Fast Gröbner Basis Computation for Boolean Polynomials

Franziska Hinkelmann[a,b] Elizabeth Arnold[c*]

October 14, 2010

[a]Department of Mathematics, Virginia Polytechnic Institute and State University,
Blacksburg, VA 24061-0123, USA
[b]Virginia Bioinformatics Institute, Virginia Polytechnic Institute and State University,
Blacksburg, VA 24061-0477, USA
[c]Department of Mathematics and Statistics, James Madison University,
Harrisonburg, VA 22807, USA
[*]Corresponding author: arnoldea@math.jmu.edu

Abstract

We introduce the Macaulay2 package *BooleanGB*, which computes a Gröbner basis for Boolean polynomials using a binary representation rather than symbolic. We compare the runtime of several Boolean models from systems in biology and give an application to Sudoku.

1 Introduction

Buchberger's algorithm will theoretically compute a Gröbner basis for any ideal of a multivariate polynomial ring with rational coefficients. However, due to memory constraints, many examples still cannot be computed in real time. Since all computations are done symbolically over the rational numbers, two difficulties occur. The coefficients of the polynomials during the computation can grow very large, and also the degrees of the polynomials can grow very large allowing a large number of monomials in each polynomial. If an ideal consists of Boolean polynomials in the quotient ring, $QR = \mathbb{F}_2[x_1, \dots, x_n]/\langle x_1^2 + x_1, x_2^2 + x_2, \dots x_n^2 + x_n\rangle$, we can modify Buchberger's algorithm so that these two issues are avoided entirely. These ideas have been implemented and discussed before in [4, 10, 3]. In this paper, we describe a fast Gröbner basis implementation for Boolean polynomials in *C++* which is included in the Macaulay2 release version 1.4 [8].

2 Algorithm

2.1 Binary Representation

Consider a polynomial in the ring $QR = \mathbb{F}_2[x_1, \ldots, x_n]/\langle x_1^2 + x_1, x_2^2 + x_2, \ldots x_n^2 + x_n \rangle$. Since the Boolean polynomials are multilinear, we can represent a monomials in n variables as a binary string of length n. We store these strings as integers. Polynomials, then, are just lists of these integers. Arithmetic on the binary representation of these polynomials is very fast. To add two monomials, we can just concatenate the binary strings. If we add two identical monomials, they sum to zero. To multiply two monomials, we use the binary operation OR. Since $x_i^2 = x_i$ for every i, if x_i appears in either monomial, then x_i will appear in the product. For the Boolean Gröbner basis algorithm, we only need to multiply a monomial by a polynomial. To do this, we just multiply the monomial by each monomial in the polynomial using bitwise OR. Then we add the monomials by concatenation, removing duplicate pairs.

The other operations that we need for Buchberger's algorithm are least common multiple and division of monomials. The least common multiple of two monomials is again just bitwise OR. In fact, the operation of least common multiple in this Boolean ring is just multiplication. To divide one monomial by another, we must first check divisibility. If a monomial m_1 is divisible by a monomial m_2, then any variable appearing in m_2 must also appear in m_1. So m_1 is divisible by m_2 if and only if bitwise $m_1 - m_2 > 0$. Once divisibility has been established, then division is binary exclusive OR. Note that this is nothing more than subtraction. In order to implement Buchberger's first criterion, we also need an operation to determine if two monomials are relatively prime. This can be determined by comparing the minimum value for each variable in the monomial. If the minimum for each variable is 0, then the monomials are relatively prime. Now we have enough operations to execute Buchberger's algorithm.

2.2 Buchberger's Algorithm

Given a set of Boolean polynomials, F in $QR = \mathbb{F}_2[x_1, \ldots, x_n]/\langle x_1^2 + x_1, x_2^2 + x_2, \ldots x_n^2 + x_n \rangle$, we want to compute a Gröbner basis for the ideal generated by F using Buchberger's Algorithm. The basic Buchberger Algorithm computes $S-$polynomials for every pair of polynomials in F. It is important to note that when working in a quotient ring, we need to perform Buchberger's algorithm on the ideal generated by F together with the field polynomials. For example, consider the ideal generated by $xy + z$ in

$\mathbb{F}_2[x,y,z]/\langle x^2+x, y^2+y, z^2+z\rangle$. Since there is just one polynomial in the ideal, Buchberger's algorithm produces the the Gröbner basis for this ideal as $xy+z$. However, $xz+z$ is also in this ideal, since $xz+z = x(xy+z)+(xy+z)$. In fact, the Gröbner basis for the ideal generated by $xy+z$ in $\mathbb{F}_2[x,y,z]/\langle x^2+x, y^2+y, z^2+z\rangle$ is $\{xy+z, yz+z, xz+z\}$. $xz+z$ is the reduction of the S-polynomial of $xy+z$ and x^2+x.

So the first difficulty in implementing Buchberger's algorithm using the binary representation of Boolean polynomials is how to encode these quadratic field polynomials. This difficulty is bypassed using the following result.

Theorem 2.1. *The S-polynomial of any multilinear Boolean polynomial and a quadratic field polynomial is always multilinear.*

Proof. Let f be a mutilinear Boolean polynomial with leading term $\mathrm{lt}(f)$. If $\mathrm{lt}(f)$ and x_i are relatively prime, then the S-polynomial of f and $x_i^2+x_i$ will reduce to 0 by Buchberger's first criterion [5]. So we only consider quadratic field polynomials whose variable in the leading term is contained in $\mathrm{lt}(f)$. Suppose x_i is in $\mathrm{lt}(f)$. Then $S(f, x_i^2+x_i) = f+(\frac{\mathrm{lt}(f)}{x_i})(x_i^2+x_i) = f-\mathrm{lt}(f)+x_i$ which is multilinear. □

Therefore, no special encoding for quadratic field polynomials is necessary.

2.3 Implementation

The package *BooleanGB* contains the algorithm *gbBoolean*, which is an implementation of Buchberger's algorithm for binary representations of Boolean polynomials over the quotient ring, $QR = \mathbb{F}_2[x1, x2, \ldots, x_n]/\langle x_1^2+x_1, x_2^2+x_2, x_n^2+x_n\rangle$. The algorithm is implemented in *C++* and is part of the Macaulay2 engine. *gbBoolean* computes a reduced Gröbner basis in lexicographic order for an ideal of Boolean polynomials in QR. All computations are done bitwise instead of symbolically, by representing a monomial as bits in an integer and a polynomials as a list of monomials. On a 64 bit machine, the algorithm works in the ring with up to 64 indeterminates. The algorithm uses Buchberger's first and second criteria [5] as described in [6, p. 109]. Here is algorithm computing the example from Section 2.2.

```
i1 : loadPackage "BooleanGB";
```

```
i2 : R = ZZ/2[x,y,z];
```

```
i3 : QR = R/ideal(x^2-x, y^2-y, z^2-z);

i4 : I = ideal(x*y+z);

o4 : Ideal of QR

i5 : gbBoolean I

o5 = ideal (x*y + z, y*z + z, x*z + z)

o5 : Ideal of QR
```

The code for *gbBoolean* is freely available with the source code distribution of Macaulay2 [8].

3 Applications

Boolean Gröbner basis algorithms such as *gbBoolean* can be used for any system of Boolean polynomials. In particular, exact cover problems, satisfiability problems and problems in systems biology described below are examples. In some cases, it may be possible to turn a system of non-Boolean polynomials into a Boolean system. This involves dramatically increasing the number of variables. But in the example of Sudoku, described below, the time saved by the bitwise computations outweighs the increase in variables.

3.1 Boolean Models in Systems Biology

Logical models [1] are widely used in systems biology. They can be translated to polynomial dynamical systems [12], in particular, logical models with binary variables result in Boolean polynomials. Key dynamic features of logical models such as fixed points correspond to the points in algebraic varieties generated by the polynomials describing the model. To assure that *gbBoolean* is efficient on “practical” ideals, we translate all binary logical models in the GINsim repository [9] to Boolean polynomial systems using ADAM [13], and compute the Gröbner basis of the ideal describing the fixed points. For all logical models in the repository, *gbBoolean* is faster than current Macaulay2 implementations, run-times are are depicted in Fig. 1 and table 1.

3.2 Sudoku

Sudoku is a popular game played on a 9×9 grid where the numbers 1-9 are filled in so that each number appears exactly once in each row column and 3×3 block. These constraints can be represented by polynomials, with one variable for each cell in the grid. (See [2] and [7] for more details.) For ease of demonstration, we use Shidoku as an example. Shidoku is played with the same rules as Sudoku, but on a 4×4 grid with the numbers 1-4.

For Shidoku, we use 16 variables that can each take on only the values 1-4. We can represent this fact, together with the constraints of the game, in a total of 40 polynomials. If we consider the ideal generated by these polynomials in $\mathbb{Q}[x_1, \ldots, x_{16}]$ with the lexicographical ordering, we cannot compute the Gröbner basis for the ideal in Macaulay2 in a reasonable amount of time. We can, however, convert the problem to a system of Boolean polynomials and use *gbBoolean* to compute the Gröbner basis.

The Boolean system has 64 variables, representing each of the possible values 1-4 for each of the 16 cells. We consider the ideal of the polynomials in $QR = \mathbb{F}_2[x1, x2, \ldots, x_n]/\langle x_1^2 + x_1, x_2^2 + x_2, x_n^2 + x_n \rangle$. The Boolean system gives us a total of 256 polynomials to represent the Shidoku constraints. As seen in Section 4 below, *gbBoolean* computes the Gröbner basis for the Boolean ideal in just 4.1 seconds.

4 Results

We test *gbBoolean* on several random ideals and compare the run times with the standard Gröbner basis implementation in Macaulay2. In addition, we also test published Boolean logical models stemming from biological systems and compute the basis of the ideals corresponding to key dynamic features. Finally, we test the algorithm on Sudoku, a problem that can be described by a system of Boolean polynomials.

We compare *gbBoolean* to the symbolic computation in lexicographic order. Since the graded reverse lexicographic (gRL) monomial order usually yields the fastest calculation, we also compare *gbBoolean* to the combined times of computing a basis in gRL and lifting it to the quotient ring with lexicographic order.

Fig. 1 shows the run-times on a 3.06 GHz Intel Core 2 Duo MacBook Pro. In almost all examples, *gbBoolean* is faster than current implementations in Macaulay2. Cumulatively, *gbBoolean* is about four times faster (0.27614741%) than the Gröbner basis calculation in the ring with gRL order, and 50 times faster (0.017562079%) than the calculation in the ring with lexicographic order. We do not list run-times for symbolic computations using the Sugar strategy [11] because it is much slower than the "sugarless" strategy for all examples.

	QR Lex	QR Lift	gbBoolean
II0	220.681	19.0562	4.4871
S2	0.002174	0.003237	0.00001
S3	0.0032	0.004926	0.000009
SS2	0.003438	0.005541	0.00001
Shidoku	470.703	19.6797	4.11442
TCR	0.007104	0.012106	0.000208
THBool	0.001946	0.003759	0.000009
BoolCC	0.002007	0.003859	0.000012
erbb2	0.002609	0.003017	0.00001
yeastLi	0.002593	0.002013	0.00009

Table 1: Run Times for Random, Biological, and Shidoku Examples

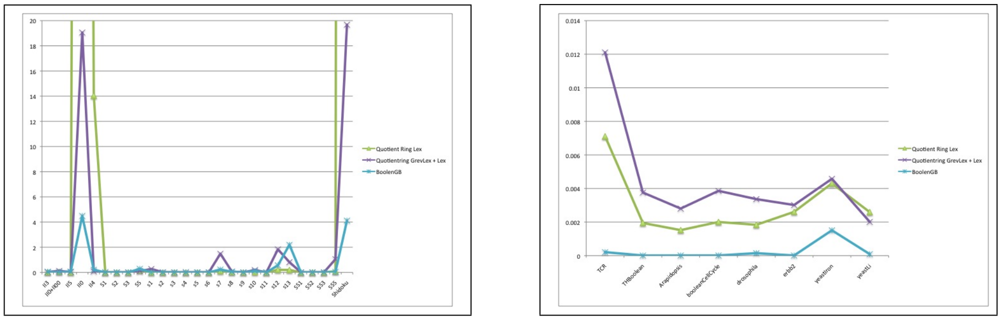

Figure 1: Benchmarks: The left graph depicts run-times for random ideals and an application to Sudoku, the right graph for Boolean ideals originating from biological systems.

5 Conclusion

Computing Gröbner bases for Boolean polynomials bitwise rather than symbolically can speed up computation time considerably. In the example of Shidoku, even though converting the problem from $\mathbb{Q}$ to $\mathbb{F}_2$ increases the number of variables from 16 to 64, the time to compute the Gröbner basis decreases. The package *BooleanGB* in Macaulay2 has proven to be generally faster than the standard algorithm in Macaulay2 over the quotient ring for Boolean polynomial ideals. Further improvements might be achieved by implementing the Sugar strategy [11] or other monomial orders.

Acknowledgments

The authors thank Mike Stillman for adding the *C++* implementation to the M2 engine and for his improvements of the implementation. Furthermore, the authors thank Samuel Lundqvist, Amelia Taylor, Dan Grayson, and Reinhard Laubenbacher for stimulating discussions, insightful comments, and organizing several Macaulay2 workshops.

References

[1] Aurélien Naldi, D. Berenguier, Adrien Fauré, F. Lopez, Denis Thieffry, and Claudine Chaouiya. Logical modeling of regulatory networks with ginsim 2.3. *Biosystems*, 97(2):134:139, 2009.

[2] E. Arnold, S. Lucas, & L. Taalman, Gröbner Basis Representation of Sudoku, *College Mathematics Journal*, **41**, 2010, 101-111.

[3] A. Bernasconi, B. Codenotti, V. Crespi & G. Resta, Computing Groebner bases in the Boolean setting with applications to counting, in: G. Italiano & S. Orlando, eds., *Proceedings of the Workshop on Algorithm Engineering (WAE'97)*, University of Venice, Venice, September 11-13, 1997, 209–218.

[4] M. Brickenstein, A. Dreyer, PolyBoRi: A framework for Gröbner-basis computations with Boolean polynomials, *Journal of Symbolic Computation*, Volume 44, I(9), September 2009, Pages 1326-1345.

[5] B. Buchberger, *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*, Ph.D. Thesis, Inst. University of Innsbruck, Innsbruck, Austria, 1965.

[6] D. Cox, J. Little, & D. O'Shea, *Ideals, Varieties, and Algorithms*, Springer-Verlag, 1992.

[7] J. Gago-Vargas, I. Hartillo-Hermoso, J. Martín-Morales, & J.M. Ucha-Enríquez, Sudokus and Gröbner bases: not only a divertimento, *Computer algebra in scientific computing*, 155–165, *Lecture Notes in Comput. Sci.*, **4194**, Springer, Berlin, 2006.

[8] D. Grayson and M. Stillman, *Macaulay2, a software system for research in algebraic geometry*, Available at http://www.math.uiuc.edu/Macaulay2/.

[9] *GINsim Model Repository*,
Available at http://gin.univ-mrs.fr/GINsim/model_repository.html/.

[10] Y. Sato, A. Nagai, & S. Inoue, On the Computation of Elimination Ideals of Boolean Polynomial Rings, in: D. Kapur, Ed., *Computer Mathematics: 8th Asian Symposium, ASCM 2007, Singapore, December 15-17, 2007, Revised and invited Papers*, Lecture Notes In Artificial Intelligence, *5081*, Springer-Verlag, Berlin, 2008, 334–348.

[11] A. Giovini, T. Mora, G. Niesi, L. Robbiano, & C. Traverso, "One Sugar Cube, Please?, or Selection Strategies in the Buchberger Algorithm." *Proceedings of the International Symposium on Symbolic and Algebraic Computation.* pp. 49-54, June 1991.

[12] Alan Veliz-Cuba, Abdul S. Jarrah, & Reinhard Laubenbacher. Polynomial algebra of discrete models in systems biology. *Bioinformatics*, 26(13):16371643, July 2010.

[13] Franziska Hinkelmann, Madison Brandon, Bonny Guang, Rustin Mc- Neill, Alan Veliz-Cuba, Greg Bleckherman, & Reinhard Laubenbacher. Adam, analysis of dynamic algebraic models. Available at http://adam.vbi.vt.edu/.