ENUMERATION OF STRINGS

A. M. Odlyzko
AT&T Bell Laboratories
Murray Hill, New Jersey 07974
USA

ABSTRACT

A survey is presented of some methods and results on counting words that satisfy various restrictions on subwords (i.e., blocks of consecutive symbols). Various applications to comma-free codes, games, pattern matching, and other subjects are indicated. The emphasis is on the unified treatment of those topics through the use of generating functions.

1. Introduction

Enumeration of words that satisfy restrictions on their subwords is a very satisfying subject to deal with. Not only are there many applications for results in this area, but in addition one can use a wide range of mathematical techniques which lead to a variety of elegant results. These methods are usually not applicable when instead of subwords (i.e., blocks of consecutive symbols) one studies general subsequences.

We will consider strings (words and patterns will be used synonymously) over a fixed finite alphabet Ω of size $q \geq 2$. The basic results, from which most of the others are derived, enumerate strings of a given length that contain no elements of some fixed set of strings as subwords. The enumeration does not yield simple explicit formulas for the number of such strings, as no such formulas exist. Instead, formulas are obtained for the generating functions of these quantities. These generating functions are rational, and their special form leads to numerous results, both in asymptotic enumeration and in more algebraic and combinatorial applications.

In Section 2 we briefly review how rational generating functions can be used in both asymptotic enumeration and other settings. In Section 3, the important notion of a *correlation* of two strings is defined. Using that notion, Section 4 then presents the main formulas for the generating functions that arise in the enumeration of strings. Section 5 discusses some applications of these generating functions to the study of worst-case behavior of pattern matching algorithms, recognizing unavoidable sets of words, and nontransitive games. Section 6 discusses some of the main applications of these generating functions to asymptotic

enumeration, especially to the study of prefix-synchronized codes and occurrences of long repetitive patterns in strings.

This paper is based largely on the author's joint work with L. J. Guibas [11-15] and B. Richmond [21]. Only a small selection of results are presented, and they are chosen so as to emphasize how generating functions can be used as a unifying theme in a variety of contexts. Many of those results can also be obtained by other methods.

2. Rational generating functions

If $f(0)$, $f(1)$,... is any sequence, we define its generating function $F(z)$ to be

$$F(z) = \sum_{n=0}^{\infty} f(n) z^{-n} , \tag{2.1}$$

where z is an indeterminate. We use the form (2.1), instead of the more common form $F(z^{-1})$, which is a sum of terms of the form $f(n)z^n$, since (2.1) yields somewhat more elegant expressions for the generating functions arising in string enumeration.

In essentially all of the cases we will be concerned with, the generating function $F(z)$ will be rational, so that

$$F(z) = \frac{P(z)}{Q(z)} , \tag{2.2}$$

where $P(z)$ and $Q(z)$ are polynomials, which we may assume satisfy $\gcd(P(z), Q(z)) = 1$, $Q(z) \neq 0$. Several conclusions follow when $F(z)$ is rational. First of all, the $f(n)$ satisfy a linear recurrence with constant coefficients; if we write

$$Q(z) = \sum_{j=0}^{d} a_j \, z^{q-j} , \tag{2.3}$$

where d is the degree of $Q(z)$, so that $a_0 \neq 0$, then (2.1) and (2.2) show that

$$f(n) = -a_0^{-1} \sum_{j=1}^{d} a_j f(n-j) \quad \text{for} \quad n > d . \tag{2.4}$$

This linear recurrence can already be used to prove nontrivial results. For example, when coupled with some information about the coefficients a_j, it can be used to obtain lower bounds on the worst case performance of string searching algorithms (see Section 5).

Perhaps the most important consequence of the rationality of a generating function is that it yields significant information about the asymptotic behavior of the sequence being enumerated. If $F(z)$ has the form (2.2), and in addition $Q(z)$ has the form

$$Q(z) = a_0 \prod_{j=1}^{d} (z - \rho_j) , \tag{2.4}$$

where the ρ_j are distinct, then the partial fraction expansion of $F(z)$ is

$$F(z) = c_0 + \sum_{j=1}^{d} \frac{c_j}{z - \rho_j} , \tag{2.5}$$

where c_0 is some constant (there is no polynomial part to this expansion, since deg $P(z) \le$ deg $Q(z)$ by (2.1)), and

$$c_j = \frac{P(\rho_j)}{Q'(\rho_j)} , \quad 1 \le j \le d . \tag{2.6}$$

If the ρ_j are not all distinct, then (2.5) has to be replaced by a slightly more complicated expression, but we will not need to use it. The partial fraction expansion (2.5) shows that

$$\begin{aligned} F(z) &= c_0 + \sum_{j=1}^{d} \frac{c_j z^{-1}}{1 - \rho_j z^{-1}} \\ &= c_0 + \sum_{j=1}^{d} c_j z^{-1} \sum_{n=0}^{\infty} \rho_j^n z^{-n} \qquad (2.7) \\ &= c_0 + \sum_{n=1}^{\infty} z^{-n} \sum_{j=1}^{d} c_j \rho_j^{n-1} . \end{aligned}$$

This shows that

$$f(n) = \sum_{j=1}^{d} c_j \rho_j^{n-1} \quad , \quad n \geq 1 \ . \tag{2.8}$$

In particular, the expansion (2.8) shows that asymptotically, the growth rate of $f(n)$ is determined by the largest of the ρ_j;

$$|f(n)| \leq c \max_{1 \leq j \leq d} |\rho_j|^n \tag{2.9}$$

for some constant *c*. For finer details of the asymptotic behavior of $f(n)$, Eq. (2.8) by itself is often insufficient. The large zeros (those for which $|\rho_j|$ is maximal) can combine so as to make $f(n)$ small for some values of *n*, as happens, for example, for $F(z) = z^2/(z^2-1)$, where $f(n) = 0$ if *n* is odd, and $f(n) = 1$ if *n* is even. Some very sophisticated methods have been developed to deal with problems of this nature, as can be seen in [18] and the references cited there. For our purposes, though, such methods are not suitable.

Another source of difficulty with the use of expansions of the form (2.8) is that even when there is only one ρ_j with maximal absolute value, the other ρ_j can be important for small *n*. To obtain uniform estimates from (2.8), it becomes necessary to estimate the sizes of the c_j as well as of the ρ_j.

Fortunately for us, in string enumeration we are able to use a simple approach that lets us bypass the difficulties that arise in using the full partial fraction expansion. We have the following well-known general result.

Lemma 2.1. Suppose that $F(z)$ *is analytic in* $|z| \geq r > 0$ *with the exception of* $z = \rho$, $|\rho| > r$, *where it has a first order pole with residue* α, *and that*

$$|F(z)| \leq C \quad \textit{for} \quad |z| = r \ . \tag{2.10}$$

Then, if

$$F(z) = \sum_{n=0}^{\infty} f_n \, z^{-n} \ ,$$

we have

$$|f_n - \alpha\rho^{n-1}| \le r^n(C+|\alpha|(|\rho|-r)^{-1}) \quad \textit{for} \quad n\ge 1 \ . \tag{2.11}$$

Proof. We use Cauchy's theorem. Since

$$F(z) - \frac{\alpha}{z-\rho} = f_0 + \sum_{n=1}^{\infty} (f_n - \alpha\rho^{n-1})z^{-n}$$

and is analytic in $|z| \ge r$, we have for $n\ge 1$

$$f_n - \alpha\rho^{n-1} = \frac{1}{2\pi i} \int_{|z|=r} z^{n-1}\left\{F(z) - \frac{\alpha}{z-\rho}\right\} dz,$$

and so

$$|f_n - \alpha\rho^{n-1}| \le r^n \max_{|z|=r} |F(z) - \frac{\alpha}{z-\rho}|$$

$$\le r^n \left[C + \frac{|\alpha|}{|\rho|-r}\right] ,$$

which was to be proved. □

In the applications to string enumeration, the rational functions we will deal with satisfy the conditions of the lemma above. They have a single pole (which is real) and no other singularities nearby. Lemma 2.1 enables us to obtain very sharp asymptotic estimates without having to worry about existence of higher order poles or sizes of the residues other than at the dominant singularity. Thus the basic method of asymptotic estimation is very simple, especially when compared to some of the other asymptotic enumeration methods that are used in other situations (cf. [4,20]). The main difficulty is to obtain the generating functions for the quantities that are of interest and show that they satisfy the conditions of Lemma 2.1. As the following sections show, this is possible in a surprising variety of cases, but not always.

3. Correlations of strings

In order to describe the generating functions that arise in string enumeration, we introduce notation (from [13]) that expresses the way that two strings can overlap each other. If X and Y are two words, possibly of different lengths, over some alphabet, the *correlation* of X and Y, denoted by XY, is a binary string of the same length as X. The i-th bit (from the left) of XY is determined by placing Y under X so that the leftmost character of Y is under the i-th character (from the left) of X, and checking whether the characters in the overlapping segments of X and Y are identical. If they are identical, the i-th bit of XY is set equal to 1, and otherwise it is set equal to 0. For example, if $X = cabcabc$ and $Y = abcabcde$, then $XY = 0100100$, as depicted below:

```
X: c a b c a b c
Y: a b c a b c d e          0
     a b c a b c d e        1
       a b c a b c d e      0
         a b c a b c d e    0
           a b c a b c d e  1
             a b c a b c d e 0
               a b c a b c d e 0
```

Note that in general $YX \neq XY$ (they can even be of different lengths), and in the example above $YX = 00000000$. The *autocorrelation* of a word X is just XX, the correlation of X with itself. In the example above, $XX = 1001001$.

It is often convenient to interpret a correlation XY as a number in some base t, or else as a polynomial in the variable t, with the most significant bits being the ones on the left. Thus in the example above,

$$XY_t = t^5 + t^2, \quad XY_2 = 36 .$$

String correlations are important because they provide very compact descriptions of how two words can overlap. They are also fascinating subjects of study in their own right, and characterizations of strings that are correlations and enumeration of strings that have a given correlation are contained in [14] (see also [10]).

4. Basic generating functions

Now that correlations of strings have been defined, we can present the basic generating function results. Suppose that A is a word over some finite alphabet Ω of $q \geq 2$ letters, and let $f(n)$ be the number of strings of length n over that alphabet that contain no occurrences of A as a subword. We set $f(0) = 1$, and define, as in (2.1),

$$F(z) = \sum_{n=0}^{\infty} f(n) z^{-n} . \tag{4.1}$$

Then the basic result is that

$$F(z) = \frac{z\, AA_z}{1+(z-q)AA_z} . \tag{4.2}$$

This elegant formula, which apparently was first published in a somewhat different form by Solov'ev [26], is quite easy to prove, as we will now demonstrate. First define $f_A(n)$ to be the number of strings over Ω of length n that end with A (at the right end) but contain no other occurrence of A as a subword anyplace else. Set

$$F_A(z) = \sum_{n=0}^{\infty} f_A(n) z^{-n} . \tag{4.3}$$

To prove the formula (4.2) for $F(z)$, we will show that

$$(z-q)\ F(z) + zF_A(z) = z , \tag{4.4a}$$

$$F(z) = zAA_z F_A(z) , \tag{4.4b}$$

from which (4.2) and

$$F_A(z) = \frac{1}{1+(z-q)AA_z} \tag{4.5}$$

follow immediately. To prove (4.4a), consider any of the $f(n)$ strings of length n that do not

contain A as a subword, and adjoin to it (at the right end) any one of the letters from Ω. In this way we obtain $qf(n)$ distinct strings of length $n+1$. Any one of them either contains an appearance of A or not, but if it does, then that appearance has to be in the rightmost position. Therefore we obtain

$$qf(n) = f(n+1) + f_A(n+1) , \tag{4.6}$$

and this is valid for all $n \geq 0$. Multiplying (4.6) by z^{-n} and summing on $n \geq 0$, we easily obtain (4.4a).

To obtain (4.4b), consider any one of the $f(n)$ strings of length n that do not contain A, and append A at its right end. Suppose that the first (from the left) appearance of A in this string has its rightmost character in position $n+r$. Then $0 < r \leq |A|$ Furthermore, since A also appears in the last $|A|$positions, the prefix of A of length r has to be equal to the suffix of A of length r, and so the r-th entry in AA (counted from the right) is 1, which we write as $r \in AA$. The string of length $n+r$ is then counted by $f_A(n+r)$. Conversely, given any string counted by $f_A(n+r)$, $r \in AA$, there is a unique way to extend it to a string of length $n+|A|$ in which the last $|A|$ characters equal A. Therefore for $n \geq 0$ we have

$$f(n) = \sum_{r \in AA} f_A(n+r) . \tag{4.7}$$

Multiplying (4.7) by z^{-n} and summing on n yields (4.4b).

The main result about string enumeration is proved by methods that are only slightly more elaborate than those shown above, and so we will only state it. First we introduce the notion of a *reduced* set of words. The set $\{A, B, \ldots, T\}$ is called reduced if no word in this set contains any other as a subword. Since we are primarily interested in strings that contain none of a set $\{A, \ldots, T\}$ of words, we can reduce to the case of a reduced set of words by eliminating from that set any words that contain any others as subwords, since if G is a subword of Q, say, strings not containing G also fail to contain Q. Given a reduced set $\{A, B, \ldots, T\}$ of words, we let $f(n)$ denote the number of strings of length n over the alphabet Ω that contain none of $A, B, \ldots, T$, with $f(0) = 1$. We also let $f_H(n)$ denote the number of strings of length n over Ω that contain none of $A, B, \ldots, T$, except for a single appearance of H at the right end of the string. We define the generating functions $F(z)$ of $f(n)$ and $F_H(z)$ of $f_H(n)$ by (4.1) and (4.3).

Theorem 4.1 ([13,25]). If $\{A, \ldots, T\}$ *is a reduced set of patterns, the generating functions* $F(z), F_A(z), \ldots, F_T(z)$ *satisfy the following system of linear equations:*

$$\begin{aligned}
&(z-q)F(z) + zF_A(z) + zF_B(z) + \ldots + zF_T(z) = z \\
&F(z) - zAA_zF_A(z) - zBA_zF_B(z) - \ldots - zTA_zF_T(z) = 0 \\
&\cdots \\
&F(z) - zAT_zF_A(z) - zBT_zF_B(z) - \ldots - zTT_zF_T(z) = 0 \ .
\end{aligned} \tag{4.8}$$

It is easy to show ([13]) that the system (4.8) is nonsingular, and therefore determines $F(z)$, $F_A(z), \ldots, F_T(z)$. Moreover, Cramer's rule shows that these generating functions are all rational. When there is only a single excluded word, A, the system (4.8) reduces to (4.4). When more than a single word is excluded, the expressions become much more complicated than the simple forms of (4.2) and (4.5), a topic we will return to later.

Of the many other rational generating functions that arise in string enumeration, we will discuss just one more. Suppose that A is a word over some alphabet Ω of q letters, and let $g(n)$ denote the number of strings of length n over Ω that contain two distinct appearances of A at the beginning and at the end of the string, but no other occurrences of A as a subword anyplace else. (In particular, $g(n) = 0$ for $n \le \lfloor A \rfloor$) Then the generating function

$$G(z) = \sum_{n=0}^{\infty} g(n) z^{-n} \tag{4.9}$$

satisfies

$$G(z) = z^{-|A|} + \frac{(q-z)z^{-1}}{1+(z-q)AA_z} \ . \tag{4.10}$$

The proof of this result, which will be used in the discussion of prefix-synchronized codes in Section 6, is very similar to the proof of the other results mentioned above and can be found in [11]. Estimates for $g(n)$ are also central to the work described in [1,6].

There are many other rational generating functions that occur in string enumeration. In fact, essentially all generating functions that enumerate strings which satisfy restrictions on the number of appearances of particular subwords are rational, even when those generating functions are multivariate (cf. [9]). We will not discuss them any further, however, because they usually cannot be utilized profitably in asymptotic enumeration. Even Theorem 4.1 above has its limitations in this regard, as we will show.

5. Elementary applications of generating functions

In this section we discuss some applications of the generating functions presented in Section 4 to subjects other than asymptotic enumeration.

5.1 Coin-tossing games

The first application is to a rather amusing coin-tossing game. Suppose that A is some sequence of heads or tails, and that we toss a fair coin until A appears as a sequence of consecutive coin tosses (subword). What is the expected number of coin tosses until this happens? We can answer that question very easily using the generating functions of Section 4 (cf. [7,19]). The probability that A does not appear in the first n coin tosses is just $f(n)2^{-n}$, where $f(n)$ denotes the number of strings of heads or tails of length n that do not contain A as a subword. Hence the expected number of coin tosses until A appears is

$$\sum_{n=0}^{\infty} f(n)2^{-n} = F(2) ,$$

and by (4.2) this is $2AA_2$. In particular, this waiting time is strongly dependent on the periods of A.

W. Penney [22] has invented a penny-tossing game that exhibits the somewhat paradoxical feature of having a nontransitive dominance relation. In Penney's game, two players agree on some integer $k \geq 2$ at the start of the game. Player I then selects a pattern A of heads and tails of length k. Player II is shown A, and then proceeds to select another pattern B of k heads and tails. A fair coin is then tossed until either A or B appears as a subword of the sequence of outcomes. If A appears first, Player I wins, and if B appears first, Player II wins.

There are several possible reactions to Penney's game. One is to say that clearly the game is fair as everything is open. Another is to note that if the two players made their choices of A and

B independently of each other (with the event $A = B$ resulting in a draw) the game would obviously be fair because of the symmetry of the players' positions, and so Penney's game ought to be favorable to Player II since that player receives additional information, namely the choice of Player I. A third reaction (usually restricted to those who know the waiting time result discussed above) is to say that Player I should have the edge in Penney's game, since he can choose A to have very small waiting time.

The most interesting feature of Penney's game is that if $k \geq 3$, then no matter what A is, Player II can choose B so as to obtain probability of winning strictly greater than 1/2. (If $k = 2$ and $A = HT$, say, then Player II can only draw.) Thus this game is nontransitive in that for any A, one can find B that beats it, but then one can find C that beats B, etc. We do not have space to discuss the theory of Penney's game to any great extent (see [13]), but we indicate how the generating functions of Section 4 can be used in its analysis.

Let $\{A,B\}$ be the set of excluded patterns. Then solving the linear system of Theorem 4.1 gives

$$F_A(z) = \frac{BB_z - BA_z}{(z-2)(AA_z \cdot BB_z - AB_z \cdot BA_z) + AA_z + BB_z - AB_z - BA_z} . \tag{5.1}$$

Now $f_A(n)2^{-n}$ is the probability that Player I wins exactly on the n-th throw. Hence Player I wins with probability

$$\sum_{n=0}^{\infty} f_A(n)2^{-n} = F_A(2) = \frac{BB_2 - BA_2}{AA_2 + BB_2 - AB_2 - BA_2} . \tag{5.2}$$

Similarly the probability that Player II wins is given by a formula like (5.2) but with A and B interchanged. Hence the odds of Player II winning are given by

$$\frac{AA_2 - AB_2}{BB_2 - BA_2} . \tag{5.3}$$

This elegant formula, first proved by Conway by a much more complicated method, can be used to analyze Penney's game, and to show that Player II always has a winning strategy. Together with some information about possible sets of periods of strings, it can also be used to determine the optimal choice for B. Details can be found in [13].

5.2 Worst-case behavior of pattern-matching algorithms

The next application of the generating function of Section 4 is to the study of pattern-matching algorithms. We consider the problem of deciding whether a given word A of length k occurs in a string S of length n, with the cost of the algorithm being measured by the number of characters of S that the algorithm looks at. Ever since the appearance of [17], it has been known that one can decide whether A appears in S in $O(n)$ character comparisons, and a lot of work has been done on deriving improved algorithms. Furthermore, starting with the Boyer-Moore algorithm [3], methods have been known that on average look at only αn of the characters of S for some $\alpha < 1$ (which depends on the alphabet and on the algorithm). This situation gave rise to the question whether there is any pattern-matching algorithm that is sublinear in the worst case; i.e., which never looks at more than βn characters of S for $n \geq n_0$, where $\beta < 1$. This question was answered in the negative by Rivest [24]. We present a simplified version of Rivest's proof which relies on the explicit form of the generating function (4.2) [13]. The problem is not trivial, since in many cases it is possible to decide whether A appears in S without looking at all the characters of S. For example, if $A = 00$ and $n = 4$, so $S = s_1 s_2 s_3 s_4$, we can query whether $s_2 = 0$. If the answer is that $s_2 \neq 0$, then there will be no need to even look at s_1. Hence we may assume that the answer is that $s_2 = 0$. But then we can query whether $s_3 = 0$. If the answer is that $s_3 = 0$, then $A = s_2\ s_3$, and we do not need to look at s_1 and s_4. If the answer is that $s_3 \neq 0$, though, we do not need to look at s_4. Thus we can always decide whether A is a subword of S by looking at ≤ 3 characters of S. Note that we have phrased the question in terms of deciding whether A occurs as a subword of S, instead of asking for the determination of all such occurrences of A. In the latter situation the problem is indeed trivial (at least in general), since if $A = 0...0$, and the answer to a query about any character is that it's 0, any algorithm will clearly have to look at all characters of S to determine all the $n-k+1$ occurrences of A.

To prove that there is no sublinear worst-case algorithm for determining whether A is a subword of S, suppose that there is an algorithm which, given any string $S = s_1 \cdots s_n$ of length n, can determine whether A is a subword of S by looking at $\leq n-1$ characters of S. Suppose that for a particular string S, the algorithm looks only at the characters in positions $i_1, \ldots, i_r$. Then, for any of the q^{n-r} strings $S' = s'_1 \cdots s'_n$ which have $s'_{i_j} = s_{i_j}$, $1 \leq j \leq r$, the algorithm will examine the same characters as it does in examining S, and will come to the same conclusion as to whether A is a subword of S or not. Since $r \leq n-1$, $q \mid q^{n-r}$, and so the total number of strings that do not contain A as a subword is divisible by q. This can of

course happen for some values of n. The generating function (4.2) shows that this cannot happen for too many values of n.

Let $f(n)$ be the number of strings of length n that do not contain A as a subword, with $f(0) = 1$. Then (4.2) implies that if

$$1+(z-q)\ AA_z = z^k - \sum_{j=0}^{k-1} h_j z^j , \tag{5.4}$$

then

$$f(n) = \sum_{j=0}^{k-1} h_j f(n-k+j) \quad \text{for} \quad n \geq k .$$

But by (5.4), $h_0 \equiv 1 \pmod{q}$, so

$$f(n-k) \equiv f(n) - \sum_{j=1}^{k-1} h_j f(n-k+j) \pmod{q} \tag{5.5}$$

for $n \geq k$. If for some n we had $f(n) \equiv f(n-1) \equiv \cdots \equiv f(n-k+1) \equiv 0 \pmod{q}$, then by (5.5) we would have $f(n-k) \equiv 0 \pmod{q}$, and by induction $f(m) \equiv 0 \pmod{q}$ for all m, $0 \leq m \leq n$. But that contradicts $f(0) = 1$.

The conclusion to be drawn from the above discussion is that among any k consecutive values of n, there is at least one for which $f(n)$ is not divisible by q. For such a value of n, any algorithm will have to examine all n characters of some string S to determine whether A is a subword of S or not. Since the number of characters that have to be examined in the worst case is a monotone function of n, this also shows that for any n, at least $n-k+1$ characters have to be examined in some string S, and so there is no algorithm that is sublinear in the worst case.

5.3 Unavoidable sets of words

The final application to be discussed in this section is to the problem of determining when a set of words is *unavoidable*; i.e., any sufficiently long string over the given alphabet contains at least one of these words as a subword. As an example, over the binary alphabet, the set $\{000,\ 1111,\ 01\}$ is unavoidable. How can we recognize when a set is unavoidable? One way is to use Theorem 4.1. If $\{A, B, \ldots, T\}$ is the set in question, we can use the system (4.8) to

determine $F(z)$, the generating function for the number of strings that contain none of $\{A, \ldots, T\}$ as subwords. This set of words is then unavoidable if and only if $F(z)$ is a polynomial in z^{-1}.

Another way to determine unavoidability of a set of words is through the construction of a finite-state automaton recognizing the words $A, \ldots, T$. The two approaches seem to be of comparable complexity, but the one presented above might be somewhat simpler to implement.

The main reason for discussing the problem of determining unavoidability here is that it points out some of the limitations of the use of generating functions in enumeration. Unavoidable sets $\{A, \ldots, T\}$ lead to the collapse of the generating functions $F(z)$ to polynomials in z^{-1}. Yet almost all the basic asymptotic enumeration results, such as those of Section 6, depend on the use of Lemma 2.1, which depends on the generating function having a single dominant singularity away from 0. Thus we cannot expect to be able to apply the simple method of Lemma 2.1 even to all the generating functions determined by Theorem 4.1.

6. Asymptotic enumeration of strings

The end of the preceding section concluded with remarks about the limitations on the use of generating functions in asymptotic string enumeration. This section demonstrates that in spite of these limitations, numerous interesting asymptotic estimates can be obtained. The basic method relies on nothing more sophisticated than Lemma 2.1 and Rouche's theorem. However, usually substantial work is needed to apply these basic tools.

Perhaps the simplest case to consider is that of counting the number of strings, $f(n)$, of length n over an alphabet of size 2 that do not contain some fixed pattern A as a subword. (The restriction to $q = 2$ is imposed here for the sake of clarity, to make it easier to compare various quantities.) By (4.2), the generating function of $f(n)$ has the form

$$F(z) = \frac{zAA_z}{1+(z-2)AA_z} \ . \tag{6.1}$$

To apply Lemma 2.1, we need to show that $F(z)$ has a single dominant pole. To do this, we need to study the zeros of the polynomial in the denominator on the right side of (6.1). As it turns out, the only feature of the denominator polynomial that we will use is that AA_z has all its coefficients 0 and 1. It is easy to obtain the following result (cf. Lemma 2 of [11]).

Lemma 6.1. There is a constant $c_1 > 0$ *such that if* $u(z)$ *is a polynomial,*

$$u(z) = z^{k-1} + \sum_{j=2}^{k} u_j z^{k-j} \quad , \quad u_j = 0 \text{ or } 1 \, , \tag{6.2}$$

then for $|z| \geq 1.7$,

$$|u(z)| > c_1 \, (1.7)^k \, . \tag{6.3}$$

Using Lemma 6.1, it is easy to show that $F(z)$ has a single dominant singularity (cf. Lemma 3 of [11]).

Lemma 6.2. There is a constant $c_2 > 0$ *such that if* $u(z)$ *satisfies the conditions of Lemma 6.1 and* $k \geq c_2$, *the* $1+(z-2) \, u(z)$ *has exactly one zero* ρ *(of multiplicity one) with* $|\rho| \geq 1.7$.

Proof. On the circle $|z| = 1.7$,

$$|z-2| \, |u(z)| > 0.3 c_1 (1.7)^k > 1 \quad \text{for} \quad k \geq c_2 \, .$$

Hence by Rouche's theorem, $(z-2) \, u(z)$ and $1+(z-2) \, u(z)$ have exactly the same number of zeros in $|z| \geq 1.7$, namely one. □

It is also possible to localize the dominant zero ρ quite precisely. In a neighborhood of $z=2$, $u(z)$ is essentially $u(2)$, so we expect the zero ρ to be close to the solution of $1+(z-2) \, u(2) = 0$; i.e., $\rho \approx 2 - u(2)^{-1}$. This can be shown to be true, and we obtain the following result. (A more precise estimate is contained in Lemma 4 of [11].)

Lemma 6.3. If $u(z)$ *satisfies the conditions of Lemma 6.1, and* $k \geq c_2$, *then the single zero* ρ *of* $1+(z-2) \, u(z)$ *in* $|z| \geq 1.7$ *satisfies*

$$\rho = 2 - \frac{1}{u(2)} - \frac{u'(2)}{u(2)^3} + O(k^2 2^{-3k}) \quad \textit{as} \quad k \to \infty \, .$$

On the circle $|z| = 1.7$, the generating function $F(z)$ of (6.1) is $O(1)$ by Lemma 6.1. Hence Lemma 2.1 applies directly, and we find that (cf. Eq. (7.3) of [13])

$$f(n) = \frac{\rho^n}{1-(2-\rho)^2 u'(\rho)} + O((1.7)^n) \quad \text{as} \quad n \to \infty , \tag{6.4}$$

where $u(z) = AA_z$.

Estimates of the form (6.4) can be used in a variety of applications. We will discuss only the case of prefix-synchronized codes. A code $\boldsymbol{C}$ of block length n (over the binary alphabet, say) is said to be *comma-free* if for every pair $(a_1, \ldots, a_n)$, $(b_1, \ldots, b_n)$ of codewords in $\boldsymbol{C}$, no subword of length n of their concatenation $(a_1, \ldots, a_n, b_1, \ldots, b_n)$, other than the first and last blocks of length n, is a codeword of $\boldsymbol{C}$. When such a code is used in a communication system, the receiver can always recover synchronization by examining the transmitted signal until a codeword is encountered. It is known [5,16] that at least for odd n, comma-free codes of length n can be found which are of size

$$n^{-1}\ 2^n(1+o(1)) \quad \text{as} \quad n \to \infty . \tag{6.5}$$

One disadvantage of general comma-free codes, though, is that it is usually hard to decide whether a word is in the code. In order to overcome this disadvantage, E. N. Gilbert invented *prefix-synchronized codes*, a special class of comma-free codes. Such a code is characterized by a prefix A of length $k < n$. Every codeword has A as a prefix, and the remaining $n-k$ characters of the codewords are chosen so that in any concatenation of codewords, A occurs only at the beginnings of codewords. Synchronization is then very easy, since all that is needed is to scan the incoming stream of characters until the prefix A is encountered, and that can be done very efficiently using string searching algorithms. The question then arises as to the efficiency of prefix-synchronized codes. Gilbert [8] showed that one can construct such codes of size

$$n^{-1}\ 2^{n-1}(1+o(1))\ \log\ 2 \sim (0.346...)n^{-1}2^n \quad \text{as} \quad n \to \infty ,$$

which is not much less than the bound (6.5) attainable (at least for odd n) by maximal comma-free codes. Gilbert also conjectured that optimal prefixes can be chosen to be of the form $A = 11...110$ for an appropriate length k. This conjecture has been shown to be true, and a more precise estimate of the size of the largest prefix-synchronized code has been obtained [11].

Theorem 6.1. Given a positive integer n, choose the unique integer $k = k(n)$ *so that* $\beta = n2^{-k}$ *satisfies*

$$\log 2 \le \beta < 2 \log 2 .$$

Then the maximal prefix-synchronized code of length n has cardinality

$$n^{-1}2^n \ \beta e^{-\beta} \ (1+o(1)) \quad \text{as} \quad n \to \infty .$$

Moreover, if n is large enough, maximal prefix-synchronized codes can be constructed with prefixes $A = 11...10$ *of length* $k-1$, k, *or* $k+1$.

The results of Theorem 6.1 can be generalized (see [11]) to non-binary alphabets. However, if the alphabet is of size $q \ge 5$, then it is not always possible to construct maximal codes using prefixes of the form $A = 11...10$. For all alphabets, the phenomenon of asymptotic oscillation holds, so that the size of the maximal prefix-synchronized codes is not asymptotic to a constant multiple of $n^{-1}q^n$. For example, Theorem 6.1 shows that for $q=2$, the ratio of the maximal code size to $n^{-1}2^n$ oscillates between $(\log 2)/2 = 0.3465...$ and $e^{-1} = 0.3678....$

The proof of Theorem 6.1 relies on the generating function formula (4.10) and the stronger forms of lemmas 6.1-6.3 that are presented in [11]. Note first of all that given a prefix A, the allowable codewords are precisely those strings of length n which, when A is adjoined to them at the right end, have A as a prefix and a suffix, but do not contain A as a subword elsewhere. Hence, in the notation of Section 4, the maximal size of a code of length n with prefix A is just $g(n+|A|)$, for which we have the generating function (4.10). In particular, this formula shows that the size of the maximal code determined by a prefix A depends only on the autocorrelation of A, and not on A itself. The proof of Theorem 6.1 is quite involved, but in principle quite straightforward, involving very careful analysis.

So far we have discussed asymptotic enumeration results only in cases where we were basically excluding a single word A, and so the generating functions ((4.2) and (4.10)) had denominators of the simple form $1+(z-q)AA_z$. When we count strings that exclude larger numbers of words, we cannot expect our denominator to be this nice and easy to deal with. The unavoidable sets of words discussed at the end of Section 5 show that in fact quite pathological behavior can occur. However, in some cases it is possible to obtain sharp asymptotic estimates from Theorem 4.1. That is the case with repetitive runs, which were studied extensively in [15].

For simplicity we will again consider only the binary alphabet. Let $B = b_1 \cdots b_n$ be a word that is nonperiodic; i.e., B cannot be written as $B = CC \cdots C$ for any word C shorter than B. (As an example, 010 is nonperiodic while 0101 is not.) A *B-run* of length k is a word $A = BB \cdots BB'$ of length k, where B' is a prefix of B. For example, if $B = 010$, then 01001 is a B-run of length 5. We will discuss the appearance of long B-runs in random sequences, when B is held fixed. In many situations, a much more interesting question to study is that of appearance of B^*-runs. A *B^*-run* of length k is a word $A = B''BB \cdots BB'$, where B' is a prefix of B and B'' is a suffix of B. Thus, if $B = 010$, then 001001 is a B^*-run of length 6.

The theory of B-runs follows easily from the results we have already discussed. If $A = BB \cdots BB'$, $|A| = k$, and $f(n)$ denotes the number of binary strings of length n that do not contain A, then the generating function for $f(n)$ is given by (4.2). Moreover, since B is nonperiodic, it is easy to show that

$$AA_z = z^{k-1} + z^{k-m-1} + z^{k-2m-1} + \ldots + z^{k-rm-1} + p(z) ,$$

where $r = [k/m] - 1$ and deg $p(z) < m$, so we obtain the following result (Theorem 2.2 of [15]).

Theorem 6.2. If B is a nonperiodic pattern of length m, and A is a B-run of length k, then the generating function of $f(n)$, the number of strings of length n that do not contain A, is of the form

$$F(z) = \frac{zQ(z)}{1+(z-2)Q(z)} ,$$

where

$$Q(z) = \frac{z^{k+m-1}}{z^m - 1} + \frac{q(z)}{z^m - 1} ,$$

and $q(z)$ is a polynomial of degree $< 2m$ with coefficients bounded by 2 in absolute value.

The form of $F(z)$ given by Theorem 6.2 is sufficiently explicit to obtain very good estimates for the dominant singularity ρ and the residue at ρ so that

$$f(n) \approx 2^n \exp(-n(1-2^{-m})2^{-k}) \tag{6.6}$$

is a very good approximation. (See [15] for more precise estimates.)

The theory of B^*-runs is considerably more complicated. The number of strings of length n that do not contain a B^*-run of length k is the number $f(n)$ of Theorem 4.1 when we exclude the m words $A(1), \ldots, A(m)$ of length k given by

$$A(r) = b_{m-r+1}\, b_{m-r+2} \cdots b_m\, BB \cdots BB' ,$$

where $B' = B'(r)$ is a prefix of B of the appropriate length. The denominator of the generating function $F(z)$ of $f(n)$ is then the determinant of an $m+1$ by $m+1$ matrix. The entries of that matrix are very strongly constrained; for example, if $1 \le r \le s \le m$, then it can be shown that

$$A(r)A(s)_z = \frac{z^{k-1-s+r} + g_{s,r}(z)}{z^m - 1} ,$$

where $g_{s,r}(z)$ is a polynomial of degree $< 2m$ with coefficients that are ≤ 2 in absolute value. Therefore by going through some very involved algebra, one can obtain a relatively explicit form for $F(z)$ (Theorem 2.3 of [15]).

Theorem 6.3. If B is a nonperiodic word of length m, then the generating function $F(z)$ of $f(n)$, the number of binary strings of length n that contain no B^*-run of length $k \ge 3m$ has the form

$$F(z) = \frac{z^{(k+1)m+1}(z^m-1)^{m-1} + g_1(z)}{(z-2)z^{(k+1)m}(z^m-1)^{m-1} + g_2(x)} .$$

Here $g_1(z)$ and $g_2(z)$ are polynomials of degrees $\le k(m-1)+c_1$ with coefficients that are $\le c_2$ in absolute value, and c_1 and c_2 are constants that depend only on m, but not on k. Furthermore,

$$g_1(2) = O(2^{k(m-1)}) ,$$

$$g_2(2) = m2^{km-k+m}(2^m-1)^{m-1} + O(2^{k(m-2)}) ,$$

where the constants implied by the O-notation depend only on m.

Once Theorem 6.3 is available, it is not too difficult to show that $F(z)$ has a dominant singularity and to estimate its location and residue. One finds that

$$f(n) \approx 2^n \exp(-nm2^{-k-1}) \tag{6.7}$$

is a very good approximation in this case (cf. (6.6), which corresponds to *B*-runs).

Once estimates like (6.6) and (6.7) (or rather, the more precise estimates established in [15]) are obtained, one can obtain a variety of results from them. For example, of $E_B(n)$ denotes the expected length of a *B*-run in a random string of length *n*, and $E_B^*(n)$ the expected length of a B^*-run, then [15]

$$E_B(n) = \lg n - 1/2 + \gamma(\log 2)^{-1} + \lg(1-2^{-m})$$

$$+ v(\lg n + \lg(1-2^{-m})) + o(1) \quad \text{as} \quad n \to \infty ,$$

$$E_B^*(n) = \lg n - 3/2 + \gamma(\log 2)^{-1} + \lg m$$

$$+ v(\lg n + \lg m) + o(1) \quad \text{as} \quad n \to \infty ,$$

where lg x is the logarithm of x to base 2, γ is Euler's constant (0.577...), and $v(x)$ is a certain nonconstant continuous periodic function of period 1 and mean 0. (There is a mistake in [15] in the statement of the result about $E_B(n)$, with -1/2 incorrectly replaced by -5/2.) When *B* is a single letter, such results were obtained first by Boyd [2].

References

1. A. Apostolico and A. S. Fraenkel, Fibonacci representations of strings of varying length using binary separators, to be published.
2. D. W. Boyd, Losing runs in Bernoulli trials, unpublished manuscript (1972).
3. R. S. Boyer and J. S. Moore, A fast string searching algorithm, *Comm. ACM 20* (1977), 762-772.
4. N. G. de Bruijn, *Asymptotic Methods in Analysis*, North-Holland 1958.
5. W. L. Eastman, On the construction of comma-free codes, *IEEE Trans. Information Theory IT-11* (1965), 263-266.
6. A. S. Fraenkel, The use and usefulness of numeration systems, to be published.
7. H. U. Gerber and S. R. Li, The occurrence of sequence patterns in repeated experiments and hitting times in a Markov chain, *Stoch. Proc. Appl. 11* (1981), 101-108.
8. E. N. Gilbert, Synchronization of binary messages, *IRE Trans. Inform. Theory IT-6* (1960), 470-477.
9. I. P. Goulden and D. M. Jackson, *Combinatorial Enumeration*, Wiley 1983.
10. L. J. Guibas, Periodicities in strings, to be published in this volume.
11. L. J. Guibas and A. M. Odlyzko, Maximal prefix-synchronized codes, *SIAM J. Appl. Math. 35* (1978), 401-418.
12. L. J. Guibas and A. M. Odlyzko, A new proof of the linearity of the Boyer-Moore string searching algorithm, pp. 189-195 in Proc. 18th IEEE Found. Comp. Sci. Symposium, 1977. Revised version in *SIAM J. Comput. 9* (1980), 672-682.
13. L. J. Guibas and A. M. Odlyzko, String overlaps, pattern matching, and nontransitive games, *J. Comb. Theory (A) 30* (1981), 183-208.
14. L. J. Guibas and A. M. Odlyzko, Periods in strings, *J. Comb. Theory (A) 30* (1981), 19-42.
15. L. J. Guibas and A. M. Odlyzko, Long repetitive patterns in random sequences, *Z. Wahrscheinlichkeits. v. Geb. 53* (1980), 241-262.

16. B. H. Jiggs, Recent results in comma-free codes, *Canad. J. Math. 15* (1963), 178-187.

17. D. E. Knuth, J. H. Morris, and V. R. Pratt, Fast pattern matching in strings, *SIAM J. Comput. 6* (1977), 323-350.

18. K. K. Kubota, On a conjecture of Morgan Ward, I and II, *Acta Arith. 33* (1977), 27-48.

19. P. T. Nielsen, On the expected duration of a search for a fixed pattern in random data, *IEEE Trans. Inform Theory IT-19* (1973), 702-704.

20. A. M. Odlyzko, Some new methods and results in tree enumeration, *Congressus Numerantium 42* (1984), 27-52.

21. A. Odlyzko and B. Richmond, On the compositions of an integer, pp. 199-210 in *Combinatorial Mathematics VII*, Proc. Seventh Austr. Conf. Comb. Math., R. W. Robinson, G. W. Southern, and W. D. Wallis, eds., Lecture Notes in Mathematics #829, Springer 1980.

22. W. Penney, Problem: penney-ante, *J. Recreational Math. 2* (1969), 241.

23. P. Révész, Strong theorems on coin tossing, pp. 749-754 in *Proc. 1978 Int'l. Congress of Mathematicians*, Helsinki, 1980.

24. R. L. Rivest, On the worst-case behavior of string-searching algorithms, *SIAM J. Comput. 6* (1977), 669-674.

25. S. W. Roberts, On the first occurrence of any of a selected set of outcome patterns in a sequence of repeated trials, unpublished Bell Laboratories report (1963).

26. A. D. Solov'ev, A combinatorial identity and its application to the problem concerning the first occurrences of a rare event, *Theory Prob. Appl. 11* (1966), 276-282.