

## Relax But Don't Be Too Lazy

*Joris van der Hoeven*

Laboratoire de Mathématique, Université Paris-Sud

January 24, 2000

*Summary by Paul Zimmermann*

Joris van der Hoeven's talk presents novel algorithms operating on formal power series. These new algorithms are based on fast multiplication methods (Karatsuba, Toom–Cook, FFT), and improve the best asymptotic complexities known, for example those obtained by Brent and Kung [1], while staying very efficient for the medium range (Karatsuba).

Most algorithms work with a linear space in the input size  $n$ , some of them require a space in  $n \log n$ . The basic idea of these new algorithms is what Joris van der Hoeven calls “the relaxed approach,” intermediate between the zealous approach and the lazy approach. This relaxed approach was invented in 1997, with the presentation of two relaxed algorithms for the multiplication of formal power series at the ISSAC'97 conference [8]. The report [9] details these algorithms and their implantation, presents some other multiplication algorithms, shows how the relaxed approach extends naturally to other operations on formal power series, and finally offers several experimental comparisons between classical and relaxed algorithms.

### 1. The Zealous Approach

Let us consider the product of two formal power series,  $f = f_0 + \dots + f_n z^n$  and  $g = g_0 + \dots + g_n z^n$ . The zealous approach consists in using *at the same time* every data  $f_0, \dots, f_n, g_0, \dots, g_n$  to calculate the product  $h = f \cdot g = h_0 + \dots + h_n z^n + O(z^{n+1})$ . So it corresponds to the classical or “off-line” approach. Several algorithms of different complexities implement this approach: the naïve multiplication in  $O(n^2)$ , Karatsuba's algorithm in  $O(n^{\log_2 3})$  [6], and the multiplication by FFT in  $O(n \log n \log \log n)$ . The following table summarizes the complexity in time and space of the best known zealous algorithms for different operations on formal power series (to facilitate the reading, we omitted the  $O(\cdot)$  terms):

Algorithm	Time	Space
Multiplication	$M(n) = n \log n \log \log n$	$n$
Division	$M(n)$	$n$
Differential equations	$M(n)$	$n$
Holonomic functions	$n$	$n$
Algebraic composition	$M(n) \log n$	$n$
General composition	$M(n) \sqrt{n \log n}$	$n \log n$
Composition in finite characteristic	$M(n) \log n$	$n$

*Newton's method.* Newton's method reduces several operations to elementary computations. For example, the logarithm of a formal power series is written:

$$\log f = \log f_0 + \int \frac{f'}{f}$$

and reduces to a division ( $f'/f$ ) and an integration of linear complexity, whence a cost in  $O(M(n))$ . Exponentiation reduces to logarithm by Newton's method. If  $g$  is such that  $\log g - f = O(z^{n/2})$ , i.e.,  $g$  is an approximation to order  $n/2$  of  $\exp f$ , then  $\tilde{g} = g - g(\log g - f)$  will be an approximation to order  $n$ , whence an algorithm again in  $O(M(n))$ . Functional inversion—given a series  $f$ , find  $g$  such that  $f \circ g = z$ —reduces to composition by:

$$\tilde{g} = g - \frac{f \circ g - z}{f' \circ g},$$

and so the complexity of inversion is that of composition.

*Polynomial composition.* The problem is as follows: given a polynomial  $f$  of degree  $p$ , a polynomial  $g$  with zero constant coefficient and of fixed degree  $q$ , and an integer  $n \geq p$ , compute  $h = f \circ g$  to order  $n$ . The divide-and-conquer algorithm consists in writing:

$$f \circ g = (f_{\text{lo}} + z^{p/2} f_{\text{hi}}) \circ g = f_{\text{lo}} \circ g + g^{p/2} (f_{\text{hi}} \circ g),$$

and so on with  $p/4$ ,  $p/8$ ,  $\dots$ , the powers of  $g$  being precomputed. It gives a complexity of  $O((pq/n)M(n) \log n)$ .

*General composition.* Given two formal power series  $f = f_0 + \dots + f_n z^n$  and  $g = g_1 z + \dots + g_n z^n$ , we want to compute  $h = f \circ g = h_0 + \dots + h_n z^n + O(z^{n+1})$ . Brent and Kung's algorithm [1] splits  $gf$  into two parts  $g = g_{\text{lo}} + g_{\text{hi}}$ :

$$\begin{aligned} g_{\text{lo}} &= g_1 z + \dots + g_q z^q \\ g_{\text{hi}} &= g_{q+1} z^{q+1} + \dots + g_n z^n, \end{aligned}$$

then writes the Taylor expansion of  $f \circ (g_{\text{lo}} + s)$  at  $s = 0$ :

$$f \circ g = f \circ g_{\text{lo}} + (f' \circ g_{\text{lo}}) g_{\text{hi}} + \frac{1}{2} (f'' \circ g_{\text{lo}}) (g_{\text{hi}})^2 + \dots$$

The computation of  $f^{(n)} \circ g_{\text{lo}}$  can be done by direct iteration:

$$f^{(i)} \circ g_{\text{lo}} = \frac{(f^{(i-1)} \circ g_{\text{lo}})'}{g'_{\text{lo}}}$$

or inverse iteration:

$$\frac{1}{(i-1)!} f^{(i-1)} \circ g_{\text{lo}} = f_{i-1} + i \int \left( \frac{1}{i!} f^{(i)} \circ g_{\text{lo}} \right) g'_{\text{lo}}.$$

## 2. The Lazy Approach

Here, we regard the formal power series not as a list of coefficients given once and for all, but as a flow of coefficients. That corresponds to “in-line” computations. The lazy approach consists in calculating the coefficients one by one; at each stage, we only perform strictly necessary computations.

Let us consider for example the equation for the generating function  $f(z)$  of binary trees counted according to their internal nodes:

$$f = 1 + zf^2.$$

Here the zealous or “off-line” approach does not apply because the coefficient  $f_n$  of order  $n \geq 1$  of  $f$  depends on  $f_0, f_1, \dots, f_{n-1}$ :

$$f_n = f_0f_{n-1} + f_1f_{n-2} + \dots + f_{n-2}f_1 + f_{n-1}f_0.$$

Thus, for the multiplication at order 3 of  $f = f_0 + f_1x + f_2x^2 + O(x^3)$  by  $g = g_0 + g_1x + g_2x^2 + O(x^3)$  giving  $h = f \cdot g = h_0 + h_1x + h_2x^2 + O(x^3)$ , the lazy approach consists in calculating the value  $h_0 = f_0g_0$  at stage 0, then  $h_1 = f_0g_1 + f_1g_0$  at stage 1 and  $h_2 = f_0g_2 + f_1g_1 + f_2g_0$  at stage 2, for a total of 6 multiplications. It is also possible to represent this computation graphically by the following table, where the value  $k$  at the intersection of line  $g_i$  and column  $f_j$  means that the value  $f_jg_i$  is obtained at stage  $k$ :

$g_2$	2		
$g_1$	1	2	
$g_0$	0	1	2
$\times$	$f_0$	$f_1$	$f_2$

The major disadvantage of this approach is that the computation of all coefficients up to order  $n$  costs  $O(n^2)$ : we cannot use fast multiplication algorithms to reduce the complexity.<sup>1</sup>

Another example is the computation of the exponential  $g = \exp f$  of a formal power series. By differentiation, we obtain  $g' = g \cdot f'$ , which reduces the exponentiation to a multiplication (the differentiation and the integration having linear complexity):

$$g = \int f'g.$$

However, here again, the series  $g$  appears in both members of the equation; with the lazy approach, we can calculate the product  $f'g$  one term at a time only, here again giving a quadratic complexity.

The article [10] by Stephen Watt describes an implementation in Scratchpad II (former name of Axiom) of that approach, based on a lazy implementation of formal power series.

In conclusion, the lazy approach has the advantage on the zealous approach to apply to the case of implicit equations; in return it does not allow the use of fast multiplication algorithms, and therefore gives higher asymptotic complexities. It is precisely this drawback which the relaxed approach solves.

### 3. The Relaxed Approach

The relaxed approach tries to use fast algorithms from the zealous approach in cases where this approach is not applicable, i.e., when “off-line” computations are not possible, like for example for the computation of the coefficients of the generating function of binaries trees  $f = 1 + zf^2$ , or of the exponential of a series  $g = \int f'g$ .

The basic idea is the following: instead of performing the minimal computations at each stage as in the lazy approach, one performs a few more calculations at certain stages, which will allow the use of fast algorithms, and in the end a global gain. As all operations considered ultimately reduce to multiplications, it is enough to detail the relaxed approach for the multiplication of formal power series.

---

<sup>1</sup>By the way, this method is precisely that used in the COMBSTRUCT library for the enumeration of combinational structures.

Let us recall the above-mentioned example of the product of  $f = f_0 + f_1x + f_2x^2 + O(x^3)$  by  $g = g_0 + g_1x + g_2x^2 + O(x^3)$ . The relaxed algorithm operates in the following way: at stage 1, instead of calculating  $h_1$  by  $f_0g_1 + f_1g_0$ , we obtain it by Karatsuba's formula  $(f_0 + f_1)(g_0 + g_1) - f_0g_0 - f_1g_1$ , thus with two multiplications as well because  $f_0g_0 = h_0$  has already been calculated. However, this already made it possible to compute part of  $h_2$ , namely  $f_1g_1$ . Then stage 2 has to compute  $f_0g_2$  and  $f_2g_0$  only, thus a gain of one multiplication compared to the lazy approach. The corresponding table is the following:

$g_2$	2		
$g_1$	1	1	
$g_0$	0	1	2
$\times$	$f_0$	$f_1$	$f_2$

where the square formed by the '0' and three '1' is obtained in three multiplications instead of four, thanks to Karatsuba's algorithm. Considering differently, we cut out the triangle of side 3 in two squares  $1 \times 1$  and a square  $2 \times 2$ , for which we used a fast algorithm. More generally, any relaxed algorithm for the multiplication of formal power series of order  $n$  consists of a tiling of the triangle of side  $n$  by a set of squares. With each tiling corresponds a new algorithm. Each square is numbered by an integer from 0 to  $n$ , indicating the stage at which it is calculated; at stage  $n$ , only the coefficients of order less than or equal to  $n$  can be used.

The example above illustrates two significant points of relaxed algorithms:

1. at the end of stage 1, it is necessary to save the value of  $f_1g_1$  which was computed in advance, for latter use at stage 2. The relaxed algorithms may thus require more memory than zealous algorithms. In most cases however, the memory used remains linear, but it can be in  $n \log n$ ;
2. if we want to continue the calculation of  $h = fg$  to a higher order, say order 4, the adopted strategy is not necessarily the best. Indeed, at stage 2 we could have calculated  $(f_0 + f_2)(g_0 + g_2) - f_0g_0 - f_2g_2$  in two multiplications, which would give  $f_0g_2 + f_2g_0$  in two multiplications as well, but would also give the term  $f_2g_2$  of  $h_4$ .

Thus we can distinguish two cases: (i) the case where the maximum order  $n$  of calculations is known in advance and thus it is a question of optimizing the total number of operations up to this order  $n$ ; (ii) the case where the maximum order is not known *a priori*, and one wants to optimize the "average" number of operations of the relaxed algorithm.

Joris van der Hoeven also shows that Karatsuba's algorithm for the multiplication of polynomials—we do not speak any more of formal power series here—is essentially relaxed, i.e., the formula giving the term  $h_k$  of the product only depends on  $f_0, \dots, f_k$  and  $g_0, \dots, g_k$ . Consequently, Karatsuba's algorithm can directly be used for the relaxed multiplication. The table corresponding to the product of two polynomials of degree 3 is the following:<sup>2</sup>

$g_3$	3	3	3	3
$g_2$	2	3	2	3
$g_1$	1	1	3	3
$g_0$	0	1	2	3
$\times$	$f_0$	$f_1$	$f_2$	$f_3$

<sup>2</sup>Exercise: Find the operations carried out with each stage from 0 to 6 and check that one indeed performs 9 multiplications. Help:  $9 = 1 + 2 + 2 + 3 + 1 + 0 + 0$ .

The major disadvantage of the relaxed alternative of Karatsuba’s algorithm is however the memory usage: on the one hand the memory required is in  $O(n \log n)$ , on the other hand the memory management is extremely complex, since for each stage it is necessary to know which values must be calculated, which should be reused—and among those, which can be destroyed—, finally which have to be saved for latter use.

Another algorithm proposed by Joris van der Hoeven consists in tiling the square  $n \times n$  by a sequence of ‘L’ shapes of increasing width. That leads to a relaxed multiplication in  $O(M(n) \log n)$ . Several other alternatives are proposed in [9], both for complete products (polynomials) and truncated products (formal power series). The other operations (division, composition) are also “essentially relaxed.” Finally we obtain the following complexities for the relaxed alternatives of the operations on formal power series:

Algorithm	Times	Space
Karatsuba’s multiplication	$n^{\log_2 3}$	$n \log n$
Multiplication via FFT	$D(n) = M(n) \log n$	$n$
Division	$D(n)$	$n$
Differential equations	$D(n)$	$n$
Holonomic functions	$n$	$n$
Algebraic composition	$D(n) \log n$	$n$
General composition	$D(n) \sqrt{n \log n}$	$n^{3/2} \log n$
Composition in finished characteristic	$D(n) \log n$	$n \log n$

The time complexities are the same ones as for the zealous approach, while replacing  $M(n)$  with  $D(n)$ . The memory complexity is identical, except when we use Karatsuba’s multiplication algorithm (there is however a slower variant by a constant factor, but in space  $O(n)$ ), or for the composition (general or in finite characteristic).

Joris van der Hoeven gives in his report [9] many experimental results for these new algorithms. Timings below correspond to an AMD processor at 200 MHz with 64 MB of main memory. Van der Hoeven’s program calculates 500 terms of the Taylor expansion of  $\exp(z \exp z)$  in 342 seconds against 1086 seconds for the zealous approach; it calculates the number of alcohols  $C_n H_{2n+1} OH$  for  $n = 5000$  in approximately 2300 seconds, whereas the naïve method does not allow this calculation in reasonable time and space; it calculates the expansion in  $1/x$  of the differential-difference equation

$$f(x) = \frac{1}{x} (1 + f(x+1) + f'(x)^2)$$

to order 2000 in 1572 seconds.

#### 4. Conclusion

Joris van der Hoeven presented us a whole panoply of algorithms which reduce the calculation of the first  $n$  coefficients of the majority of the formal power series defined by algebraic equations, differential equations or difference equations, to a quasi-linear complexity, whereas the best algorithms known before were almost quadratic (in the implicit case, i.e, where the zealous approach does not apply).

It would be nice if these algorithms were implemented in enumerative combinatorics softwares like COMBSTRUCT<sup>3</sup> or CS [3]. More generally, all computer algebra systems worthy of the name should implement these new algorithms, both for formal power series, polynomials, and integers. Indeed,

<sup>3</sup><http://algo.inria.fr/libraries/software.html>

one of the by-products of Joris van der Hoeven's report is a division algorithm with remainder in  $K(n)$  operations, whereas the best known algorithm was in  $2K(n)$  [7, 2, 5].

*Related work.* For truncated division and square root, new algorithms based on Karatsuba's multiplication are detailed in the report [4].

*Acknowledgement.* People who don't read French may thank Gina Pierrelée-Grisvard who helped to translate this summary.

### Bibliography

- [1] Brent (Richard P.) and Kung (H. T.). –  $O((n \log n)^{3/2})$  algorithms for composition and reversion of power series. In Traub (J. F.) (editor), *Analytic computational complexity (Proc. Sympos., Carnegie-Mellon Univ., Pittsburgh, Pa., 1975)*. pp. 217–225. – Academic Press, New York, 1976.
- [2] Burnikel (Christoph) and Ziegler (Joachim). – *Fast recursive division*. – Research Report n° MPI-I-98-1-022, Max-Planck-Institut für Informatik, Saarbrücken, Germany, October 1998.
- [3] Denise (Alain), Dutour (Isabelle), and Zimmermann (Paul). – CS: a MuPAD package for counting and randomly generating combinatorial structures. In *Formal Power Series and Algebraic Combinatorics*, pp. 195–204. – 1998. Proceedings of FPSAC'98, June 1998, Toronto. Software Demonstration.
- [4] Hanrot (Guillaume), Quercia (Michel), and Zimmermann (Paul). – *Speeding up the division and square root of power series*. – Research Report n° 3973, Institut National de Recherche en Informatique et en Automatique, July 2000. Available from <http://www.inria.fr/RRRT/RR-3973.html>.
- [5] Jebelean (Tudor). – Practical integer division with Karatsuba complexity. In Küchlin (Wolfgang W.) (editor), *ISSAC'97 (July 21–23, 1997. Maui, Hawaii, USA)*. pp. 339–341. – ACM Press, New York, 1997. Conference proceedings.
- [6] Karatsuba (A. A.) and Ofman (Yu. P.). – Multiplication of multidigit numbers by automata. *Physics Doklady*, vol. 7, 1963, pp. 595–596. – Translated from *Doklady Akad. Nauk*, vol. 145, n° 2, 1962, pp. 293–294.
- [7] Moenck (R.) and Borodin (A.). – Fast modular transforms via division. In *Proceedings of the 13th Annual IEEE Symposium on Switching and Automata Theory*, pp. 90–96. – October 1972.
- [8] van der Hoeven (Joris). – Lazy multiplication of formal power series. In Küchlin (Wolfgang W.) (editor), *ISSAC'97 (July 21–23, 1997. Maui, Hawaii, USA)*. pp. 17–20. – ACM Press, New York, 1997. Conference proceedings.
- [9] van der Hoeven (Joris). – *Relax, but don't be too lazy*. – Technical Report n° 78, Université de Paris-Sud, Mathématiques, Bâtiment 425, F-91405 Orsay, 1999. Submitted to the Journal of Symbolic Computation. Available from <http://www.math.u-psud.fr/~vdhoeven/>.
- [10] Watt (Stephen M.). – A fixed point method for power series computation. In Gianni (Patrizia M.) (editor), *Symbolic and Algebraic Computation (International Symposium ISSAC'88, Rome, Italy, July 4–8, 1988). Lecture Notes in Computer Science*, vol. 358, pp. 206–217. – Springer Verlag, 1989. Conference proceedings.