Symbolic Determinants: Calculating the Degree

Technical Report
by
Brent M. Dingle

Texas A&M University
Original: May 2004
Updated: July 2005

Abstract:

There are many methods for calculating the determinant of any matrix of numerical entries. Most of these methods can be applied to matrices with symbolic entries, however they often require significantly more time than their numerical counterparts. This is mainly due to the many hardware and compiler optimizations that are in place for numerical calculations coupled with the fact that "one" symbolic multiply often involves more than just multiplying two simple items together. Anyone involved in writing a computer algebra system is aware of this. Most are also aware that the determinant of a matrix containing polynomial entries may be calculated using an interpolation technique, and for significantly sized problems this is faster than directly calculating the symbolic determinant. One of the main difficulties in this technique is determining the degree of the polynomial determinant. While methods to determine the degree exist, most are approximations or involve convoluted techniques. In this paper we present a straightforward, easy to implement, robust method to determine such a degree. This method may be applied to any n x n matrix with univariate or multivariate polynomial entries.

1 Introduction

There has been a great deal of work done in attempting to find a fast and easy way to calculate symbolic determinants ([Gelf1994], [Henr1999], [Kapur1995], [Mano1993], [Sasa1982]). Most of this work has focused on the calculation of resultants or been associated with solving systems of multivariate polynomials. For this paper we are concerned with only those methods that attempt to use an interpolation technique [Gasc2000], to calculate the determinant of a symbolic matrix [Mano1992], [Vand2002]. More specifically we wish to offer an easy to implement, robust method to determine the degree of the polynomial determinant. One method to do this, as well as a description of why such a method is needed, can be found in [Henr1999]. However the method presented therein seems to a be rather complex and only applies to matrices involving a single variable. Others have offered methods for the multivariable case, but usually for only specific scenarios [Marc2001], [Marc2002], or they require a rather extensive understanding of higher mathematics [Olve2004], which is a reasonable expectation, but difficult to implement. It is our intent in this paper to offer a more direct method of calculating the degree of the polynomial determinant for either the univariate or multivariate case for any n x n matrix, without requiring the reader to have any more mathematical background than that required to calculate the determinant in the first place.

We begin in section 2 with a review of a method for calculating the determinant of any matrix, symbolic or numeric. We will then, in section 3, offer a simple modification of this method that will provide an upper bound for the degree of the polynomial determinant. This first method will be limited to matrices of a single variable. However, it should be noted this bound is often exact in practice. Also in section 3 we will walk through some examples of applying the algorithm. Having established this basis, in section 4, we will present the modifications necessary to make the algorithm function for a matrix of more than one variable as well as some examples using the multivariate algorithm and section 5 will wrap things up with a brief summary.

2 Improved Recursive Method for Determinant Calculation

Let us recall a method to calculate the determinant of a symbolic matrix that is often referred to as “fraction free.” This method uses both recursion and an elimination trick to assist in the calculation of the determinant. This method is based on one proposed in [Bare1968]. The concept is to create an upper triangular matrix but to also keep track of the determinant as that is being done. The algorithm goes something as follows:

```
CalcDetRecurse(input is an n x n Matrix, output is Determinant)
{
   Check for invalid conditions (not square etc)
   If number of rows = 1 then return only element, e[0][0]
   If number of rows = 2 then return e[0][0]*e[1][1] – e[0][1]*e[1][0]

   // Construct SubMatrix (with 1 less row and column than this Matrix)
   For i = 1 to (n – 1)
   {
      For j = 1 to (n – 1)
      {
```

```
            subtract_me = e[i][0] * e[0][j]
            e[i][j] = e[i][j] * e[0][0] – subtract_me
         }
      } // Submatrix is e[1][1] to e[n-1][n-1] inclusive
      Determinant = CalcDetRecurse(SubMatrix, Determinant)

      For i = 1 to num_rows – 2
      {
         Determinant = Determinant / e[0][0]
      }
}
```

Notice this algorithm runs with about $O(n^3)$ multiplications and additions. Specifically, it has $n^2 + (n\text{-}1)^2 + \ldots = (n\text{-}1)(2n^2 + 5n + 6) / 6$ multiplications. It will work with any type of matrix. However it is slightly complicated. To understand this method it is best to work through an example. We will begin with a numerical 4x4 matrix and show each submatrix created.

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 3 & 7 & 1 \\ 3 & 2 & 2 & 5 \\ 13 & 11 & 3 & 6 \end{bmatrix} \rightarrow \begin{bmatrix} -7 & -8 & -19 \\ -4 & -7 & -7 \\ -15 & -36 & -46 \end{bmatrix} \rightarrow \begin{bmatrix} 17 & -27 \\ 132 & 37 \end{bmatrix} \rightarrow 4193$$

And then as the recursion unwinds: 4193 / -7 = -599 → (-599 / 1) / 1 = -599

Now to fully realize why the method works it might be a good idea to look at a symbolic matrix and notice how things cancel and why they cancel. To demonstrate this we will offer a 4x4 matrix and reduce it to an upper triangular matrix. The notation will be as follows: the subscripts will remain the same and the superscripts will denote the iteration, the lack of a superscript means iteration 1.

So let us begin with the following matrix:

$$\begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ b_0 & b_1 & b_2 & b_3 \\ c_0 & c_1 & c_2 & c_3 \\ d_0 & d_1 & d_2 & d_3 \end{bmatrix} \rightarrow \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ 0 & b_1 - a_1\frac{b_0}{a_0} & b_2 - a_2\frac{b_0}{a_0} & b_3 - a_3\frac{b_0}{a_0} \\ 0 & c_1 - a_1\frac{c_0}{a_0} & c_2 - a_2\frac{c_0}{a_0} & c_3 - a_3\frac{c_0}{a_0} \\ 0 & d_1 - a_1\frac{d_0}{a_0} & d_2 - a_2\frac{d_0}{a_0} & d_3 - a_3\frac{d_0}{a_0} \end{bmatrix}$$

Let $b_1^2 = b_1 - a_1\frac{b_0}{a_0} = (a_0b_1 - a_1b_0) / a_0$

$b_2^2 = b_2 - a_2\frac{b_0}{a_0} = (a_0b_2 - a_2b_0) / a_0$

and so on.

Notice it is the latter form that is being used in this section's algorithm.

Continuing on:

$$\begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ 0 & b_1^2 & b_2^2 & b_3^2 \\ 0 & c_1^2 & c_2^2 & c_3^2 \\ 0 & d_1^2 & d_2^2 & d_3^2 \end{bmatrix} \rightarrow \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ 0 & b_1^2 & b_2^2 & b_3^2 \\ 0 & 0 & c_2^2 - b_2^2 \frac{c_1^2}{b_1^2} & c_3^2 - b_3^2 \frac{c_1^2}{b_1^2} \\ 0 & 0 & d_2^2 - b_2^2 \frac{d_1^2}{b_1^2} & d_3^2 - b_2^2 \frac{d_1^2}{b_1^2} \end{bmatrix}$$

Again performing a renaming and another iteration we arrive at:

$$\begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ 0 & b_1^2 & b_2^2 & b_3^2 \\ 0 & 0 & c_2^3 & c_3^3 \\ 0 & 0 & d_2^3 & d_3^3 \end{bmatrix} \rightarrow \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ 0 & b_1^2 & b_2^2 & b_3^2 \\ 0 & 0 & c_2^3 & c_3^3 \\ 0 & 0 & 0 & d_3^3 - c_3^3 \frac{d_2^3}{c_2^3} \end{bmatrix}$$

A straightforward (Gaussian) method would then calculate the determinant by multiplying: $a_0 \, {b_1}^2 \, {c_2}^3 \left(d_3^3 - c_3^3 \frac{d_2^3}{c_2^3} \right)$. However, the recursive method described in this section notices that there is automatically a large amount of cancellation. Specifically it can be shown, ${c_2}^3 \left(d_3^3 - c_3^3 \frac{d_2^3}{c_2^3} \right)$ can be evenly divided by ${b_1}^2$ exactly 3-2 = 1 time and the quotient of that division can in turn be evenly divided by a_0 exactly 4-2 = 2 times.

3 Finding the Determinant via Interpolation (univariate)

Let us now, briefly, review the method of calculating the determinant of a matrix using an interpolation trick. For this paper we will limit things to be as basic as possible. This section's method is specifically designed for a matrix that has entries which are univariate polynomials. The motivation for this method is that numeric computations can be performed much faster than symbolic computations – they have the advantage of hardware and compiler optimization techniques. So if we put a number in for the variable we will have a strictly numeric matrix for which we can quickly find the determinant.

The reasoning behind this method begins by noticing that the determinant if solved symbolically would be a univariate polynomial equation. If we can predetermine the degree of this equation to be d we can use d+1 values for the variable and calculate the

determinant *d*+1 times. This would give us *d*+1 points to use to interpolate what the univariate polynomial determinant would be. This of course assumes we have a fast way to calculate numeric determinants and a fast interpolation method, both of which can readily be found, for example in [Pres2002].

3.1 Determining the Degree of the Determinant

So the actual problem to solve is finding *d*, the degree of the resulting determinant. A method to do this is presented in [Henr1999] which in turn was based on a method proposed in [Door1979]. While the method described in those papers is effective we will present a less complex method, which may or may not be as efficient. It is based on the improved recursive method described above. The idea is as follows:

Given a symbolic *n* x *n* matrix, create a new *n* x *n* matrix where each entry e[i][j] is the highest degree of the variable appearing in the i^{th} row and j^{th} column of the original matrix. Follow the general steps of the Improved Recursive Algorithm of section 2, however just keep track of the degree of the variable that would result from the various multiplications. Notice it is possible that this highest degree may get cancelled in a subtraction, however we will assume this does not happen, and thus will arrive at a maximum bound on the degree. It should be obvious this calculation will take no more time than required to calculate one numerical determinant. In fact it should be less time as the multiplications all turn into additions. The algorithm would go something as follows, we assume the input matrix is the matrix of maximum degrees:

```
CalcDegreeOfDet(input n x n Matrix, output MaxDeg)
{
   Check for invalid conditions (not square etc)
   If number of rows = 1
      then return only element, e[0][0]

   If number of rows = 2
      then return Max(e[0][0]+e[1][1], e[0][1]+e[1][0])

   // Construct SubMatrix (with 1 less row and column than this Matrix)
   For i = 1 to (n – 1)
   {
      For j = 1 to (n – 1)
      {
         subtract_me = e[i][0] + e[0][j]
         e[i][j] = Max( e[i][j] + e[0][0], subtract_me)
      }
   }  // Submatrix is now e[1][1] to e[n-1][n-1] inclusive

   MaxDeg = CalcDetRecurse(SubMatrix, MaxDeg)

   For i = 1 to num_rows – 2
   {
      MaxDeg = MaxDeg - e[0][0]
   }

    Return MaxDeg
```

```
}
```

3.2 Examples of Degree Calculation (univariate)

A 3x3 Example:

Let $A = \begin{bmatrix} x & x^2 & 5 \\ x^3 & 7 & x \\ x^2 & x^2 & x^4 \end{bmatrix}$, then the matrix of maximum degrees is $\begin{bmatrix} 1 & 2 & 0 \\ 3 & 0 & 1 \\ 2 & 2 & 4 \end{bmatrix}$.

Applying the algorithm we get the following:

$$\begin{bmatrix} 1 & 2 & 0 \\ 3 & 0 & 1 \\ 2 & 2 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} \max(0+1,2+3) & \max(1+1,3+0) \\ \max(2+1,2+2) & \max(4+1,2+0) \end{bmatrix} \rightarrow \begin{bmatrix} 5 & 3 \\ 4 & 5 \end{bmatrix} \rightarrow \max(10, 7) = 10$$

Unwinding the recursion we subtract 1 from 10 exactly 3-2 = 1 time, for a result of $d = 9$. This is correct as the determinant is $-x^9 + 13x^5 - x^4 - 35x^2$.

A 4x4 Example:

Let $A = \begin{bmatrix} x^3 & x^2 & 5 & x \\ x^2 & x^3 & 7 & x \\ 1 & x^2 & x^2 & x^4 \\ 3x & 5 & 9x^2 & 2 \end{bmatrix}$, then the matrix of maximum degrees is $\begin{bmatrix} 3 & 2 & 0 & 1 \\ 2 & 3 & 0 & 1 \\ 0 & 2 & 2 & 4 \\ 1 & 0 & 2 & 0 \end{bmatrix}$

$$\begin{bmatrix} 3 & 2 & 0 & 1 \\ 2 & 3 & 0 & 1 \\ 0 & 2 & 2 & 4 \\ 1 & 0 & 2 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 6 & 3 & 4 \\ 5 & 5 & 7 \\ 3 & 5 & 3 \end{bmatrix} \rightarrow \begin{bmatrix} 11 & 13 \\ 11 & 9 \end{bmatrix} \rightarrow 24$$

Unwinding we see that 24 – 6 = 18, and then 18 – 3 – 3 = 12. So the degree of the determinant should be no more than 12. And in fact the degree is exactly 12 as the determinant is $-9x^{12} + 9x^{10} + 26x^8 + 2x^7 - 20x^6 - 18x^5 + 16x^4 - 10x^3 + 14x^2 - 10x$.

3.3 Calculating the Interpolation Points (univariate)

Once we have determined the degree of the determinant to be d we will need to calculate the determinant at $d + 1$ unique values. To illustrate this consider the 3x3 example where $A = \begin{bmatrix} x & x^2 & 5 \\ x^3 & 7 & x \\ x^2 & x^2 & x^4 \end{bmatrix}$. We found the degree of this determinant to be 9, so we need to evaluate the determinant for 9+1 = 10 unique values of x. For notation purposes let $|A(v_i)|$ denote the determinant of A when a value of v_i is placed in for x. Let $v_i = i$ for i = -4 to 5, thus giving us 10 values = { -4, -3, -2, -1, 0, 1, 2, 3, 4, 5 }. Using whatever fast numerical determinant method we like, we find that

$|A(v_{-4})| = 248016$
$|A(v_{-3})| = 16128$
$|A(v_{-2})| = -60$
$|A(v_{-1})| = -48$
$|A(v_0)| = 0$
$|A(v_1)| = -24$
$|A(v_2)| = -252$
$|A(v_3)| = -16920$
$|A(v_4)| = -249648$
$|A(v_5)| = -1914000$

We then use whatever fast numerical interpolation routine we like to find the polynomial that goes through the points:

(-4, 248016),	(1, -24)
(-3, 16128),	(2, -252)
(-2, -60),	(3, -16920)
(-1, -48),	(4, -249648)
(0, 0),	(5, -1914000)

And arrive at the correct answer of: $-x^9 + 13x^5 - x^4 - 35x^2$.

4 Modifications Needed for the Multivariate Case

The most direct method to apply the above univariate algorithm to a multivariate matrix would be to somehow reduce the multivariate case to a univariate case. Assuming this is not possible, or too complicated, we will need to modify the above algorithm to accommodate more than one variable being present in the original matrix.

4.1 The Easy Modification

The easiest way to modify the above method to work with the multivariate case is simply to consider each variable separately from the others and apply the univariate method multiple times. Specifically if there were two variables present in the matrix, say x and y, then the above method to calculate the degree bounds on the variable would be applied

two times, once for the x and once for the y. This would give us the maximum degree of x, d_x, and the maximum degree of y, d_y, that could occur in the determinant. Note however that it would not tell us if the monomial term $x^{d_x} y^{d_y}$ appears in the determinant. For safety we would have to assume it does, as well as all the other possible terms. Thus we would need at most $(d_x + 1)*(d_y + 1)$ interpolation points. This number of points is maximum by the argument that x and y can be considered to be perpendicular directions in space (like the axes). Thus if we need $d_x + 1$ points to interpolate in the x direction and $d_y + 1$ points to interpolate in the y direction we will need $(d_x + 1)*(d_y + 1)$ to interpolate across the xy-plane. This is analogous to the case of spline patches, particularly Bezier patches on (0,1) x (0,1). Another specific case demonstrating this as a maximum bound can be found in [Marco2002] where they are using l, m, and n as the maximum degrees for x, y, and z and concluded $(l + 1)*(m + 1)*(n + 1)$ interpolation points would be required.

Once we have a bound for the degree of each variable we will be able to place values into the original matrix to obtain one of strictly numerical values, of which we can "quickly" calculate the determinant. Thus we will obtain the necessary number of points on which to interpolate. And we can then apply whatever technique we desire to perform the interpolation. This will not necessarily be easy to implement. However this paper is dedicated only to finding the degree of the determinant, so details of the interpolation will be left for another time. A good summary of such methods can be found in [Gasc2000].

4.2 A Two Variable Example

Let $A = \begin{bmatrix} x+1 & 0 & 3y^2 \\ 1 & x & xy \\ 0 & x^2+1 & 2y \end{bmatrix}$.

Applying our method above for x we get:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 2 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} \max(1+1,0+0) & \max(1+1,0+0) \\ \max(2+1,0+0) & \max(0+1,0+0) \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 2 \\ 3 & 1 \end{bmatrix} \rightarrow 5$$

Unwinding we get 5 – 1 = 4. So the maximum degree on the x is 4.

Applying our method for y we get:

$$\begin{bmatrix} 0 & 0 & 2 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 2 \\ 0 & 2 \end{bmatrix} \rightarrow 2$$

Unwinding we get 2 – 0 = 2. So the maximum degree on the y is 2.

Both of these upper bounds are exact as the determinant can be found by hand to be: $-x^4y - x^3y + 3x^2y^2 + x^2y + xy + 3y^2$.

4.3 A Three Variable Example

Let $A = \begin{bmatrix} x+1 & x^2z & 3y \\ 7 & x^3+y & xy \\ y^2+y+1 & x & x^2+2y \end{bmatrix}$.

Applying our method for x:

$$\begin{bmatrix} 1 & 2 & 0 \\ 0 & 3 & 1 \\ 0 & 1 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 4 & 2 \\ 2 & 3 \end{bmatrix} \rightarrow 7$$

Unwinding we get 7 – 1 = 6. So the maximum degree of x is 6.

Applying the method for y:

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 2 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix} \rightarrow 4$$

Unwinding we get 4 – 0 = 4. So the maximum degree of y is 4.

Applying our method for z:

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \rightarrow 1$$

Unwinding we get 1 – 0 = 1. So the maximum degree of z is 1.

These bounds are also exact as the determinant of this matrix can be calculated to be:

$$x^6 + x^5 + 2x^4y - 7x^4z + x^3y^3z - 3x^3y^3 + x^3y^2z - 3x^3y^2 + x^3yz - x^3y - 14x^2yz + 2xy^2 + 21xy - 3y^4 - 3y^3 - y^2$$

5 Summary

So we have now presented an easily implemented, robust method of determining an upper bound of the degree of the determinant of any symbolic n x n matrix. We have also provided some examples illustrating how the method works.

While the above technique only offers an upper bound for the degree of the variables of the determinant, it often gives exact bounds and could be modified to always give exact bounds. This would be accomplished by including information about the coefficients of the highest degree exponents and watching for cancellations. If a cancellation of terms

occurs the algorithm would then need to select the “second highest” degree and continue from there. For example in the univariate case, the inner for loop might be changed from:

```
subtract_me = e[i][0] + e[0][j]
e[i][j] = Max( e[i][j] + e[0][0], subtract_me)
```

to something like:

```
subtract_me_coeff = c[i][0] * c[0][j]
other_coeff = c[i][j] * c[0][0]
if (other_coeff – subtract_me_coeff != 0) then
   subtract_me = e[i][0] + e[0][j]
   e[i][j] = Max( e[i][j] + e[0][0], subtract_me)
else
   pick from “second highest” possibilities
```

While this makes the algorithm more complicated it would yield tighter bounds. However the method as presented is still sufficient to work in any case and would serve as a starting point for any similar degree calculations.

It is noted that the multivariate case is not necessarily as useful as the univariate case. This could easily be rectified by discovering a way to simplify the multivariate case into a univariate case, but that would be another paper for another time.

Bibliography

[Bare1968] Erwin H. Bareiss, “Sylvester's Identity and Multistep Integer-Preserving Gaussian Elimination,” *Mathematical Computation* 22, 103, pp. 565 – 578, 1968.

[Dodg1867] C. L. Dodgson, *An Elementary Treatise on Determinants, with Their Application to Simultaneous Linear Equations and Algebraical Geometry.* London: Macmillan, 1867.

[Door1979] P. M. Van Dooren, P. Dewilde and J. Vandewalle, “On the Determination of the Smith-MacMillan Form of a Rational Matrix From Its Laurent Expansion,” *IEEE Transactions on Circuits and Systems*, Vol. 26, No. 3, pp. 180-189, 1979.

[Gasc2000] Mariano Gasca and Thomas Sauer, “Polynomial interpolation in several variables,” *Advances in Computational Math*. Vol. 12 pp. 377 – 410, 2000.

[Gelf1994] I.M. Gelfand, M.M. Kapranov and A.V. Zelvinsky. Discrimants, Resultants and Multidimensional Determinants. Birkhauser, Boston-Basel-Berlin, 1994.

[Gent1973] W. M. Gentleman and S. C. Johnson, “Analysis of algorithms, a case study: Determinants of polynomials,” *Proceedings of the fifth annual ACM symposium on Theory of computing*, ACM Press, pp. 135-141, 1973.

[Henr1999] D. Henrion and M. Sebek, "Improved Polynomial Matrix Determinant Computation." *IEEE Trans. on CAS - Pt I. Fundamental Theory and Applications*, Vol. 46, No. 10, pp. 1307-1308, October 1999.

[Kapur1995] Deepak Kapur and Tushar Saxena, "Comparison of Various Multivariate Resultant Formulations," International Symposium on Symbolic and Algebraic Computation (ISSAC), Concordia University, Montreal, Canada July 1995.

[Mano1992] Dinesh Manocha and John F. Canny, "Multipolynomial Resultants and Linear Algebra," *Papers from the International Symposium on Symbolic and Algebraic Computation (ISSAC),* ACM Press, Berkeley, California, USA, 1992.

[Mano1993] Dinesh Manocha and John F. Canny, "Multipolynomial Resultant Algorithms," *Journal of Symbolic Computing*, Vol. 15, No. 2, pp. 99 – 122, 1993.

[Marc2001] A. Marco and J.J. Martinez, "Using polynomial interpolation for implicitizing algebraic curves," *Computer Aided Geometric Design*, Vol. 18, pp. 309-319, 2001.

[Marc2002] A. Marco and J.J. Martinez, "Implicitization of rational surfaces by means of polynomial interpolation," *Computer Aided Geometric Design*, Vol. 19, pp. 327-344, 2002.

[Olve2004] Peter J. Olver, "On Multivariate Interpolation," Unpublished, available at http://www.math.umn.edu/~olver, April 2004.

[Pres2002] William H. Press (Editor), Saul A. Teukolsky (Editor), William T. Vetterling (Editor), Brian P. Flannery (Editor), Numerical Recipes in C++ 2nd edition, Cambridge University Press, February 2002.

[Sasa1982] Tateaki Sasaki and Hirokazu Murao, "Efficient Gaussian Elimination Method for Symbolic Determinants and Linear Systems," *ACM Transactions, Mathematical Software*, Vol. 8, No. 3, pp. 277 – 289, 1982.

[Vand2002] S. Vandebril, M. Van Barel, O. Ruatta, B. Mourrain, "A new algorithm for solving multivariate polynomial problems by means of interpolation." Report TW 343, Department of Computer Science, K.U.Leuven, 2002, available at: http://www.cs.kuleuven.ac.be/publicaties/rapporten/tw/TW343.abs.html