A non-symmetric divide-and-conquer recursive formula for the convolution of polynomials and power series

Thomas Baruchel[†]

[†]Éducation nationale, France
`baruchel@riseup.net`

November 2019

Abstract

Some changes in a recent convolution formula are performed here in order to clean it up by using more conventional notations and by making use of more referrenced and documented components (namely Sierpiński's polynomials, the Thue-Morse sequence, the binomial modulo 2 transform and its inverse). Several variants are published here, by reading afterwards summed coefficients in another order; the last formula is then turned back from a summation to a new divide-and-conquer recursive formula.

1 Introduction

In a recently published paper, a new convolution formula was written down by tracking all terms along a recursion tree built from a variant of Karatsuba's well-known algorithm [2]. While several variants of the formula were already given there, some unusual notations were heavily used in order to "pack" the required terms into a single summation. Despite the conciseness of these formulas, one could thus argue that they may be too complicated to stand as an inspiring starting point for ensuing researches.

Small changes in one of them can however lead to another more explicit variant, by noticing that three different arbitrary symbols are actually related to Sierpiński's polynomials. The coefficients of these polynomials are those from the well-documented Sierpiński triangle, and it may be expected that publishing a new simpler formula relying on such polynomials for something as significant as multiplicating two polynomials (or convoluting two power series) could have some benefits. In Sections 2 to Section 4, some elementary properties of Sierpiński's polynomials are used in order to prove two new variants as Theorems 4.1 and 4.2, one for polynomials and one for infinite power series.

1

Rewriting now Theorem 4.2 by summing all embedded coefficients "vertically" rather than "horizontally" in Section 5 allows to take benefit of a little-studied transform called the "binomial modulo 2 transform" (and its inverse) in order to achieve a still more compact variant. The resulting formula in theorem 5.4 being, like all these variants, a summation, it is then turned into to a new *divide-and-conquer* recursive formula in section 6.

This final recursive formula is said to be *non-symmetric* because both convolved power series are not handled in the same way, one being handled by performing subtractions and the other by performing additions.

2 Sierpiński's polynomials

The Sierpiński triangle is best known as a graphical figure (see below); it is a fractal object built by adding at each iteration two new copies of the same whole object at its own bottom (and scaling down the whole figure in order to keep its original size). Thus iterating over its rows from top to bottom is possible: the nth row (counting from the top) is always the same whatever the number of previous iterations is. Iterating over the rows of the triangle and reading
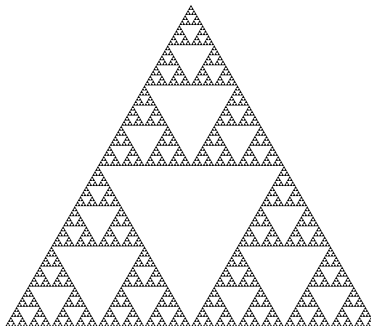
them as finite sequences of binary digits (0 for "white" and 1 for "black") gives another mathematical object which is also refferred to as the Sierpiński triangle but now in some combinatorial context. Such coefficients are those from Pascal's triangle *modulo* 2.

The sequential rows may also be read as polynomials by taking the previously described 0 and 1 as coefficients, resulting in the sequence S of Sierpiński's polynomials defined among the comments of the sequence A047999 in the *On-Line Encyclopedia of Integer Sequences* [1]. The initial polynomials are:

$$1,\, 1 + x,\, 1 + x^2,\, 1 + x + x^2 + x^3,\, \ldots$$

The power series expanding $(1 - x)^{-1}S_k^{-1}$ will be referred to as δ_k in the current paper (the symbol δ being chosen because δ_k happens to be the kth main *diagonal* in the Sierpiński triangle).

All formulas below rely on the *termwise* product of sequences (or power series), which will be denoted here as $f \odot g$ (meaning that coefficients of the same rank are multiplicated with no convolution).

Lemma 2.1. *Let $m = (d_{n-1} \ldots d_2 d_1 d_0)_2$ some nonnegative integer between 0 and $2^n - 1$ having the finite sequence d as the digits of its binary encoding; then*

$$S_m = (1+x)^{d_0} \left(1+x^2\right)^{d_1} \left(1+x^4\right)^{d_2} \ldots \left(1+x^{2^{n-1}}\right)^{d_{n-1}}.$$

Proof. This is obviously true for $m = 0$ and $m = 1$. Then, we refer to the building rule described as a comment of the sequence `A047999`:

$$\begin{cases} S_{2n+1}(x) = (x+1) S_n\left(x^2\right) \\ S_{2n}(x) = S_n\left(x^2\right) \end{cases}$$

in order to prove by induction that if the lemma is true for any m smaller than some power of 2, it is also true for any value of m smaller than the following power of 2. The proof is straightforward since the binary encoding of $2n$ is known to be the same as the one of n shifted to the left by one digit (thus performing $d_{j+1} \leftarrow d_j$) while replacing x by x^2 is the same as replacing

$$\left(1+x^{2^j}\right) \quad \text{by} \quad \left(1+x^{2^{j+1}}\right)$$

everywhere in the whole product above. □

Lemma 2.2. *Let n be some power of 2 and k some nonnegative integer smaller than n; then:*

$$S_k \times S_{n-1-k} = S_{n-1}.$$

Proof. The binary encoding of $n - 1$ is $(111 \ldots 111)_2$ since n is a power of 2; thus $n - 1 - k$ and k have complementary binary encodings. Thus, according to Lemma 2.1, S_k and S_{n-1-k} have complementary factors in regards to the whole product defined in that lemma, which soon leads to the above statement. □

3 The Thue-Morse sequence

Let σ be some specific encoding of the Thue-Morse sequence defined as the sequences `A106400` in the *On-Line Encyclopedia of Integer Sequences* [1], namely:

$$1, -1, -1, 1, -1, 1, 1, -1, -1, 1, 1, -1, 1, -1, -1, 1, \ldots$$

Like the Sierpiński triangle, the sequence may be built from a duplicating process: the 2^k initial coefficients are copied to their right with their sign being flipped in order to build the initial 2^{k+1} coefficients.

The notation $\bar{f}$ will be used here for representing the *termwise* product of σ and some polynomial f (or power series[1]). We can thus write: $\bar{f} = \sigma \odot f$ with the symbol $\odot$ indicating the *termwise* product of two objects.

---

[1]The same notation could obviously also be used for a sequence of numbers if needed.

Let also $\bar{S}$ be specifically the sequence of polynomials such that

$$\bar{S}_m = (1-x)^{d_0} \left(1-x^2\right)^{d_1} \left(1-x^4\right)^{d_2} \ldots \left(1-x^{2^{n-1}}\right)^{d_{n-1}}$$

with $m = (d_{n-1} \ldots d_2 d_1 d_0)_2$ some nonnegative integer between 0 and $2^n - 1$. This notation is very slightly different from the $\bar{f}$ one since $\bar{S}$ is a sequence of polynomials while f is a polynomial (or a power series), but it is easy to show that $\bar{S}_m = \overline{(S_m)} = \sigma \odot S_m$ by noticing that a coefficient in $\bar{S}_m$ will be -1 if and only if an odd number of digits occurs in the binary encoding described above, which exactly matches the corresponding term in σ since $\sigma_n = (-1)^{\mathcal{H}_n}$ (with $\mathcal{H}_n$ being the Hamming weight of n).

Lemma 3.1. *Let n be some power of 2 and k some nonnegative integer smaller than n; let also f be some polynomials in the indeterminate x; then:*

$$S_{n-1-k}\, x^k \,\odot\, \bar{S}_k\, f \;=\; \overline{S_{n-1-k}\, x^k \,\odot\, S_k\, \bar{f}}\,.$$

Proof. This is true for $k = 0$ and we want to prove by induction that when the identity is true for some k we can also write:

$$S_{n-1-k-2^j}\, x^{k+2j} \,\odot\, \left(1-x^{2^j}\right) \bar{S}_k\, f \;=\; \overline{S_{n-1-k-2^j}\, x^{k+2j} \,\odot\, \left(1+x^{2^j}\right) \overline{S_k\, \bar{f}}}$$

with 2^j some power of 2 not already present in the binary expansion of k.

The left-hand side of the previous equation means that we want to take consecutive blocks of 2^{j+1} coefficients; subtract the initial 2^j ones to the following 2^j ones; and finally cancel the second half of such blocks (cancelling half of each block being performed by removing one more factor from the *mask* S_{n-1-k}).

The right-hand side of the same equation performs the very same thing in another way: we flip the sign of coefficients in such a way that in consecutive blocks of 2^{j+1} coefficients, the 2^j initial ones will be flipped in an opposite manner than in the following 2^j ones (this comes from the building rule of the Thue-Morse sequence); then we add (rather than subtract) the two parts; then we flip back the signs of the coefficients to their initial state. $\qquad\square$

4 New variants of the convolution formulas

The general idea of a previous paper was to study a variant of Karatsuba's algorithm and "flatten" the recursion tree into a summation formula (see [2]). Before going further, the latter is rewritten with more expressive notations in order to help manipulating it.

Theorem 4.1. *Let f and g two polynomials of degree $n-1$ with n some power of 2 in the same indeterminate x; then*

$$\begin{aligned}
f \times g \;&=\; \sum_{k=0}^{n-1} \sigma_k\, S_{n-1-k} \left(S_{n-1-k}\, x^k \,\odot\, \bar{S}_k\, f \,\odot\, \bar{S}_k\, g\right) \\[2mm]
&=\; \sum_{k=0}^{n-1} \sigma_k\, S_{n-1-k} \left(S_{n-1-k}\, x^k \,\odot\, S_k\, \bar{f} \,\odot\, S_k\, \bar{g}\right)\,.
\end{aligned}$$

Proof. We copy the the formula (8) from [2] as it is typeset in the original paper despite some differences in used notations; it will be translated to the current notations afterwards:

$$A \times B = \sum_{k=0}^{n-1} \bar{f}_k \left(\sigma_k \bar{f}_k X^k \odot \dot{f}_k A \odot \dot{f}_k B \right) .$$

In the previous formula, $\dot{f}_k$ means exactly the same thing as $\bar{S}_k$ in the current paper, while $\bar{f}_k$ can be recognized here as S_{n-1-k} with the help of Lemma 2.2. Thus, the formula can now be translated as:

$$f \times g = \sum_{k=0}^{n-1} S_{n-1-k} \left(\sigma_k S_{n-1-k} x^k \odot \bar{S}_k f \odot \bar{S}_k g \right) .$$

According to Lemma 3.1, $\bar{S}_k f$ and $\bar{S}_k g$ can be replaced above by $S_k \bar{f}$ and $S_k \bar{g}$ since all flipped signs will cancel themselves when evaluating the termwise product $S_k \bar{f} \odot S_k \bar{g}$, leading to the stated formula. □

Theorem 4.2. *Let f and g be two power series in the same indeterminate x; then*

$$\begin{aligned} f * g &= \sum_{k=0}^{\infty} \sigma_k \delta_k \left(x^k \delta_k \odot \bar{S}_k f \odot \bar{S}_k g \right) \\ &= \sum_{k=0}^{\infty} \sigma_k \delta_k \left(x^k \delta_k \odot S_k \bar{f} \odot S_k \bar{g} \right) . \end{aligned}$$

Proof. Formula (9) from [2] is now taken into account. We can not refer here to Lemma 2.2 any longer for building some complementary polynomial because we do not work on a finite number n of terms, but the theory of generating function is useful for building the relevant power series, since

$$\frac{1}{1-x} \quad \text{expands to} \quad (1+x)\left(1+x^2\right)\left(1+x^4\right)\left(1+x^8\right) \ldots$$

where the required factors can easily be cancelled by a simple division: the previously defined δ_k term matches the required infinite product. □

5 The binomial modulo 2 inverse transform

In a comment of `A100735` in the *On-Line Encyclopedia of Integer Sequences* [1] are defined the "modulo 2 binomial transform" and its inverse. In the current paper, the name of this transform is slightly changed to "binomial modulo 2 transform". Both transforms are defined as:

$$\begin{aligned} B_n &= \sum_{k=0}^{n} T_{n,k} A_k && \textit{(transform)} \\ A_n &= \sum_{k=0}^{n} \sigma_{n-k} T_{n,k} B_k && \textit{(inverse transform)} \end{aligned}$$

5

for a given sequence A, where σ is the signed Thue-Morse sequence and $T_{n,k}$ is the relevant coefficient in the Sierpiński triangle. The notations $B = A'$ and $A = B^*$ will be used from now on; the same notations will be applied to power series as well.

Theorem 5.1. *The power series f being defined as $f = a_0 + a_1 x + a_2 x^2 + \ldots$, we define also $f_k = a_k + a_{k+1} x + a_{k+2} x^2 + \ldots$, and f_k^* is the power series defined from the binomial modulo 2 inverse transform of the sequence $a_k, a_{k+1}, a_{k+2}, \ldots$ (while f' is the binomial modulo 2 transform of f). Then for two power series f and g in the same indeterminate x,*

$$f * g = \sum_{k=0}^{\infty} \left(\overline{\delta_k} \odot f_k^* \odot g_k^* \right)' x^k \, .$$

Proof. In Theorem 4.2, the whole parenthesis is turned into

$$\sum_{j=0}^{\infty} \delta_j[k] \, f_j^*[k] \, g_j^*[k] \, x^{k+j}$$

where $f[k]$ is the coefficient of term of degree k in f.

Then we gather all contributions of some degree m by noticing that some previously defined term of degree $k + j$ is shifted to degree m (by performing the remaining multiplication in Theorem 4.2) if and only if $m - (k + j)$ and k have no common digit 1 in their binary encodings:

$$(f * g)[m] = \sum_{j=0}^{m} \sum_{k=0}^{m-j} T_{m-j,k} \, \sigma_k \, \delta_j[k] \, f_j^*[k] \, g_j^*[k]$$

where T is Sierpiński triangle with $T_{n,k} = 1$ if and only if $(n - k)\&k = 0$, the symbol $\&$ being the bitwise `and` operator.

This identity quickly leads to the expected theorem. $\qquad \square$

Lemma 5.2. *The power series f being defined as $f = a_0 + a_1 x + a_2 x^2 + \ldots$, we define also $f_k = a_k + a_{k+1} x + a_{k+2} x^2 + \ldots$, and f_k^* is the power series defined from the binomial modulo 2 inverse transform of the sequence $a_k, a_{k+1}, a_{k+2}, \ldots$ (with of course $f^* = f_0^*$). Then,*

$$\left(\delta_k \odot f_k^* \right) x^k = \delta_k \, x^k \odot f^* S_k \, .$$

Proof. We prove this by induction; the statement is obviously true for $k = 0$; then we assume it is true for some nonnegative integer k, and we show that it is still true for $k + n$ with n being some power of 2 greater than k. Let $g = f_k$ and $g_n = f_{k+n}$ in order to focus on a power of 2; we want to study $(\delta_{k+n} \odot g_n^*) \, x^{k+n}$ in order to match the left-hand side above.

Because of the self-similarity in the Sierpiński triangle, we can notice that $(1 + x^n) \, g^*$ and $x^n g_n^*$ share the same coefficient of degree m if $m \bmod 2n \geqslant n$.

For the same reasons, δ_k and $x^n \delta_{k+n}$ share the same coefficient of degree m if $m \bmod 2n \geqslant n$ (all other coefficients of $\delta_{k+n} x^n$ are null). Thus,

$$
\begin{aligned}
\left(\delta_{k+n} \odot f_{k+n}^* \right) x^{k+n} &= \left(\delta_{k+n} \odot g_n^* \right) x^{k+n} \\
&= \left(x^n \delta_{k+n} \odot x^n g_n^* \right) x^k \\
&= \left(x^n \delta_{k+n} \odot (1 + x^n) g^* \right) x^k \\
&= x^{k+n} \delta_{k+n} \odot (1 + x^n) x^k f_k^*
\end{aligned}
$$

where, again, the (rather restrictive) mask $x^{k+n} \delta_{k+n}$ allows to replace $x^k f_k^*$ with $f^* S_k$ (by using the initial assumption) *because all coefficients to be added and kept in the multiplication by $(1 + x^n)$ where also taken into account by the initial (less restrictive mask).* Of course $(1 + x^n) f^* S_k = f^* S_{k+n}$. □

Lemma 5.3. *Let f be some power series in the indeterminate x. Then,*

$$
\left(\delta_k \odot f \right)' x^k = \left(\left(\delta_k \odot f \right) x^k \right)' S_k .
$$

Proof. We prove this by induction; the statement is obviously true for $k = 0$; then we assume it is true for some nonnegative integer k, and we show that it is still true for $k + n$ with n being some power of 2 greater than k.

Because of the self-similarity in the Sierpiński triangle, we can notice that all coefficients of $\left(\delta_{k+n} \odot f \right)'$ in the left-hand part above can be found among the coefficients of $\left(\delta_k \odot f \right)'$ according to the following rule:

$$
\begin{cases}
\left(\delta_{k+n} \odot f \right)' [m] = \left(\delta_k \odot f \right)' [m] & \text{if } m \bmod 2n < n; \\
\left(\delta_{k+n} \odot f \right)' [m] = \left(\delta_k \odot f \right)' [m - n] & \text{if } m \bmod 2n \geqslant n;
\end{cases}
$$

which can also be written as $\left(\delta_{k+n} \odot f \right)' = \left(\delta_n \odot \left(\delta_k \odot f \right)' \right) (1 + x^n)$.

The same idea is separately followed for the right-hand part of the identity to be proved: $\left(\left(\delta_{k+n} \odot f \right) x^{k+n} \right)' S_{k+n}$ is also made of coefficients from $\left(\left(\delta_k \odot f \right) x^k \right)' S_k$ according to the very same same rule because of the self-similarity property of the Sierpiński triangle again. Thus we see that both sides of the statement match exactly. □

Theorem 5.4. *Let f and g be two power series in the same indeterminate x. Then,*

$$
f * g = \sum_{k=0}^{\infty} \left(\overline{\delta_k} \, x^k \odot f^* S_k \odot g^* S_k \right)' S_k .
$$

Proof. Theorem 5.1 is rewritten with the help of Lemma 5.2:

$$
f * g = \sum_{k=0}^{\infty} \left(\frac{\overline{\delta_k} \, x^k \odot f^* S_k \odot g^* S_k}{x^k} \right)' x^k
$$

which quickly leads to the expected statement with the help of Lemma 5.3. □

6 A recursive formula for the convolution

Summations involving Sierpiński's polynomials may be easy to convert into a recursive formula because "jumping" from S_k to S_{k+n} (with n being a power of 2 greater than k) is merely achieved by using the relation $S_{k+n} = (1+x^n)S_k$.

Furthermore, the parenthesis in Theorem 5.4 contains terms which still follow the previously studied *interleaved splitting scheme* (see Section 3 in [2]), meaning that they can be computed from another ones by splitting them into two increasingly-sparse terms. An exact definition of the scheme used below is:

$$\begin{cases} u_{\text{low}}^{(n)} & = & u \odot \delta_n \\ u_{\text{high}}^{(n)} & = & u \odot x^n \delta_n \end{cases} \quad \text{with } n \text{ being some power of 2.}$$

For traversing the recursion tree, an *ad hoc* operator will be used here in order to "pack" the summation from Theorem 5.4 into the following formula:

$$f * g = \left(\overline{f^*} \odot g^* \right)' + \overline{f^*} \underset{(1)}{\diamond} g^* \tag{1}$$

this parametrized operator symbol $\underset{(n)}{\diamond}$ being recursively defined as:

$$\begin{aligned} u \underset{(n)}{\diamond} v & = & (1+x^n) \left(x^n u_{\text{low}}^{(n)} - u_{\text{high}}^{(n)} \odot x^n v_{\text{low}}^{(n)} + v_{\text{high}}^{(n)} \right)' \\ & & + u \underset{(2n)}{\diamond} v \\ & & + (1+x^n) \left(x^n u_{\text{low}}^{(n)} - u_{\text{high}}^{(n)} \underset{(2n)}{\diamond} x^n v_{\text{low}}^{(n)} + v_{\text{high}}^{(n)} \right) \end{aligned} \tag{2}$$

(where the digits in the binary encoding of the variable k occuring in Theorem 5.4 are read from right to left by either choosing a digit 0 or a digit 1, explaining the two recursive calls).

Obiously the previously-defined operator is non-commutative. Building this formula as a non-symmetric one allows to track and preserve the signs from the $\overline{\delta_k}$ term without using an external mask (which is the case in Theorem 5.4). Because of the properties of the Thue-Morse sequence, subtractions actually embed "hidden" additions (since exactly one of both terms occuring in each subtractions previously had its sign flipped).

7 Conclusion

The new theorems 4.1 and 4.2 are equivalent to their previous versions in [2], but they now make use of more documented notations. They are therefore claimed to be of a more general interest for later researches. The binomial modulo 2 transform is used a couple of times in the *On-Line Encyclopedia of Integer Sequences*, but it does not seem to have been actually studied in published papers; thus theorems and lemmas proved in Section 5 and 6 may also provide a basis for further investigating the properties of this transform. The conciseness of formulas in Theorems 5.1 and 5.4 show that "transposing" the initial ones suit more the approach being investigated since the previous paper.

Formula (2) in Section 6 "packs" the whole computation into a rather simple statement with no need for symbols related to the Sierpiński triangle any longer, allowing to focus henceforth on the binomial modulo 2 transform only. Directly implementing this formula as code or pseudo-code was not considered at this point, because it would certainly not be very efficient as it is; it should be noticed however that while the definition of the binomial modulo 2 transform in Section 5 obviously is in $O(n^2)$, much better algorithms for computing the whole transform in $O(n \log n)$ are achievable. Once initially computed, the transform could probably be propagated and partially updated during the recursion process if attempting to efficiently implement the formula.

Conflict of Interest: The authors declare that they have no conflict of interest.

The current article is accessible on `http://export.arxiv.org/pdf/1912.00452`.

References

[1] *The On-Line Encyclopedia of Integer Sequences*, published electronically at `https://oeis.org`. Accessed 1 Nov. 2019.

[2] Thomas BARUCHEL, *Flattening Karatsuba's recursion tree into a single summation*, in *Computer Science*, ed. Springer Nature, Jan. 2020. Also available at `https://arxiv.org/abs/1902.08982`.