Computer Science Journal of Moldova, vol.**9**, no.3(27), 2001

Non-commutative computer algebra and molecular computing *

Svetlana Cojocaru Victor Ufnarovski

Abstract

Non-commutative calculations are considered from the molecular computing point of view. The main idea is that one can get more advantage in using molecular computing for non-commutative computer algebra compared with a commutative one. The restrictions, connected with the coefficient handling in Gröbner basis calculations are investigated. Semigroup and group cases are considered as more appropriate. SAGBI basis constructions and possible implementations are discussed.

1 Introduction

Every woman knows that when a new fashion comes in power it is time to reconsider her dresses and to plan what can be done to arrange new ones or to adjust one of the available. The same is true for any mathematical discipline. A new fashion that is coming in power is molecular computing and we would like to consider Computer Algebra from this new fashion point of view – what can be done and what should be adjusted. So let us first analyze which possibilities suggest the new trend.

Starting with the first successful experiment in 1994 the molecular computing involves a lot of researchers into developing of theory and practice of this domain. In [6] the three directions are mentioned:

* The authors acknowledge the very helpful contribution of *INTAS project* 97-1259, *NATO project* PST.CLG.976912 and *Royal Swedish Academy project* N1023505 for enhancing their cooperation, giving the best conditions for producing the present result.

- laboratory experiments,
- elaboration and developing the theoretical models,
- algorithms for solving of the concrete problems using the technique of molecular computing

One can find n large bibliography (maintained by P.Frisco) on http://www.wi.LeidenUniv.nl/ pier/dna.html, which gives the possibility to be convinced that now we have a lot of quite developed theories and methods to solve some hard (from the computational point of view) problems. Among them the most popular are the algorithms for the different graphs problems, SAT problem, the data encryption breaking etc. In other words there are first of all computationally hard problems, especially those where some kind of parallelism can give the advantage or at least some hope to achieve results which would be impossible by usual computations, e.g. to solve a known NP-problem in polynomial time.

Discussing the class of problems for which molecular computing is the preferable technique, D.Wood and R.Williams affirm, that one of the suitable may be Computer Algebra. According to [10]

1. Computer Algebra is similar to DNA laboratory reactions: both rearrange identical units.
2. DNA is an attractive medium for computation because of its potential parallelism.
3. Both DNA reaction and computer algebra are performing the transformation of exact, non-numeric input into exact, non-numeric output.

There are not so many results in computer algebra where the technique of molecular computing was successfully applied. We can mention, for example, symbolic determinants or permanents expansion (R.Williams, D.Wood [10]), a surface based algorithm for the expansion of symbolic determinants (Z.Frank Qiu, Mi Lu [5]), matrix multiplication (E.Oliver [9]). Probably this list can be continued, but in any

case it is not very large and the main aim of this article us to suggest some other possible directions for applications of molecular computing in Computer Algebra and to consider some obstacles in those directions. The main conclusion of our article is that it is non-commutative Computer Algebra that fits better to the ideology of molecular computation.

2 Gröbner basis computation

The kernel of any developed Computer Algebra package is the implementation of the Gröbner basis computation, which has a series of useful applications, for example, it permits to solve (in some sense) systems of the polynomial equations.

The notion of the Gröbner basis and the algorithm of its computation was proposed by Bruno Buchberger [1, 2, 3]. Is there a possibility to apply the molecular calculations?

Let us first recall the definitions. If I is an ideal in the polynomial algebra $K[X]$ then factor-algebra $A = K[X]/I$ has a linear basis (e.g. a basis as a vector space) consisting sf monomials, e.g. words in the alphabet X. To construct such a basis suppose that the set of all monomials is well-ordered, the unit is the least monomial and the order is preserved under multiplication, e.g. $f > g \Rightarrow fh > gh$. Then the linear basis N of A consists of normal monomials – those which cannot be rewritten (in A) as linear combinations of lower words.

Let F be the set of all monomials that are not normal by themselves, but are minimal in this sense, i.e. have the property that every its proper factor is a normal monomial. It is not difficult to see that the set F is finite and N can be recovered from F as the set of all monomials that are not divisible by any monomial from F.

Being not normal, every $f_i \in F$ can be rewritten in A as u_i – uniquely determined linear combination of normal words. The set $G = \{f_i - u_i\}$ is called a (reduced) Gröbner basis. In his original work Buchberger have found how to construct G from the set of defining relations of algebra A. Algorithm itself can be parallelized in a very natural way, but computationally it is hard and this makes it attractive

from the molecular calculations point of view.

The key moment in the Buchberger's algorithm is so-called S-polynomial of two polynomials p and q, which is constructed by considering the least common multiple of their highest terms. For example, if they are represented by monomials $xxyyzz$ and $xyyyzt$, then S-polynomial is constructed with the help of $xxyyyzzt$. The first problem which we meet here from the molecular computations point of view is that it is difficult to obtain such a monomial using the concatenation as a basic operation.

But suppose now that an algebra A is non-commutative. We can repeat the definition above replacing $K[X]$ by free algebra $K < X >$. The main difference is that now F and G should not be finite in general case. On the other hand the corresponding analog of Buchberger's algorithm (so-called Mora's algorithm, see [7]) uses the construction that is similar to the S-polynomial, but uses words that are formed by the concatenations only. More exactly, a composition of two (non-commutative) polynomials p and q is a word abc, such that ab is a leading monomial (word) for p and bc for g. So, it is a natural concatenation, which can be easily performed by the molecular computations.

The reduction, which is another important part of any computer algebra calculations, is also much more natural in the non-commutative case, where it can be realized by the series of substitutions of a given word (sample-word) by another one. In the commutative case reduction is more complicated. For example if xz should be reduced by yt, then the monomial $xyzz$ is replaced by $yyzt$, which is much more complicated construction from the DNA-operations point of view.

We already have mentioned the important difference between commutative and non-commutative Computer Algebra: most problems in commutative case are finite (because Gröbner basis is finite), so molecular computing applications are mainly interesting for huge, for example NP-problems. It means that the only use of the molecular computing approach is it massive character, its advantage to produce a lot of variants.

Non-commutative algebra is much more interesting in sense that most problems are infinite and to solve them means usually to present

an infinite solutions in the finite form (e.g. an automaton), or better to say in some kind of program, which can produce this infinite solution. Because in the nature the DNA are exactly compact programs, producing rather complicated solutions (life forms, for example) it is DNA-language that promises a lot here.

An easy example is the calculation of the Gröbner basis in an algebra with the single relation $xx = xy$, which produce an infinite Gröbner basis

$$xx - xy,$$

$$xyx - xyy,$$

$$xyyx - xyyy, \ldots$$

It is not so difficult to create a prediction algorithm which generates this Gröbner basis, using DNA-calculations (or ordinary ones), but it is not clear how to produce a corresponding proof that this really is a Gröbner basis. Of course, in general this problem is unsolvable, but for important cases such type of regularity often happens and, may be, it is more naturally to have this proof in a DNA-form.

3 Binomial and group relations

Another problem, arising both in the commutative and non-commutative Gröbner basis calculations is coefficient handling. Both for the reduction and the calculations of S-polynomials we need linear combinations of words, but for this we need arithmetic over ground field K, which is far from being trivial from the molecular calculations point of view. Of course, it if an interesting problem by itself even for finite fields (though small fields does not look difficult to implement). We do not plan to discuss this area now and instead want to mention that in the beginning the most natural way to escape this problem is to restrict possible algebras by binomial relations only, e.g. to consider only those algebras whose defining relations have the form $f = g$, where both f and g are monomials. In other words we can restrict our algebras by semigroup algebras only. Even here we still have an

infinite Gröbner bases and prediction problems are still difficult. Some approach here can be found in the article [8] and to implement the corresponding approach from the molecular calculations point of view looks as a perspective problem.

Let us consider another approach for Gröbner basis calculations, which does not use Buchberger's or Mora's algorithm, but can use some advantages that the molecular calculation has. Suppose for simplicity that all binomial relations are homogeneous. Then we can create Gröbner basis inductively – degree after degree. Suppose we know all elements of Gröbner basis in the degree n. Then we know all normal words in the degree n and can multiply them by one letter to get (a lot of) words of length $n + 1$. Trying to reduce them by known elements of Gröbner basis we get new relations if we obtain the same word as a result of the reduction of two different words. If we consider all the possibilities we will get all the elements of the Gröbner basis in the degree $n + 1$. This sounds not so efficient. But suppose now, that we know a bound for the number of normal words (for example, that we know a Hilbert series – it is always the case in the generic algebras). Then the situation changes dramatically – we do not need to check all the possibilities – if we have got sufficiently many new relations we know that they already form the Gröbner basis in the degree $n + 1$! This is exactly what we can do in the molecular calculations: produce a lot of combinations rather chaotically. In other words we can create a set that with a high probability would be the desired Gröbner basis (always in the generic case or if we achieved the known bound of the Hilbert series).

This approach may be used even in the non-homogeneous case, but here we never will be sure that the obtained set is a Gröbner basis already. Of course one of the most important case of semigroup is a group. Even the implementation of the quite elementary calculations in a free group looks as non-trivial, but reasonable problem for the molecular computing approach. Another interesting problem that looks very promising and useful is the implementation of the Todd-Coxeter algorithm (see [4]) of coset enumeration.

Here we would like to mention one more advantage of the non-

commutative approach. For the non-commutative group with 2 generators we need 4 semigroup generators (two additional generators for the inverse elements). We have these 4 letters in the DNA, so we do not need additional encoding. Even for algebra case 4 generators are sufficiently many. But what is more important that a free subgroup with a larger number of generators (even infinitely many) is simply a subgroup of our 2-generated group. The same is true for non-commutative algebras, but it is not the case for the commutative algebra: the polynomial ring with 3 generators cannot be realized as a subring of a 2-generated polynomial algebra! From all points of view Computer Algebra in new fashion should be non-commutative.

4 SAGBI-basis

Another important Computer Algebra problem is SAGBI-basis calculations. (SAGBI stands for Subalgebra Analog of Gröbner Basis to Ideals). The idea is the following. A set G is a Gröbner basis for ideal I, if an ideal, generated by its highest monomials consists of all highest monomials from the ideal I. If we replace the word ideal by subalgebra we get the definition: a set S is a SAGBI basis for a subalgebra A if the subalgebra, generated by highest monomials from S, coincides with the set of highest monomials in A. One of the example is $S = xx - y, yy - x$. But $S = xyy - y, x, xy - y, yx$ is not SAGBI basis for the subalgebra it generates, because $(xyy - y) * x - (xy - y) * (yx) = yyx - yx$ and yyx cannot be written as the product of words xyy, x, xy, yx.

To complete the last S to SAGBI-basis we need to add the new calculated element and consider new ambiguities, for example $x * (yyx) = xy * yx$. One nontrivial problem in SAGBI-basis calculations is to determine all necessary ambiguities for a given set of words. Here the massive parallelism, offered by the molecular computing technique, can be very useful too.

References

[1] B.Buchberger, *On Finding a Vector Space Basis of the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal (German)*. PhD Thesis, Univ of Innsbruck, Austria, 1965

[2] B.Buchberger, *A Theoretical Basis for the Reduction of Polynomials to Canonical Forms*. ACM SIGSAM Bull.,1970, 10/3: pp.19–29, and 10/4: 19-24.

[3] B.Buchberger, *A Criterion for Detecting Unnecessary Reductions in the Construction of Groebner Bases*. In: Proc. of the EUROSAM 79 Symp. on Symbolic and Algebraic Computation, Marseille, June 26-28, 1979, (Lect. Notes in Comp. Sci. 72, Springer): pp.3–21.

[4] H.S.M. Coxeter, W.O.J. Moser, *Generators and relations for discrete groups*, Third. Ed. Ergeb. Math 14, Springer Berlin 1972.

[5] Z.Frank Qiu, Mi Lu, *A Surface-Based DNA Algorithm for the Expansion of Symbolic Determinants*. Available at http://ee.tamu.edu/ zhiquan/dna/pub/symb.ps.

[6] C.Maley, *DNA Computation:Theory, Practice and Prospects*. Evolutionary Computation, 6(3):pp.201–229, 1998. Available at http://mitpress.mit.edu/journals/EVCO/sample-article.html .

[7] T.Mora, *Gröbner bases for non-commutative polynomial rings*. In: J.Calmet (ed.), AAECC-3,1986, Lect. Notes Comp. Sc. 229: pp.353–362.

[8] J.Månsson, P.Nordbeck, *Regular Gröbner basis*. To appear in J. of Symbolic Computations.

[9] J.Oliver, *Computation With DNA – Matrix Multiplication*. American Mathematical Society. Proceedings of the Second Annual Meeting on DNA Based Computers, held at Princeton University, June 10–12, 1996., DIMACS: Series in Discrete Mathematics and

Theoretical Computer Science., ISSN 1052-1798, 1996. Available at http://www.chem.brown.edu/brochure/people/jso/DNA.html.

[10] R.Williams, D.Wood, *Exascale Computer Algebra Problem Interconnect with Molecular Reactions and Complexity Theory.* DIMACS series in Discrete Mathematics and Theoretical Computer Science. Vol.44, 1999, pp.267–275.

S. Cojocaru, V. Ufnarovski, Received November 20, 2001

Svetlana Cojocaru,
Institute of Mathematics and Computer Science,
Academy of Sciences of Moldova,
str. Academiei, 5, Chişinău, MD 2028, Moldova.
E–mail: sveta@math.md

Victor Ufnarovski,
Lund University,
Department of Mathematics
Sölvegatan, 18,
Box 118, S-22100,
Lund, Sweden,
E–mail: ufn@maths.lth.se