Algorithms and Software for Symbolic Solvers of Polynomial Systems

Marc Moreno Maza
University of Western Ontario
CS652 – Winter 2004

February 4, 2005

1 Introduction

The proposed projects aim to develop new techniques or enhance existing ones for computing triangular decompositions. Each of these projects involves algorithmic and experimental activities and should lead to very interesting results.

The *lextriangular*[1] algorithm [Laz92] and the TRIADE[2] algorithm [Mor99] implemented in AXIOM, ALDOR and MAPLE will be the core of these projects.

One of the key ideas behind these two algorithms is the *D5 Principle* [DDD85] presented during the lectures in February. The *lextriangular* algorithm assumes that its input is a lexicographical Gröbner Basis of a zero-dimensional ideal whereas the TRIADE algorithm can take as input any finite set of polynomials. In addition, when the same input can be used with both algorithms, the output is essentially the same. A more detailed presentation of these algorithms will be given during the lectures in March.

Section 2 shows how to use the AXIOM implementations of *lextriangular* and TRIADE. Remember that AXIOM is freely accessible on the web. It is

[1] *lextriangular* is an acronyme for *LEXicographical to TRIANGULAR, meaning transformation of a lexicographical Gröbner Basis to a triangular decomposition*

[2] TRIADE is an acronyme for *TRIAngular DEcompositions*

also available on the ORCCA machines. The MAPLE version of TRIADE will be made available at the beginning of March.

Section 3 specifies the guidelines for the project and, in particular, for the *preliminary mini-project*.

Section 4 describes the projects offered to the students.

2 Experimenting with AXIOM

We first show how to solve a system of equations with the AXIOM implementation of TRIADE.

```
(1) -> V := OVAR [x,y,z]

   (1)  OrderedVariableList [x,y,z]
                                                                 Type: Domain
(2) -> P := NSMP(Integer, V)

   (2)  NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z])
                                                                 Type: Domain

(3) -> T := RegularChain(Integer, [x,y,z])

   (3)  RegularChain(Integer,[x,y,z])
                                                                 Type: Domain

(4) -> lp: List P := [x^2 + y + z -1, x + y^2 +z -1, x + y +z^2 -1]

          2                  2                  2
   (4)  [x  + y + z - 1,x + y  + z - 1,x + y + z  - 1]
Type: List NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z])

(5) -> zeroSetSplit(lp)$T

           2
   (5)  [{z  + 2z - 1,y - z,x - z},{z,y - 1,x},{z,y,x - 1},{z - 1,y,x}]
                                      Type: List RegularChain(Integer,[x,y,z])
(6) -> )set message time on
(7) -> lp: List P := [x^3 + y + z -1, x + y^3 +z -1, x + y +z^3 -1]

          3                  3                  3
   (7)  [x  + y + z - 1,x + y  + z - 1,x + y + z  - 1]
Type: List NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z])
                                                    Time: 0.01 (IN) = 0.01 sec
(8) -> zeroSetSplit(lp,true,true)$T
[1 <3,0> -> |3|; {0}]W[2 <4,0>,<2,1> -> |6|; {0}][2 <3,1>,<2,1> -> |5|; {0}]
G[2 <2,1>,<2,1> -> |4|; {0}]WW[3 <2,1>,<2,1>,<1,2> -> |5|; {0}]
Gwwww[5 <3,1>,<3,1>,<3,1>,<2,1>,<1,2> -> |12|; {0}]
GI[5 <2,2>,<3,1>,<3,1>,<2,1>,<1,2> -> |11|; {0}]
GG[5 <1,2>,<3,1>,<3,1>,<2,1>,<1,2> -> |10|; {0}]
[5 <0,3>,<3,1>,<3,1>,<2,1>,<1,2> -> |9|; {0}]
[4 <3,1>,<3,1>,<2,1>,<1,2> -> |9|; {1}]
```

```
WW[5 <2,2>,<2,2>,<3,1>,<2,1>,<1,2> -> |10|; {1}]
[5 <1,2>,<2,2>,<3,1>,<2,1>,<1,2> -> |9|; {1}]
W[5 <0,3>,<2,2>,<3,1>,<2,1>,<1,2> -> |8|; {1}]
[4 <2,2>,<3,1>,<2,1>,<1,2> -> |8|; {2}]
[4 <1,2>,<3,1>,<2,1>,<1,2> -> |7|; {2}]
WW[4 <0,3>,<3,1>,<2,1>,<1,2> -> |6|; {2}]
[3 <3,1>,<2,1>,<1,2> -> |6|; {3}]
GIWWW[3 <2,2>,<2,1>,<1,2> -> |5|; {3}]
[3 <1,2>,<2,1>,<1,2> -> |4|; {3}]
WWW[3 <0,3>,<2,1>,<1,2> -> |3|; {3}]
[2 <2,1>,<1,2> -> |3|; {4}]
WWWW[2 <1,2>,<1,2> -> |2|; {4}]
WWWW[2 <0,3>,<1,2> -> |1|; {4}][1 <1,2> -> |1|; {5}]
WWWWW[1 <0,3> -> |0|; {5}]
WwwWwWwWWwwWwWwWwWwWwWwWwwwWwWwwwwWwwwwwwwwww
   *** QCMPACK Statistics ***
      Table     size:  42
      Entries reused:  34
   *** REGSETGCD: Gcd Statistics ***
      Table     size:  6
      Entries reused:  0

   *** REGSETGCD: Inv Set Statistics ***
      Table     size:  2
      Entries reused:  0

   (8)
   [
       7    6    5     4     3     2
     {z  + z  + z  - 2z  - 2z  - 2z  - z - 1,
        2          2     5    4    3    2              2               3
      (z  + z + 1)y  + (z  + z  + z  - z  - z - 1)y + z  + z, x + y + z  - 1}
     ,
      8    7    6     5     4     3     2               3           3
    {z  + z  + z  - 2z  - 2z  - 2z  + 5z  + 5z - 3,2y + z  - 1,2x + z  - 1},
           2
    {z - 1,y  - 1,x + y}, {z,y - 1,x}, {z - 1,y,x}, {z,y,x - 1}]
                                    Type: List RegularChain(Integer,[x,y,z])
                                      Time: 0.09 (EV) + 0.01 (OT) = 0.10 sec
```

Now we show how to call the AXIOM implementation of *lextriangular*

```
V := OVAR [x,y,z]


   (1)  OrderedVariableList [x,y,z]
                                                             Type: Domain
                                                Time: 0.02 (OT) = 0.02 sec

P := NSMP(Integer, V)


   (2)  NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z])
                                                             Type: Domain
                                                Time: 0.01 (IN) = 0.01 sec
```

```
pack := LexTriangularPackage(Integer,[x,y,z])


   (3)  LexTriangularPackage(Integer,[x,y,z])
                                                              Type: Domain
                                                               Time: 0 sec

lp: List P := [x^2 + y + z -1, x + y^2 +z -1, x + y +z^2 -1]

          2                  2                  2
   (4)  [x  + y + z - 1,x + y  + z - 1,x + y + z  - 1]
Type: List NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z])
                           Time: 0.07 (IN) + 0.05 (OT) + 0.03 (GC) = 0.15 sec

lg :=  groebner(lp)$pack


                  2      2        2       2     4    2  6     4     3    2
   (5)  [x + y + z  - 1,y  - y - z  + z,2z y + z  - z ,z  - 4z  + 4z  - z ]
Type: List NewSparseMultivariatePolynomial(Integer,OrderedVariableList [x,y,z])
                           Time: 0.08 (EV) + 0.01 (OT) + 0.01 (GC) = 0.10 sec

lexTriangular(lg,false)$pack


   (6)
     2
   [{z  + 2z - 1,(2z - 1)y + 5z - 2,x + y - 2z}, {z,y,x - 1}, {z - 1,y,x},
    {z,y - 1,x}]
                                     Type: List RegularChain(Integer,[x,y,z])
                           Time: 0.08 (EV) + 0.01 (OT) + 0.01 (GC) = 0.10 sec

lexTriangular(lg,true)$pack


   (7)
     4     2                 2           2        2  2
   [{z  - 4z  + 4z - 1,2y + z  - 1,2x + z  - 1},{z ,y  - y + z,x + y - 1}]
                                     Type: List RegularChain(Integer,[x,y,z])
                                                   Time: 0.02 (EV) = 0.02 sec
```

3 Guidelines

There are two steps: one baby step, called the *preliminary mini-project* and one giant step, simply called the *project*. The goal of the *preliminary mini-project* is to get you familiar with the language of your *project* on a very simple algorithm.

3.1 The preliminary mini-project

Date of postage. Thursday 12 of February.

Due format. A porgram in AXIOM, ALDOR or MAPLE. The language must be the same as for the project.

Date for submission. Thursday 11 of March.

Description. You have the write a function that takes as input two polynomials f_1, f_2 in $\mathbb{Q}[x, y]$ and returns their common roots in a similar form as TRIADE (that is, a list of triangular sets). Assume that $x < y$ holds. This means that we want to compute univariate conditions in x rather than in y. Then, you can call an existing function for computing the resultant of f_1, f_2 w.r.t. x. Let r be the square-free part of this resultant and let $\mathbb{A}$ be the product of fields $\mathbb{Q}[x]/\langle r \rangle$. The next and last step is the computation of the gcd of f_1, f_2 regarded as univariate polynomials in $\mathbb{A}[y]$. You must implement this step yourself, without callling any gcd function from TRIADE.

3.2 The project

Date of postage. Thursday 12 of February.

Due format. A documented software in AXIOM, ALDOR or MAPLE and/or a report (depending on the nature of the project) together with a public presentation.

Date for submission. Friday 30 of April for the presentation. This date may vary by a few days if major conflicts happen. The final dead-line for submitting the software and/or report is $\min(D_{CS}, D_{AM}) - 1$ where D_{CS} and D_{AM} are the days your grades are due to the Applied Maths Department and Computer Science Department, respectively.

Project Choice. The students are proposed a large variety of projects on Thursday 12 of February. On Thursday 4 of March, each student must tell the instructor which project she/he would like to realize. The chosen projects must be pairwise different. If several students are interested by the same project a lottery will be organised on Thursday 19 of February.

4 Description of the Proposed Projects

4.1 Projects aiming to speed up the gcd computations of TRIADE

Project 1 (van Hoeij & Monagan algorithm for TRIADE **in** MAPLE) The algorithm by M. Monagan and M. van Hoeij [HM02] computes polynomial gcds over number fields by modular techniques. Its MAPLE implementation has been made available to us by M. Monagan.

A first goal of this project is to optimize the MAPLE version of TRIADE by calling Monagan's package where it applies. The second goal is to modify Monagan's package in order to compute the *Bézout coefficients* of the input polynomials. That is, to implement an *Extended Euclidean* version of the algorithm by M. Monagan and M. van Hoeij. This will also help improving the MAPLE version of TRIADE

Project 2 (van Hoeij & Monagan algorithm for TRIADE **in** AXIOM) The main concern is the same as for Project 1. The differences are the language: AXIOM rather than MAPLE and the fact there is no implementation of [HM02] in AXIOM yet. So only the gcd (without Bézout coefficients) will be required here.

Project 3 (Moreno Maza & Oancea algorithm for TRIADE **in** MAPLE) The algorithm by algorithm by M. Moreno Maza and C. Oancea [MO04] computes polynomial gcds over products of fields by modular techniques. It has been implemented in ALDOR by the authors.

A first goal of this project is to realize a MAPLE implementation of this algorithm. Then, the second goal is to optimize the MAPLE version of TRIADE by calling the algorithm [MO04] where it applies.

Project 4 (Moreno Maza & Oancea algorithm for TRIADE **in** AXIOM) The goals are the same as for Project 3. The only differences is the language: AXIOM rather than MAPLE.

4.2 Projects aiming to apply the TRIADE technology in other problems

Project 5 (Lextriangular in MAPLE) A first goal of this project is to implement in MAPLE the *lextriangular* of D. Lazard. Some of the functionalities of TRIADE can be used for this implementation. A second goal is to add an option which will guarantee that the output is a prime decomposition.

Project 6 (Computations with Algebraic Varieties based on TRIADE**)** Based on the MAPLE or AXIOM version of TRIADE, the aim is to write a package for manipulating algebraic varieties. Each algebraic variety will be encoded by a square-free triangular decomposition. To save on the cost of factorization, these triangular decompositions will not be necessarily prime decompositions. The main operations provided by the package will be $=$, $\cup$, $\cap$, $\setminus$ and the (explicit) Zariski closure of a quasi-component.

Project 7 (Optimized Pardi in MAPLE**)** The PARDI algorithm is probably the best candidate for gaining a great speed up from the algorithms [HM02] and [MO04]. A first goal of this project is to implement the *algebraic* version of PARDI (called PALGIE) as an extension package of the MAPLE version of TRIADE. The second one is to measure the speed up obtained by calling the MAPLE implementations of the algorithms [HM02] and [MO04].

4.3 Research Projects

Project 8 (Lextriangular in Positive Dimension in AXIOM**)** A first goal is to generalize the *lextriangular* algorithm such it can take as input the lexicographical Gröbner basis of any polynomial ideal, without restriction to the dimension zero case. A second goal is to implement this new algorithm in AXIOM where the dimension zero case is already available.

Project 9 (From characteristic sets to Gröbner bases) In [ALM99] the authors show how a characteristic set C (that is a particular kind of triangular set) of a prime ideal $\mathcal{P}$ of $\mathbf{k}[x_1, \ldots, x_n]$ can be obtained in $\mathcal{O}(1)$ computing time from a lexicographical Gröbner basis G of $\mathcal{P}$. The only known algorithm for reconstructing G from C is essentially the computation of a Gröbner basis of the saturated ideal of C. This process appears to be very expensive. I beleieve that, at least in dimension zero, a much cheaper algorithm exists. The goal of this project is to discover it! Tests and implementation could be realized in AXIOM or MAPLE.

Project 10 (Task Management for TRIADE **in** MAPLE**)** The top level procedure of the TRIADE algorithm is a task manager, which is meant to be implemented on a distributed environment. Indeed, this procedure is a *branch-and-cut* algorithm. This means that a running task T_1 may be aborted before it is completed because the completion of another running T_2 shows that T_1 is superfluous. Therefore, performing $\{T_1, T_2\}$ in a distributed environment will require less resources (time and space) than in a sequential one starting with T_1. The key point for a sequential implementation is to develop strategies that lead to perform T_2 before T_1.

The current implementations (AXIOM, ALDOR, MAPLE) of the TRIADE algorithm are all sequential and the goal of this project is to improve the existing strategies for avoiding superfluous tasks. The work will be on two fronts: algorithmic and experimental. On the algorithmic front, a better understanding of the task generation will be needed. From there, one hopes to refine the existing strategies. On the experimental front, one will have to measure the concrete effect of the various strategies. The experimentations will be conducted with the MAPLE version of TRIADE.

Project 11 (Distributed Computation with TRIADE in ALDOR) The main concern is the same as for Project 10. However, the experimental front is emphasized. Indeed, experimentations will be conducted with the ALDOR version of TRIADE together with the Π^{it} library.

Project 12 (El Kahoui Gap Structure Theorem) This project is mathematically deeper and would require little implementation. However, it will require a very good understanding of the notions developed in the course. Based on the paper [El 03] by M. El Kahoui, the aim is to answer the following questions

- Does the Gap Structure Theorem of M. El Kahoui lead to a practical subresultant algorithm over non-integral domains? products of fields?
- When two polynomials have a quasi-monic gcd over a product of fields can we effectively compute such gcd without splitting?

Answering these questions will probably lead to a research article.

References

[ALM99] P. Aubry, D. Lazard, and M. Moreno Maza. On the theories of triangular sets. *J. Symb. Comp.*, 28(1-2):105–124, 1999.

[BLM01] F. Boulier, F. Lemaire, and M. Moreno Maza. PARDI ! In *proceedings of ISSAC 2001*, pages 38–47, London, Ontario, 2001. ACM Press.

[DDD85] J. Della Dora, C. Discrescenzo, and D. Duval. About a new method method for computing in algebraic number fields. In *Proc. EUROCAL 85 Vol. 2*, volume 204 of *Lect. Notes in Comp. Sci.*, pages 289–290. Springer-Verlag, 1985.

[El 03] M'hammed El Kahoui. An elementary approach to subresultants theory. *J. Symb. Comp.*, 35:281–292, 2003.

[HM02] M. van Hoeij and M. Monagan. A modular gcd algorithm over number fields presented with multiple extensions. In Teo Mora, editor, *Proc. ISSAC 2002*, pages 109–116. ACM Press, July 2002.

[Laz92] D. Lazard. Solving zero-dimensional algebraic systems. *J. Symb. Comp.*, 15:117–132, 1992.

[MO04] M. Moreno Maza and C. Onacea. On polynomial gcds over products of fields presented by towers of simple extensions. Technical Report xxx, University of Western Ontario, January 2004.

[Mor99] M. Moreno Maza. On triangular decompositions of algebraic varieties. Technical Report TR 4/99, NAG Ltd, Oxford, UK, 1999. Presented at the MEGA-2000 Conference, Bath, England.