ is the set of polynomial variables; $R = \mathbb{K}[\mathbb{X}]$ is a polynomial ring over field $\mathbb{K}$ of characteristic 0; $R' = \mathbb{F}_2[\mathbb{X}]$ is a polynomial ring over field $\mathbb{F}_2$; $\mathrm{Id}(F)$ is the ideal generated by polynomial set F; $\tilde{R} = \mathbb{F}_2[\{x_1, \dots, x_n\} \in \mathbb{F}_2^n]$ is a (*Boolean*) polynomial ring over field $\mathbb{F}_2$ whose variables take values in $\mathbb{F}_2$. Therefore, multiplication of monomials in this ring reads

$$m_1 \cdot m_2 = x_1^{i_1} \cdots x_n^{i_n} \cdot x_1^{j_1} \cdots x_n^{j_n} = x_1^{\max(i_1, j_1)} \cdots x_n^{\max(i_n, j_n)}.$$

As a monomial ordering we shall use the *pure lexicographic ordering* which is defined as follows: $m_1 := x_1^{i_1} \cdots x_n^{i_n} \succ m_2 := x_1^{i_1} \cdots x_n^{i_n}$ if $i_1 > j_1$ if and only if there exists $1 \leqslant k < n$ such that the first $k-1$ exponents of m_1 and m_2 are equal but the kth exponent of m_1 is larger than the kth exponent of m_2.

A monomial m will be called a *quotient of monomials b and a in the ring $\tilde{R}$* if $m \cdot a = b$ and $\deg(m) = \min\{\deg(m_i) \mid m_i \otimes a = b\}$. For a given finite set of polynomials $P = \{p_1, \dots, p_k\}$ the set $\mathrm{Id}(P) = \left\{ \sum\limits_{i=1}^{i=k} h_i p_i \mid p_i \in P,\ h_i \in R(R', \tilde{R}) \right\}$ is an *ideal generated by set P*.

Given an ideal $\mathcal{I} \subset R(R', \tilde{R})$ and an order $\succ$, a finite subset $G \subset R(R', \tilde{R})$ is a *Gröbner basis* of $\mathcal{I}$ if $(\forall f \in \mathcal{I})\, (\exists g \in G)\ [\mathrm{lm}(g) \mid \mathrm{lm}(f)]$, where $u \mid v$ denotes divisibility of monomial v by monomial u. For the further notations and definitions we refer to [12, 13].

The general strategy of the Gröbner (or involutive) bases approach is as follows. Given a set F of polynomials that describes the problem $F = \{f_1, f_2, \dots, f_m\}$, $f_i \in R(R')$, we transform F into the set G of polynomials which is a Gröbner or involutive basis and such that $\mathrm{Id}(F) = \mathrm{Id}(G)$. Hence polynomial sets F and G are fully equivalent, and in particular they have the same common roots. However, due to some special properties of Gröbner (or involutive) bases, many problems that are difficult for general F are «easier» for G.

2. IMPLEMENTED ALGORITHMS

2.1. Buchberger's Algorithm. We implemented the following algorithm as a version of Buchberger's algorithm [10] for the purpose of preliminary testing of data structures and for experimental comparison with the Involutive algorithm [12, 13] whose version for $\tilde{R}$ is considered below. Unlike the latter, the former algorithm deals with all S-polynomials (critical pairs). They are collected in set B at step 1.

Buchberger's Algorithm (F)

```
Input: $F \in R(R') \setminus \{0\}$ — finite set of polynomials
Output: $G$ — Gröbner bases of $\mathrm{Id}(F)$

 1: $B := \{[i,j] : 1 \leqslant i < j \leqslant \mathrm{length}(G)\}$
 2: while $B \neq \emptyset$ do
 3:    $[i,j]$ :=SelectPair$(B, G)$
 4:    $B := B \setminus \{[i,j]\}$
 5:    $h$ :=NormalForm(Spoly$(G_i, G_j), G)$
 6:    if $h \neq 0$ then
 7:       $G := G \cup \{h\}$;
 8:       $B := B \cup \{[i, \mathrm{length}(G)] : 1 \leqslant i < \mathrm{length}(G)\}$
 9:    end if
10: end while
11: return $G$
```

Then the **while**-loop starts at step 2. Here a pair from the set B is selected at step 3 and the normal form [10] of the selected S-polynomial is computed (step 5). The non-vanishing normal form is added to the basis at step 7 and set B is upgraded to include the new S-polynomials (step 8). The loop is terminated [10, 11] with a Gröbner basis G.

2.2. Involutive Algorithm. Involutive algorithm is based on partition of variables for every polynomial into two sets, one of which contains *multiplicative* variables, and the other contains *non-multiplicative* ones. This partition generates so-called *involutive division* of monomials [12, 13]. Accordingly, the product of a polynomial and its multiplicative variable is called *multiplicative prolongation* of the polynomial whereas its multiplication by a non-multiplicative variable is called *non-multiplicative prolongation.* We implemented Involutive algorithm for the Janet division [12]. For this reason we call the algorithm *Janet Division Based Involutive Algorithm.*

At the initial steps 1–3 of the algorithm a polynomial with the minimal leading monomial is chosen. Then it is inserted into set G, whereas the remaining polynomials are inserted into set Q. Then the **while**-loop starts. At step 5 a polynomial with the minimal leading monomial is chosen from the set Q. Then this polynomial is Janet-reduced [12] modulo set G. If its Janet normal form h is nonzero, then the corresponding polynomial is inserted into G at step 9. The insertion causes the augmentation of the set Q with non-multiplicative prolongations of all its elements done at steps 10 and 11. The *JanetNormalForm* function computes the Janet normal form [12,13] of the given polynomial modulo polynomials in G.

Janet Division Based Involutive Algorithm ($F, \succ$)

```
Input: $F \subset R(R') \setminus \{0\}$ — finite set of polynomials
Output: $G$ — Gröbner bases of $\mathrm{Id}(F)$
 1: choose $f \in F$ such that $\mathrm{lm}(f) = \min\{\mathrm{lm}(F)\}$
 2: $G := \{f\}$
 3: $Q := F \setminus G$
 4: while $Q \neq \emptyset$ do
 5:   choose $p \in Q$ such that $\mathrm{lm}(p) = \min_{\prec}\{\mathrm{lm}(Q)\}$
 6:   $Q := Q \setminus \{p\}$
 7:   $h :=$JanetNormalForm$(p, G)$
 8:   if $h \neq 0$ then
 9:     $G := G \cup \{h\}$
10:     for all $q \in G$ and $x \in NM_J(q, G)$ do
11:       $Q := Q \cup \{q \cdot x\}$
12:     end for
13:   end if
14: end while
15: return $G$
```

3. PECULIARITIES OF IMPLEMENTATION

3.1. Polynomials and Ideals in $\tilde{R}$. The ring $\tilde{R}$ has some significant differences from rings R and R'. For instance, any monomial order in the ring $\tilde{R}$ is not admissible in the usual sense [10,11]. Moreover, a basis consisting of single polynomial may not be a Gröbner basis at all [15] which is impossible in rings R and R'. For these reasons both Buchberger's and Involutive algorithms cannot be directly applied to the ring $\tilde{R}$ (for more details see [15]). Here we only note that in order to construct a Gröbner basis in ring $\tilde{R}$, one should compute a Gröbner basis in the ring R' by adding binomials $x_i^2 + x_i$ to the polynomial set for every monomial variable.

3.2. Vectorization. As far as all polynomials both in the initial basis and in the Gröbner basis contain all variables in the degree either 0 or 1, it is rather convenient from the computational point of view to use one-bit vectorization for the monomial inner data structures. By the one-bit vectorization we mean an opportunity to store exponents of the monomial variables by using for each variable one bit only. Here we use the bit arrays of lengths 32, 64 and so on depending on the number of variables under consideration: up to 32, up to 64, etc. Thus, the corresponding variable has the degree one if and only if this variable occurs in the monomial. From the programming point of view, exploitation of the maximal vectorization is a useful property to speed up computation.

But as said in Subsec. 3.1, we can only use ring R' and must add certain binomials. To store these binomials, the one-bit vectorization is not enough, and one has to deal with the two-bit vectorization.

In our implementation of Buchberger's algorithm we managed to avoid the use of these binomials and, thus, the one-bit vectorization is applied. But in Involutive algorithm the added binomials, as well as their non-multiplicative prolongations, affect the partition of the variables and cannot be omitted. For this reason, in our implementation of Invoutive algorithm the two-bit vectorization is exploited.

4. COMPUTER EXPERIMENTS

We did an intensive comparison of the running time for our implementation of the algorithms described above with some other computer algebra systems and packages implementing computation of Gröbner bases over F_2. Namely, we compared our timings with those for CoCoA 4.6 [16], Singular 3.0.2 [17], and Mathematica 5.0 [18].

As benchmarks we took some of the serial examples in famous collections [19] that are widely used for testing the Gröbner bases software. In doing so, those of the serial benchmarks were used that do not contain variables with exponents 2 or higher. These serial examples are *cyclic, redcyclic, noon,* and *katsura.* In addition to those serial benchmarks, a new series was taken from paper [20]. We call the last series *life* since they are derived in [20] from analysis of the famous Game of Life by J. Conway.

Some of the timings are depicted in Figs. 1–4. One can see that our implementation reveals, on average, a better behavior. Especially this applies to Involutive algorithm.

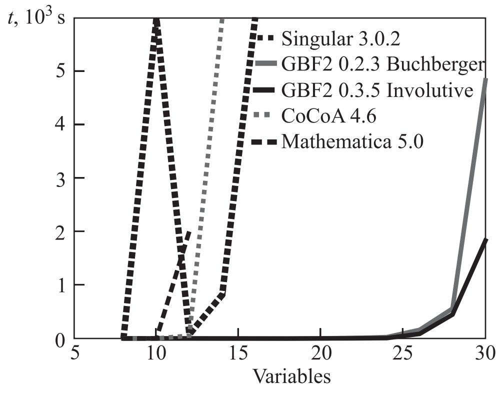

Fig. 1. Timings for *cyclic*

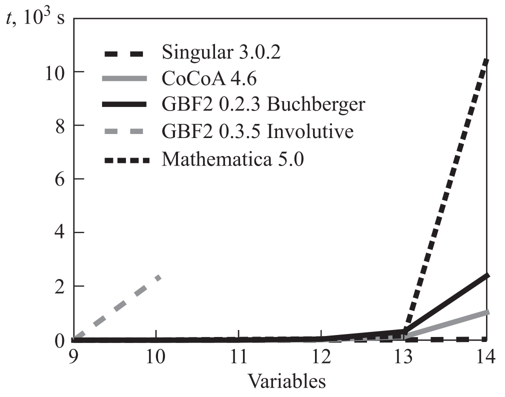

Fig. 2. Timings for *katsura*

5. FUTURE WORK

We are going to improve the above-presented version of Involutive algorithm as well as of its implementation for constructing Gröbner bases in the ring $\tilde{R}$. In particular, we plan:

• Improvement of the inner data structures. In the present implementation we use bitsets to store monomials and one-way lists for polynomials, although there may be more suitable structures.

• Search for more appropriate strategies for selecting critical pairs to be processed in Buchberger's algorithm and non-multiplicative prolongations in involutive algorithm.

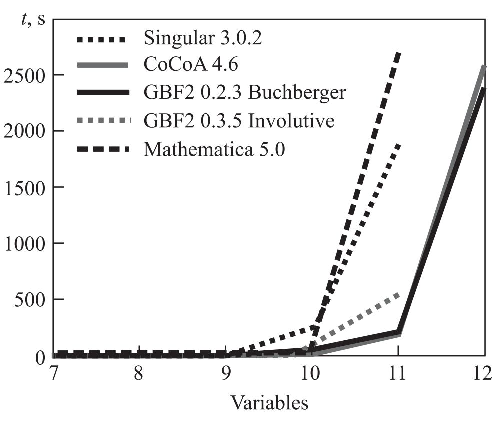

Fig. 3. Timings for *life*

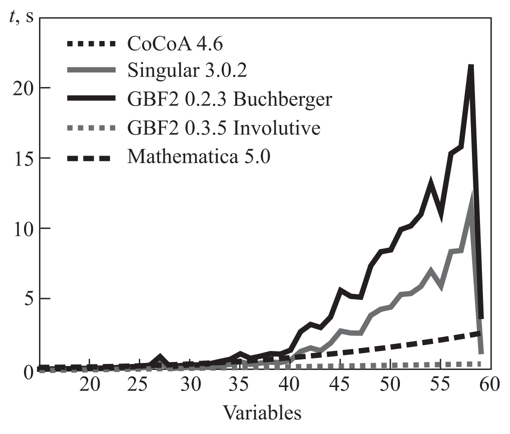

Fig. 4. Timings for *noon*

• Investigation of applicability to the ring $\tilde{R}$ of Involutive divisions different from the Janet one (Pommaret division [12], etc.).

• Installation of our implementation for Involutive algorithm in the open source computer algebra system GINV [21] as its module.

• Extension of the code to include counting of the number of solutions in F_2.

REFERENCES

1. *Dawson C. M. et al.* Quantum Computing and Polynomial Equations over the Finite Field Z_2. quant-ph/0408129.

2. *Gerdt V. P., Kragler R., Prokopenya A. N.* A Mathematica Package for Construction of Circuit Matrices in Quantum Computation // Comp. Algebra and Differential Equations. Acta Acad. Aboensis B. 2007. V. 67, No. 2. P. 28–38.

3. *Gerdt V. P., Kragler R., Prokopenya A. N.* A Mathematica Program for Constructing Quantum Circuits and Computing Their Unitary Matrices // This volume.

4. *Aharonov D.* A Simple Proof that Toffoli and Hadamard Gates Are Quantum Universal. quant-ph/0301040.

5. *Nielsen M., Chuang I.* Quantum Computation and Quantum Information. Cambridge Univ. Press, 2000.

6. *Dawson C. M., Nielsen M. A.* The Solovay–Kitaev Algorithm. quant-ph/0505030.

7. *Nagy A. B.* On an Implementation of the Solovay–Kitaev Algorithm. quant-ph/0606077.

8. *Kitaev A. Y., Shen A. H., Vyalyi M. N.* Classical and Quantum Computation // Graduate Studies in Math. Am. Math. Soc. Providence, Rhode Island, 2002. V. 47.

9. *Gerdt V. P., Severyanov V. M.* A C# Package for Assembling Quantum Circuits and Generating Associated Polynomial Sets // Part. Nucl., Lett. 2007. V. 4, No. 2. P. 225.

10. *Buchberger B.* Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory // Recent Trends in Multidimensional System Theory / Ed. N. K. Bose. Reidel; Dordrecht, 1985. P. 184–232.

11. Gröbner Bases and Applications / Eds.: Buchberger B., Winkler F. Cambridge Univ. Press, 1998.

12. *Gerdt V. P., Blinkov Yu. A.* Involutive Bases of Polynomial Ideals // Math. and Comp. in Simulation. 1998. V. 45. P. 519–542; math.AC/9912027; Minimal Involutive Bases // Ibid. P. 543–560; math.AC/9912029.

13. *Gerdt V. P.* Involutive Algorithms for Computing Gröbner Bases // Comp. Commutative and Non-Commutative Algebraic Geometry / Eds.: S. Cojocaru, G. Pfister, V. Ufnarovski. NATO Sci. Ser. IOS Press, 2005. P. 199–225; math.AC/0501111.

14. *Giovinni A. et al.* One Sugar Cube, Please, or Selection Strategies in the Buchberger Algorithm // Proc. of ISSAC'91. ACM Press, 1991. P. 49–54.

15. *Gerdt V. P., Zinin M. V.* On Computation of Gröbner Bases over F_2 // Comp. Algebra and Differential Equations. Acta Acad. Aboensis B. 2007. V. 67, No. 2. P. 59–68.

16. http://cocoa.dima.unige.it/

17. *Greuel G.-M., Pfister G., Schönemann H.* Singular 3.0. A Computer Algebra System for Polynomial Computations. Centre for Computer Algebra, Univ. of Kaiserslautern, 2005; http://www.singular.uni-kl.de

18. http://www.wolfram.com/

19. http://www-sop.inria.fr/saga/POL; http://www.math.uic.edu/˜ jan/demo.html

20. *Kornyak V. V.* On Compatibility of Discrete Relations. LNCS 3718. Springer-Verlag, 2005. P. 272–284; math-ph/0504048.

21. *Gerdt V. P., Blinkov Yu. A.* Specialized Computer Algebra System GINV // Programming and Computer Software. 2008. V. 34; the system is available on the Web page: http://invo.jinr.ru/