THE UMBRAL TRANSFER-MATRIX METHOD.
V. The Goulden-Jackson Cluster Method for Infinitely Many Mistakes

Doron ZEILBERGER [1]

Abstract: This is the fifth, and last, installment of the saga on the Umbral Transfer-Matrix method, based on Gian-Carlo Rota's seminal notion of the umbra. Here we extend the powerful Goulden-Jackson Cluster method, that enables one to compute generating functions enumerating words that avoid, as factors, any given *finite* set of "mistakes", to the case where there are infinitely many mistakes. This infinite set of mistakes should be a union of finitely many finite-parameter families, given symbolically in frequency notation. We illustrate the method by introducing a new 'toy model' for self-avoiding walks, that is much more interesting and complex than finite-memory approximations, yet much simpler than the (probably intractable) "real thing".

Required Reading

The reader is expected to be familiar with [Z1], and the classical Goulden-Jackson Cluster Method ([GJ1],[GJ2]), lucidly described in [NZ].

How Likely is a Monkey to Type YHVH?

It is a sin to utter the name of the Lord in vain, or even to have it written down, or printed, except in a holy text. Ignoring spaces, it is also bad luck to have the letters Yod, Heh, Vav, Heh consecutively. The classical Goulden-Jackson Cluster Method answers the following question. What is the probability that a random word will not contain, *consecutively*, the word YHVH?. Or for that matter, any other given word, or *finite* set of words, for example the set of words $\{YHVH, DORON, DAVID, IAN\}$.

It is immediate that the enumerating generating function is always a rational function, since the 'language' of 'clean words' is regular, and can be modeled by a finite automate. This also implies an algorithm for computing the generating function that, however, is impractical. The amazing Goulden-Jackson method is an *efficient* method for computing such generating functions.

But a naughty monkey may type YYYHHHHVVVVHHHH, which does not contain YHVH, so by the *letter* of the law it should not be punished, but by the *spirit* of the law it is committing an even graver sin. So what we really want to avoid is any factor of the form $Y^{a+1}H^{b+1}V^{c+1}H^{d+1}$, for any a, b, c, d *non-negative integers*. Of course, now we have *infinitely* many 'mistakes' to avoid, and the classical Goulden-Jackson method will take infinite time and infinite space.

$Y^{a+1}H^{b+1}V^{c+1}H^{d+1}$ is an example of what we will call *symbolic word*.

---

[1] Department of Mathematics, Temple University, Philadelphia, PA 19122, USA. `zeilberg@math.temple.edu` `http://www.math.temple.edu/~zeilberg/`. May 21, 2001. Accompanied by the Maple packages `UGJ`, `SymUGJ`, and `SiPerUGJ`, available from Zeilberger's homepage, or from `http://www.math.temple.edu/~zeilberg/utm.html`. Supported in part by the NSF.

Symbolic Words

Definition: Given a *finite* alphabet W, and a *finite* set of symbols $\{a_1, a_2, \ldots, a_s\}$, that represent generic non-negative integers, an (s-parameter) symbolic word is a creature of the form

$$w_1^{L_1} w_2^{L_2} \ldots w_r^{L_r} \quad ,$$

where for all i, $1 \leq i \leq r$, $w_i \in W$, and each L_i is an affine linear combination, with *non-negative coefficients* of the a_i. In other words, each L_i can be written as

$$L_i = c_0 + \sum_{j=1}^{s} c_j a_j \quad ,$$

where the c_j are *specific* non-negative integers, and $c_0 > 0$.

An Example of A Symbolic Word: If the alphabet is $\{1,2,3\}$ and the set of generic non-negative integers is $\{a,b,c\}$, then $1^{a+1}2^{a+b+1}3^{b+c+1}1^{c+1}$ is an example of a symbolic word. Its embodiment is the (triply) *infinite* set of mistakes

$$\{1^{a+1}2^{a+b+1}3^{b+c+1}1^{c+1} | a,b,c \geq 0\}$$

i.e. $\{1231, 112231, 11122231, \ldots, 122331, \ldots\}$.

A plain word is also symbolic, since it can be viewed as a 0-parameter symbolic word. For example the word 3112 is written $3^1 1^2 2^1$.

The GOAL

Given a finite alphabet and a finite set of symbolic words, to be called *mistakes*, in the letters of the alphabet, compute the generating function

$$f(t) := \sum_{n=0} b_n t^n \quad ,$$

where b_n is the number of n-letter words in the alphabet that do not contain, as *factors*, any instance of any of these (symbolic) mistakes.

What do we mean by compute? Since $f(t)$ is not going to be a rational function, there is no *a priori* reason to expect that $f(t)$ can be represented *explicitly*, whatever that means.

So we have to be content to find some kind of *equation*: algebraic, differential, functional, integral, or what have you (or even of mixed type), that would be satisfied by $f(t)$, and that would enable computing the first n terms of $f(t)$ in time (and space) polynomial in n. I confess that I can't do it, and I am almost sure that it can't be done.

Failing this, something almost as good can be achieved. I will show that there exist finitely many *catalytic variables* (corresponding to the set of generic integers), $x_1, x_2, \ldots, x_s$, and a finite set of *generating functions* (corresponding to the symbolic mistakes) $F_1, \ldots, F_r$ that depend on t and $x_1, \ldots, x_s$, such that *collectively*

$$F_1(t; x_1, \ldots, x_s) \quad , \quad F_2(t; x_1, \ldots, x_s) \quad , \quad \ldots \quad , \quad F_r(t; x_1, \ldots, x_s)$$

satisfy a *system* of (inhomogeneous) *linear-functional equations*, and such that $f(t)$ can be written as

$$f(t) = \frac{1}{1 - dt - \sum_i^r F_i(t; 1, \ldots, 1)} \quad ,$$

where d is the number of letters in the alphabet. Note that the system can be used to iteratively find higher and higher order terms (in t) of the F_i's, and hence, after plugging in all the x_i's to be 1, will yield successive terms of $f(t)$.

An Important Assumption

No symbolic mistake can ever have an instance that is a factor of an instance of (the same or another) symbolic mistake.

For example the input $Y^{a+1}H^{b+1}V^{c+1}H^{d+1}$ is not legitimate, since, for example, $YHHVVH$ is a factor of $YYYHHVVHH$. But we do not lose any generality by insisting on this. All we have to do is consider the minimal symbolic word $Y^1H^{b+1}V^{c+1}H^1$, that has the desired property, and of course the original problem is equivalent to the problem of enumerating words that avoid $Y^1H^{b+1}V^{c+1}H^1$ as factors, and the modified problem is even simpler, since it only involves a 2-parameter symbolic mistake.

The Clusters

The Classical Goulden-Jackson method goes verbatim to the present case, except that now the entries of the clusters are *symbolic* mistakes, and each cluster really represents infinitely many possible instances.

A cluster of m mistakes can be represented as a sequence

$$M_1 \, , \, i_1 \, , \, M_2 \, , \, i_2, \, \ldots \, , \, M_{m-1} \, , \, i_{m-1} \, , \, M_m$$

where $M_1, M_2, \ldots, M_m$ are symbolic mistakes, and $i_1, i_2, \ldots, i_{m-1}$ are integers such that $i_k \leq length(M_k)$ (for $k = 1, \ldots, m-1$), that indicate at what part of M_k starts the overlap with the next mistake M_{k+1}.

For example, if

$$M_1 = 1^{a+1}2^{a+b+1}3^{a+b+1}1^{b+1} \quad , \quad M_2 = 3^{a'+1}1^{a'+b'+1}2^{a'+b'+1}3^{b'+1} \quad , \quad M_3 = 2^{a''+1}3^{a''+b''+1}1^{b''+1} \quad ,$$

the following is one (out of many) clusters:

$$(M_1, 3, M_2, 3, M_3) \quad .$$

It is being assumed that the overlapping sections coincide. It may well happen that a symbolic cluster can't be realized, i.e. its set of instances is empty.

There is still some choice involved in instantiating a symbolic cluster. It is the decision where exactly does the first (symbolic) letter of M_{k+1} start under the i_k-th letter of M_k? That distance is also symbolic, let's denote it by c_k. Another decision to make is: how much does the symbolic letter of M_{k+1} that lies right under the last letter of M_k stick out? Let's call this distance d_k.

For a symbolic cluster to be realizable, the respective lengths ('frequencies') of overlapping letters that lie on top of each other must match, getting a system of linear diophantine equations. If they are all satisfied, then the whole cluster has a well-defined symbolic length: $length(cluster)$. If the last mistake in the cluster is the s-parameter mistake M_m, that has parameters $a_1, a_2, \ldots, a_s$, say, then

Definition:

$$Weight(M_1, i_1, M_2, i_2, \ldots, i_{m-1}, M_m) := -t^{length(cluster)} x_1^{a_1} x_2^{a_2} \ldots x_s^{a_s} \quad .$$

The Umbral Evolution

To each symbolic mistake associate a vertex. It is clear that each symbolic cluster is a walk in that graph, with the obvious adjacency pattern, where there is an edge between mistake M_i and mistake M_j for each of the possible interfaces, i.e. for each of the times a proper suffix of M_i coincides with a proper prefix of M_j, and their set of instances is non-empty.

Suppose that we have an already constructed cluster

$$M_1\ ,\ i_1\ ,\ M_2\ ,\ i_2\ , \ldots,\ M_{m-1}\ ,\ i_{m-1}\ ,\ M_m \quad .$$

In how many ways can one add another mistake M_{m+1}? Of course, all that matters is M_m, so things are Markovian.

Let's try to understand the edges and the Rota operators. Let $M_m = A$ and $M_{m+1} = B$. How can B follow A? Suppose A is

$$A = A_1^{L_1} A_2^{L_2} \ldots A_r^{L_r} \quad ,$$

where L_i are affine linear combinations of a set of discrete variables $\{a_1, \ldots, a_R\}$ and

$$B = B_1^{N_1} B_2^{N_2} \ldots B_s^{N_s} \quad ,$$

where N_i are affine linear combinations of a set of discrete variables $\{b_1, \ldots, b_S\}$, and the letters of both A and B belong to the alphabet W, i.e. $A_i \in W, i = 1, \ldots, r$ and $B_i \in W, i = 1, \ldots, s$.

Let's call the *underlying word* of a symbolic word $SymWord := a_1^{L_1} a_2^{L_2} \ldots a_k^{L_k}$ the plain word $a_1 a_2 \ldots a_{k-1} a_k$, and denote this by $U(SymWord)$. **Question**: What are the potential edges between A and B? **Answer:** Whenever a proper suffix of $U(A)$ coincides with a proper prefix of

$U(B)$. In other words, whenever there is a u such that

$$A_{r-u+1} = B_1 \quad , \quad A_{r-u+2} = B_2 \quad , \quad \ldots \quad , \quad A_r = B_u \quad .$$

In addition, the following system of *compatibility conditions*, in the variables

$$\{a_1, \ldots, a_R\} \cup \{b_1, \ldots, b_S\} \cup \{c, d\}$$

must be satisfied:

$$L_{r-u+1} = c + N_1 \quad , \quad L_{r-u+2} = N_2, \quad \ldots \quad , \quad L_{r-u+i} = N_i \quad ,$$

$$\ldots, \quad L_{r-1} = N_{u-1} \quad , \quad L_r + d = N_u \quad . \qquad (Ian)$$

Here c denotes the 'vacuum' before $B_1^{N_1}$ that lies under $A_{r-u+1}^{L_{r-u+1}}$, and d denotes the 'vacuum' after $A_r^{L_r}$ that lies above $B_u^{N_u}$.

The set of equations (Ian) needs to me modified when $u = r$, when $u = 1$, and when $s = 1$. When $u = r$, the c in the first equation of (Ian) needs to be replaced by $c+1$, since when $c = 0$ we won't have a proper cluster. When $s = 1$, i.e. $U(B)$ consists of just one letter, then the d has to be replaced by $d+1$. Finally, when $u = 1$, (Ian) only contains one equation: $L_r + d = c + N_1$.

If (Ian) has solutions with all the unknowns a_i, b_i, c, d *non-negative* integers, then there is an edge between A and B corresponding to this particular interface. If the set of solutions of (Ian) is empty, then there is no edge.

For any two symbolic mistakes A and B, there may be zero, or several, edges, each corresponding to the scenario where a proper suffix of $U(A)$ coincides with a proper prefix of $U(B)$ and the corresponding set of Linear Diophantine Equations, (Ian), is satisfied.

But how do we determine whether the system (Ian) is solvable? More generally, how do we solve it? Luckily, there are algorithms for doing this. First, there was the theoretical seminal work of Richard Stanley (see [S], section 4.6), that had an algorithm implicit in it. However, for our purposes, it is most convenient to adapt the beautiful work of George Andrews and his collaborators, Peter Paule and Axel Riese, in resurrecting, extending, and implementing McMahon's Partition Analysis (e.g. [APR]). In particular, the Mathematica package `OMEGA`, written by Mathematica Wizard Axel Riese, is exactly what we need.

Alas, I personally have two problems with `OMEGA`. The first one is *linguistic*. I am a Maple person, who has not, and *will not* use Mathematica. The other problem is even more serious. It is *cultural*, perhaps even *theological*. George Andrews, and his gang, justify the algorithm using the dubious (to say the least) notion of *analytic convergence*, and their power series represent 'analytic' functions (whatever that means).

I remedy both defects in [Z6], that is accompanied by a Maple package `LinDi` that is a Maple rendition of Riese's `OMEGA`, with a few minor technical enhancements, and one major conceptual improvement. The parts of `LinDi` that were needed for the present project were transported to the Maple packages `UGJ, SymUGJ, SiPerUGJ`.

Let's introduce 'continuous' variables $x_1, \ldots, x_R$ and $y_1, \ldots, y_S$ corresponding to the discrete variables $a_1, \ldots, a_R$, $b_1, \ldots, b_S$ respectively, plus z and w corresponding to the 'slack' variables c and d. Consider the formal power series

$$F(x_1, \ldots, x_R; y_1, \ldots, y_S; z, w) := \sum x_1^{a_1} x_2^{a_2} \ldots x_R^{a_R} \cdot y_1^{b_1} y_2^{b_2} \ldots y_S^{b_S} \cdot z^c w^d \quad , \qquad (Richard)$$

where the sum ranges over the set of solutions of (Ian) with all the a_i, b_j, c and d being *non-negative* integers.

By the celebrated theorem of Stanley ([S], Theorem 4.6.11), F must be a *rational function* of its arguments. Furthermore the numerator has positive coefficients, and the denominator is a product of terms of the form $(1 - monomial)$. Using `OMEGA`, or my Maple analog, `LinDi`, one can compute this rational function F explicitly.

Finally we are ready for the Umbral Evolution. The Pre-Umbra is

$$x_1^{a_1} \ldots x_R^{a_R} \rightarrow ConstantTerm_{x_1, \ldots, x_R} \frac{F(x_1, \ldots, x_R; y_1, \ldots, y_S; z, w)}{x_1^{a_1} x_2^{a_2} \ldots x_R^{a_R}} \quad .$$

The right side is an expression in the remaining variables $y_1, \ldots, y_s$ and z and w. Once we find it, we plug in $z = 1$ and $w = 1$.

However, this operation only takes care of the catalytic variables. The most important variable, t, carrying the length of the cluster, has not been mentioned yet. First, we must have a minus sign in front, since the sign of a cluster is (-1) to the power of the number of mistakes participating in it, and adding the mistake B changes the sign. Next we have to multiply by t raised to the power of the 'newly acquired length', which, in the above notation is $d + N_{u+1} + N_{u+2} + \ldots + N_s$. So what we *really* need to compute is the generating function

$$G(x_1, \ldots, x_R; y_1, \ldots, y_S; z, w) := \sum x_1^{a_1} x_2^{a_2} \ldots x_R^{a_R} \cdot y_1^{b_1} x_2^{b_2} \ldots y_S^{b_S} \cdot z^c w^d t^{d + N_{u+1} + N_{u+2} + \ldots + N_s} \quad , \qquad (George)$$

where the summation set in $(George)$ is the same as in $(Richard)$.

It is easy to see (and compute) G in terms of F. Indeed if the coefficient of a variable b_i in the affine linear combination $d + N_{u+1} + N_{u+2} + \ldots + N_s$ is, say, e_i, and its free term is e_0, then, obviously

$$G(x_1, \ldots, x_R; y_1, \ldots, y_S; z, w; t) := -t^{e_0} F(x_1, \ldots, x_r; t^{e_1} y_1, \ldots, t^{e_S} y_S; z, wt) \quad .$$

The full Pre-Umbra is

$$x_1^{a_1} \ldots x_r^{a_R} \rightarrow ConstantTerm_{x_1, \ldots, x_R} \frac{G(x_1, \ldots, x_r; y_1, \ldots, y_s; 1, 1; t)}{x_1^{a_1} x_2^{a_2} \ldots x_r^{a_R}} \quad . \qquad (GianCarlo)$$

This is unlike the situations in [Z2],[Z3], and [Z4], where our Maple packages found explicit expressions for pre-umbras, that were then converted to Umbral Schemes with procedures transported from the general package `ROTA`.

However, it turns out that the Pre-Umbra given in the form

$$x_1^{a_1} \dots x_r^{a_r} \rightarrow ConstantTerm \frac{H(x_1, \dots, x_r; y_1, \dots, y_s)}{x_1^{a_1} \dots x_r^{a_r}} \quad ,$$

can also be converted, often, into what we called Rota Operators in [Z1]. And the central tool of `OMEGA` and `LinDi`, viz. *partial fraction decomposition*, comes to the rescue once again.

Let's illustrate this with a simple example in one variable. Consider the pre-umbra

$$x^n \rightarrow Coeff_{x^n} \frac{a}{1 - Ax} \quad ,$$

yielding the pre-umbra $x^n \rightarrow aA^n$, which gives the umbra $f(x) \rightarrow af(A)$.

Since the denominators of the rational functions of interest are products of terms of the form $(1 - monomial)$, we can write:

$$R(x) = \frac{P(x)}{(1 - A_1x)(1 - A_2x) \cdots (1 - A_mx)} \quad ,$$

where x is the current variable, with the degree of $P(x)$, in x, being less than m. Performing a partial-fraction decomposition splits $R(x)$ into a sum of terms of the form $a/(1 - Ax)$, which was considered above.

In the generic case, things should work out, even when we have to constant-termize with respect to several variables. But once in a while, we might get stuck, in which case there is no Umbral Scheme. It is possible to also treat the non-generic case, but since in all the natural examples that I tried things worked out, I did not bother.

A Very Simple Example

Let's count the number of words in the alphabet $\{1, 2, 3\}$, that avoid any factor of the form $A := 12^{a+1}3^{a+1}1$, i.e. you can't have $\{1231, 122331, 12223331, \dots\}$ as factors. Even in this simple case we already have to avoid infinitely many mistakes.

The Clusters must be of the form $(A, 4, A, 4, A, 4, \dots, A)$. Let's call the cluster generating function $F(x; t)$, i.e. the weight-enumerator of all clusters with the weight $(-1)^m t^{length} x^a$, where the last A was $12^{a+1}3^{a+1}1$. The set of clusters consisting of one A has weight-enumerator $-t^4/(1 - t^2x)$, while the rest has weight-enumerator $-t^3F(1; t)/(1 - t^2x)$. It follows that $F(x; t)$ satisfies the functional equation:

$$F(x; t) = \frac{-t^4}{1 - t^2x} - \frac{t^3F(1; t)}{1 - t^2x} \quad .$$

The desired generating function is $1/(1-3t-F(1;t))$. In this trivial case, we can actually *solve* the Umbral Scheme. Plugging in $x=1$ yields

$$F(1;t) = \frac{-t^4}{1-t^2} - \frac{t^3F(1;t)}{1-t^2} \quad ,$$

and hence $F(1;t) = -t^4/(1-t^2+t^3)$, and the generating function is $f(t) = 1/(1-3t-F(1;t)) = (1-t^2+t^3)/(1-3t-t^2+4t^3-2t^4)$.

The Maple Package UGJ

This is implemented in the Maple package `UGJ`. In order to use it, download it from my website, keeping its name `UGJ`. Now get into Maple, by typing `maple` (or the appropriate command on your system), followed by `ENTER`. Once in Maple, assuming that you are in the same directory as `UGJ`, type `read UGJ:`, and follow the instructions given there. In particular, to see the list of the main procedures, type `ezra();`, and for help with any given procedure, type `ezra(ProcedureName);`. For example, try `ezra(UmSc);`.

Given a list of symbolic words ListSW, `UmSc(ListSW,x,t)` constructs an Umbral Scheme for the Cluster generating function for enumerating words that avoid instances of the words of `ListSW` as factors. For example, to get automatically the trivial example above, type:

```
UmSc( [ [[[1,1],[2,a+1],[3,a+1],[1,1]],[a]]],x,t);
```

Another main procedure is `UGJseries`. `UGJseries(ListSW,NumLetters,L)` finds the first `L` terms of the series expansion enumerating words that avoid any instances of the mistakes of ListSW, where `NumLetters` is the number of letters. For example, type:

```
UGJseries( [ [[[1,1],[2,a+1],[3,a+1],[1,1]],[a]]],3,10);
```

In order to check the program empirically, I wrote the procedure `CheckU`. `CheckU` uses the classical (finite) Goulden-Jackson method to check the validity of `UGJseries`, and hence of `UmSc`, since the former depends on the latter. The syntax of `CheckU` is: `CheckU(ListSW,Alphabet,L);`, where `Alphabet` is the alphabet. For example type:

```
CheckU( [ [[[1,1],[2,a+1],[3,a+1],[1,1]],[a]]],{1,2,3},10);.
```

You should get the same output.

The Symmetric Case

If the set of symbolic mistakes is symmetric with respect to the action of the symmetric group, then it is much more efficient to only consider each representative. This is implemented in the analogous Maple package `SymUGJ`. However, we had to slightly modify the notion of Umbral Scheme, to that of *Weighted Umbral Scheme*, to take care of the multiplicities resulting from each equivalence class. Hence the main procedure of `SymUGJ` is `WtUmSc`. Everything else is analogous.

Symmetry Under the Action of the Group of Signed Permutations

If the alphabet is $\{-1, 1, -2, 2, \ldots, -n, n\}$, then a stronger symmetry is possible. Namely, the action of the group of signed permutations. If our set of symbolic mistakes is invariant under the action of this group, we can have even a bigger saving in time and memory. In this case, this means that if, say, $2^1 1^{a+1} 2^1$ is a symbolic mistake, then so are $2^1(-1)^{a+1}2^1$, $(-2)^1 1^{a+1}(-2)^1$, $(-2)^1(-1)^{a+1}(-2)^1$, and the four other symbolic mistakes obtained by transposing 1 and 2. If we use `UGJ`, then we need an Umbral Matrix of size 8. If we use `SymUGJ`, we still need the Umbral Matrix to be of size 4. But if we exploit the full symmetry, then we would only need an Umbral Matrix of size 1.

This is implemented in the package `SiPerUGJ`, that is completely analogous.

Fancy Toy Models for Self-Avoiding Walks

John Noonan [N] used the finite Goulden-Jackson method, as extended in [NZ], to derive new upper bounds for the so-called connective constants for self-avoiding walks. He did this by considering the finite-memory approximation, that considers random walks on the lattice that do not visit any site visited in the previous m steps. In other words, the walker only remembers what she did in the previous m steps, and avoids the sites visited then, but she may return to sites visited more than m steps ago.

It is easy to see that a finite (m) memory walk on the d-dimensional lattice can be modeled in terms of a Goulden-Jackson type problem of enumerating the words in the alphabet $\{1, -1, 2, -2, \ldots, d, -d\}$ that avoid all the walks one obtains by starting anywhere on one of the (finitely many) self-avoiding polygons of length $\leq m$, and traversing it in either direction. This set of 'mistakes' is obviously invariant under the group of signed permutations, and John Noonan took advantage of this.

Even though finite-memory approximation give relatively good upper bounds, they are not theoretically interesting, since the generating functions are always rational, giving a boring asymptotics of the form $C\mu^n$ rather than the more realistic $C\mu^n n^\theta$ ($\theta \neq 0$).

A natural toy problem is to consider walks that avoid any rectangular subwalks (in addition to retracing), or, those that avoid retracing, rectangular, and hexagonal subwalks, etc.. Now we have infinitely many mistakes, and `SiPerUGJ` is ideally suited for the task. Procedures `SawUm4` and `SawUm6` compute the Umbral Schemes for these problems, respectively, and `Tony4` and `Tony6` find the 'series expansions'. Unfortunately, even though the generating functions are (probably) not rational, the asymptotics seems to still be $C\mu^n$ (i.e. $\theta = 0$), like in the Markovian, finite-memory case. So even though we excluded infinitely many kinds of bad events, apparently this was not enough. Also the empirically derived connective constants do not compete with the ones found by Noonan.

By combining rectangles and hexagons with larger, specific (not symbolic) mistakes, it should be possible to get improved upper bounds for connective constants of self-avoiding walks.

The webpage for this article: `http://www.math.temple.edu/~zeilberg/mamarim/mamarimhtml/umbV.html` contains sample input and output files for the three Maple packages `UGJ`, `SymUGJ` and `SiPerUGJ`, with Umbral Schemes and series expansions.

REFERENCES

[APR] George E. Andrews, Peter Paule, and Axel Riese, *MacMahon's partition analysis III: The Omega package*, to appear. Available from
`http://www.risc.uni-linz.ac.at/research/combinat/risc/publications`.

[GJ1] Ian Goulden and David M. Jackson, *An inversion theorem for cluster decompositions of sequences with distinguished subsequences*, J. London Math. Soc.(2)**20** (1979), 567-576.

[GJ2] Ian Goulden and David M. Jackson, *"Combinatorial Enumeration"*, John Wiley, 1983, New York.

[N] John Noonan, *New upper bounds for the connective constants of self-avoiding walks*, J. Stat. Physics **91** (1998), 871-888.

[NZ] John Noonan and Doron Zeilberger, *The Goulden-Jackson Cluster Method: Extensions, Applications, and Implementations*, J. Difference Eq. Appl. **5**, 355-377, (1999).

[S] Richard Stanley, *"Enumerative Combinatorics", vol. 1*, Wadsworth, Monterey, 1986. Reprinted by Cambridge University Press.

[Z1] Doron Zeilberger, *The Umbral Transfer-Matrix Method. I. Foundations*, J. Comb. Theory Ser. A **91** (2000), 451-463. [Rota memorial issue].

[Z2] Doron Zeilberger, *The Umbral Transfer-Matrix Method. II. Counting Plane Partitions*, Personal Journal of Ekhad and Zeilberger, `http://www.math.temple.edu/~zeilberg/pj.html` .

[Z3] Doron Zeilberger, *The Umbral Transfer-Matrix Method. III. Counting Animals*, submitted. Available from `http://www.math.temple.edu/~zeilberg/papers1.html` .

[Z4] Doron Zeilberger, *The Umbral Transfer-Matrix Method. IV. Counting Self-Avoiding Polygons and Walks*, submitted. Available from `http://www.math.temple.edu/~zeilberg/papers1.html` .

[Z5] Doron Zeilberger, *The Umbral Transfer-Matrix Method. V. The Goulden-Jackson Cluster Method for Infinitely Many Mistakes*, this article.

[Z6] Doron Zeilberger, *George Andrews's Resurrection of MacMahon's Partition Analysis Done Right: Let's Be fORmAL, not ANALytic*, in preparation.