

The Non-Communicative Goulden-Jackson Cluster Method

Xiangdong Wen

1. Introduction

Let V be a finite alphabet with $|V| = d$ letters a_1, a_2, \dots, a_d . Let B be a finite set of *words* (on V) and let $\mathcal{L}(B)$ denote the set of words (language) over V that avoid the members of B as factors. Suppose $q(n)$ is the total number of words with length n that avoid the words in B as factors. Then using the Goulden-Jackson cluster method ([4],[5] [1]), we could efficiently find the generating function

$$f(s) = \sum_{n=0}^{\infty} q(n)s^n. \quad (1)$$

However the function (1) lost the information of the letters and thus the information of the order of the letters. In order to keep track of these informations, we set up the general generating function

$$g(s_{a_1}, s_{a_2}, \dots, s_{a_d}) = \sum_{n=0}^{\infty} \sum_{a_1 a_2 \dots a_n \in \mathcal{L}(B)} s_{a_1} \cdot s_{a_2} \cdot s_{a_3} \cdot \dots \cdot s_{a_n}, \quad (2)$$

where the operand “ \cdot ” is non-communicative. The generating function (2) preserves not only the information of letters in a word but also the order of the letters. Actually it contains all the information of the language $\mathcal{L}(B)$. In this paper, we use the *non-communicative Goulden-Jackson method* to find a rational form of the function (2).

Keywords:

Goulden-Jackson cluster method, marked word, cluster, self-avoiding walks.

2. The non-communicative Goulden-Jackson cluster method

The non-communicative Goulden-Jackson cluster method has the same analog as the normal Goulden-Jackson cluster method except that the definitions of the weight of a word are different. Amazingly the procedures developed in [1] and [6] could all be applied to the non-communicative case. Here we provide an analog version to make it self-contained.

A **factor** of the word $a_1 a_2 \dots a_n$ is one of the words $a_i a_{i+1} \dots a_{j-1} a_j$, $1 \leq i \leq j \leq n$, that we shall denote by $[i, j]$. Two factors $[i, j]$ and $[i', j']$ **overlap** if they have at least one common letter. The **weight** of a word $w = a_1 a_2 a_3 \dots a_n$ is the non-communicative product $W(w) = s_{a_1} \cdot s_{a_2} \cdot \dots \cdot s_{a_n}$. Summing weights of all the words in $\mathcal{L}(B)$,

$$g(s_{a_1}, s_{a_2}, \dots, s_{a_d}) = \sum_{w \in \mathcal{L}(B)} W(w), \quad (3)$$

gives us the general generating function (2).

A word with some factors marked is called a **marked word**. A marked word could be written in the form:

$$(w; [i_1, j_1], [i_2, j_2], \dots, [i_l, j_l]), \text{ where } [i_r, j_r], 1 \leq r \leq l, \text{ are marked factors.}$$

A **cluster** is a marked word

$$(w_1 w_2 \dots w_n; [i_1(=1), j_1], [i_2, j_2], \dots, [i_l, j_l(=n)]),$$

where $[i_k, j_k]$ overlaps with $[i_{k+1}, j_{k+1}]$ for all $k = 1, 2, \dots, l-1$. The set of clusters \mathcal{C} can be partitioned into

$$\mathcal{C} = \bigcup_{u \in B} \mathcal{C}[u],$$

where $\mathcal{C}[u]$, $u \in B$, is the set of clusters whose first entry is u .

Define the **weight**, \overline{W} , of a marked word w with marked factors S , $S \subset B$, as

$$\overline{W}(w, S) = (-1)^{|S|} s_{a_1} \cdot s_{a_2} \cdot s_{a_3} \cdot \dots \cdot s_{a_n} = (-1)^{|S|} W(w),$$

where $|S|$ is the cardinality of S .

Let $B(w)$ be a subset of B whose members are factors of w . And let $N_B(w)$ denote the number of marked factors of w that belong to B . Using the inclusion-exclusion principle, we have:

$$\begin{aligned} f(s) &= \sum_{w \in \mathcal{L}(B)} W(w) \\ &= \sum_{w \in V^*} W(w) 0^{N_B(w)} \quad (\text{Define } 0^0 = 1) \\ &= \sum_{w \in V^*} W(w) [1 + (-1)]^{N_B(w)} \\ &= \sum_{w \in V^*} W(w) \sum_{S \subset B(w)} (-1)^{|S|} \\ &= \sum_{w \in V^*} \sum_{S \subset B(w)} (-1)^{|S|} W(w) \\ &= \sum_{w \in V^*} \sum_{S \subset B(w)} \overline{W}(w, S). \end{aligned}$$

Hence the generating function (2) is exactly the same as the generating function for the weighted marked word,

A non-empty marked word either starts with a letter that is not part of a cluster, or starts with a cluster. Peeling-off the maximal cluster, we get a shorter marked word. Thus we have the following decomposition

$$\mathcal{M} = \{\text{empty_word}\} \cup \mathcal{M}V \cup \mathcal{M}\mathcal{C}.$$

Taking weights on both sides and solving it for $\overline{W}(\mathcal{M})$, we have

$$g(s_1, s_2, \dots, s_d) = \overline{W}(\mathcal{M}) = \frac{1}{1 - (s_1 + s_2 + \dots + s_d) - \overline{W}(\mathcal{C})}. \quad (4)$$

For a given word $w = a_1 a_2 \dots a_n$, let $HEAD(w)$ be the set of all proper prefixes:

$$HEAD(w) := \{a_1 a_2 \dots a_k \mid k = 1, 2, \dots, n-1\},$$

and $TAIL(w)$ be the set of all proper suffixes

$$TAIL(w) := \{a_k a_{k+1} \dots a_n \mid k = 2, 3, \dots, n\},$$

and

$$OVERLAP(u, v) := TAIL(u) \cap HEAD(v).$$

Let u/r denote the operation of the word u chopping off its tail r , and let

$$(u : v) := \sum_{x \in OVERLAP(u, v)} \overline{W}(u/x).$$

Given a cluster in $\mathcal{C}[u]$, $u \in B$, it either consists of just u or chopping its head u results in a shorter cluster in $\mathcal{C}[v]$, $v \in B$ if $OVERLAP(u, v)$ is not empty. On the other hand, given a cluster in $\mathcal{C}[v]$, we could always reconstitute the bigger cluster in $\mathcal{C}[u]$ by adding some words in $OVERLAP(u, v)$. Hence, there exists a bijection:

$$\mathcal{C}[u] \leftrightarrow \{(u; [1, |u|])\} \bigcup_{v \in B} \mathcal{C}[u] OVERLAP(u, v).$$

Taking weights on both sides, we have:

$$\overline{W}(\mathcal{C}[u]) = (-1) \overline{W}(u) - \sum_{v \in B} (u : v) \cdot \overline{W}(\mathcal{C}[v]). \quad (5)$$

This is a system of $|B|$ linear equations with $|B|$ unknowns $\overline{W}(\mathcal{C}[v]), v \in B$. The classical method can not be used to solve the system for the product “ \cdot ” in the equation is non-communicative, Fortunately all the unknowns in the equations are at the rightmost part of the operand “ \cdot ”. We can use multiplication on the left to extract out the unknowns and then use the Gaussian elimination method to solve the system. This procedure could automatically be done by computer.

After solving these equations we could obtain $\overline{W}(\mathcal{C})$ by: $\overline{W}(\mathcal{C}) = \sum_{v \in B} \overline{W}(\mathcal{C}[v])$

The symbolic method ([6]) could also be implemented because $\overline{W}(\mathcal{C})$ is independent of the cardinality of the alphabet.

Example 1: Let $V = \{1, 2\}$, find the generating function for the language which does not have words in $B = \{111\}$ as factors. Introduce the non-communicative indeterminate s_1, s_2, s_3 . Using (5) we set up a system of one equation:

$$\overline{W}(\mathcal{C}[111]) = -s_1^3 - s_1 \cdot \overline{W}(\mathcal{C}[111]) - s_1^2 \cdot \overline{W}(\mathcal{C}[111]).$$

After solving it, we have

$$\overline{W}(\mathcal{C}[111]) = -\frac{1}{1 + s_1 + s_1^2} \cdot s_1^3,$$

and

$$g(s_1, s_2) = \frac{1}{1 - s_1 - s_2 + \frac{1}{1 + s_1 + s_1^2} \cdot s_1^3}.$$

This could be simplified further as

$$g(s_1, s_2) = \frac{1}{(1 + s_1 + s_1^2) \cdot (1 - s_1 - s_2) + s_1^3} \cdot (1 + s_1 + s_1^2)$$

Example 2: Let $V = \{1, 2\}$ and $B = \{12, 22\}$. Find the generating function of the language which does not have words in B as factors. Using (5) we set up a linear system of two equations:

$$\begin{cases} \overline{W}(\mathcal{C}[12]) &= -s_1 \cdot s_2 - s_1 \cdot \overline{W}(\mathcal{C}[22]) \\ \overline{W}(\mathcal{C}[22]) &= -s_2^2 - s_2 \cdot \overline{W}(\mathcal{C}[22]) \end{cases}.$$

After solving it, we have

$$\begin{cases} \overline{W}(\mathcal{C}[22]) &= -\frac{1}{1 + s_2} \cdot s_2^2 \\ \overline{W}(\mathcal{C}[12]) &= -s_1 \cdot s_2 + s_1 \cdot \frac{1}{1 + s_2} \cdot s_2^2 \end{cases}.$$

And finally we get

$$g(s_1, s_2) = \frac{1}{1 - s_1 - s_2 + \frac{1}{1 + s_2} \cdot s_2^2 + s_1 \cdot s_2 - s_1 \cdot \frac{1}{1 + s_2} \cdot s_2^2}.$$

3. Avoid Pairs

One of the most interesting problems is to find the generating function for the language which avoid some pair of the letters as factors. It is a special case for using of the Goulden-Jackson method. But there also exist a direct way to set up equations and thus to calculate the generating function: Divide the whole language into d sub-languages in which the words start with the same letter. The generating functions for each sub-group satisfy a system of linear equations which could be solved by revised Gaussian elimination method. And hence we could get the generating function for the language by summing up the generating functions for the sub-languages.

Here we give an example to illustrate how it works.

Example 3: Given an alphabet $V = \{1, 2\}$, and a set of pairs $B = \{11, 22\}$. Find the generating function of the language that avoids words in B as factors.

Let $C[i]$ be the sub-language in which each word starts with the letter $i, i = 1, 2$. then

$$C[1] = \{[1]\} \cup \{[1] \cdot C[2]\}.$$

Taking weight on both sides we have

$$\overline{W}(C[1]) = s_1 + s_1 \cdot \overline{W}(C[2]).$$

By the same way we could get another equation:

$$\overline{W}(C[2]) = s_2 + s_2 \cdot \overline{W}(C[1]).$$

Thus we can see all the unknowns are at the rightmost part of the operand “.”. Using multiplication on the left and also the Gauss elimination method, we could solve the system:

$$\begin{cases} \overline{W}(C[1]) &= s_1 \cdot [1 - \frac{1}{s_1 - \frac{1}{s_2}} \cdot (1 + s_1)] \\ \overline{W}(C[2]) &= -\frac{1}{s_1 - \frac{1}{s_2}} \cdot (1 + s_1) \end{cases}.$$

Finally the generating function could be calculated by

$$f(s_1, s_2) = 1 + \overline{W}(C[1]) + \overline{W}(C[2]).$$

4. The Maple package

All the procedures are included in the package “NONCOMM_GJ”, downloadable from the web address:

http://www.math.temple.edu/~wen/gj/noncomm/.

Users could get the detailed online help when opened the package by maple. The direct way to set up and solve the linear system to get the generating function for avoiding pairs are also included in the package.

5. Acknowledgment

This work will be a part of the author’s Ph.D. dissertation, written under the direction of Professor Doron Zeilberger (Rutgers University). I wish to thank Dr. Zeilberger for his generous support and encouragement.

References

- [1] John Noonan and Doron Zeilberger, “The Goulden-Jackson Cluster Method: Extensions, Applications, and Implementations”, *J. Difference Eq. Appl.*, 5(1999), 355-377.
- [2] John Noonan, “New Upper Bounds for The Connective Constants of Self-Avoiding Walks”, *J. Stat. Phys.*, 91(1998), 871-888.
- [3] N. Madras, and G. Slade, “The Self avoiding Walk”, Birkhauser, Boston (1993).
- [4] I. Goulden and D.M. Jackson, *An inversion theorem for cluster decompositions of sequences with distinguished subsequences*, J. London Math. Soc.(2)**20** (1979), 567-576.
- [5] I. Goulden and D.M. Jackson, “*Combinatorial Enumeration*”, John Wiley, 1983, New York.
- [6] X. Wen, “The symbolic Goulden-Jackson cluster method” *J. Difference Eq. Appl.*, vol. 11, No. 2(2005), 173-179.

**Xiangdong Wen; 638 Wachman Hall(038-16); 1805 N. Broad Street
 Department of Mathematics; Temple University; Philadelphia, PA 19122
 wen@math.temple.edu; 215-204-1655;
 AMS Classification codes:05A15
 Electronic version: <http://www.math.temple.edu/~wen/gj>**