Bulletin of Mathematical Biology Vol. 49, No. 6, pp. 737–759, 1987.
Printed in Great Britain.

0092–8240/87$3.00 + 0.00
Pergamon Journals Ltd.

FORMAL LANGUAGE THEORY AND DNA: AN ANALYSIS OF THE GENERATIVE CAPACITY OF SPECIFIC RECOMBINANT BEHAVIORS

■ TOM HEAD
Department of Mathematical Sciences,
University of Alaska,
Fairbanks, AK 99775, U.S.A.

A new manner of relating formal language theory to the study of informational macromolecules is initiated. A language is associated with each pair of sets where the first set consists of double-stranded DNA molecules and the second set consists of the recombinational behaviors allowed by specified classes of enzymatic activities. The associated language consists of strings of symbols that represent the primary structures of the DNA molecules that may potentially arise from the original set of DNA molecules under the given enzymatic activities.

Attention is focused on the potential effect of sets of restriction enzymes and a ligase that allow DNA molecules to be cleaved and reassociated to produce further molecules. The associated languages are analysed by means of a new generative formalism called a splicing system. A significant subclass of these languages, which we call the persistent splicing languages, is shown to coincide with a class of regular languages which have been previously studied in other contexts: the strictly locally testable languages.

This study initiates the formal analysis of the generative power of recombinational behaviors in general. The splicing system formalism allows observations to be made concerning the generative power of general recombination and also of sets of enzymatic activities that include general recombination.

1. Introduction. The purpose of this article is to establish a new relationship between formal language theory and the study of informational macromolecules. Formal language theory is a branch of theoretical computer science that is devoted to the study of sets of finite strings (called languages) of symbols chosen from a prescribed finite set (called an alphabet). In considering the primary structure of a protein molecule, an RNA molecule, or a double-stranded DNA molecule it is natural to think of the first as a string over an alphabet of twenty symbols each of which represents an amino acid, the second as a string over an alphabet of four symbols each of which represents a ribonucleotide, and the third as a string over an alphabet of four symbols each of which represents a hydrogen-bonded deoxyribonucleotide pair. The theory of formal languages has already been related in at least three distinct ways to the study of these macromolecules: (i) Eberling and Jimenez-Montano (1980) have established measures of complexity of protein molecules through the use of context-free grammars and Jimenez-Montano (1984) has continued this line of research; (ii) Brendel and Busse (1984) have expressed the process of the

translation of messenger RNA into protein by means of finite state transducers and have derived conclusions using the closure properties of the class of regular languages; (iii) algorithms for string comparisons have been analysed and applied to the handling of sequence data for macromolecules in an extensive literature that can be entered through the special issue of this *Bulletin* honoring M. O. Dayhoff (Martinez, 1984). Two very recent, significant additions to this growing literature are made by Landau *et al.* (1986).

In this article we represent the set of double-stranded DNA molecules that may arise from an initial set of DNA molecules in the presence of specified enzyme activities as a language over the four-symbol alphabet of deoxyribonucleotide pairs. The recombinational power of such enzyme activities is idealized as a set of operations on the strings over this four-symbol alphabet. The fundamental theoretical construct is the language that is the closure of the original set of strings under the associated set of operations. This language is then analysed using the methods of formal language theory.

In Section 2 we explain in detail the language used to model a set of DNA molecules in the absence of enzymes. Sections 3 and 4 are the core of this article. In Section 3 the action of sets of restriction enzymes on sets of DNA molecules is given a formal representation and analysis. In Section 4 examples are worked out that illustrate each of the concepts introduced in Section 3. These examples can be read in parallel with Section 3 or consulted as needed in the reading of that section. In Section 5 general recombination is discussed. In Section 6 recombinational processes of additional types are mentioned as examples for future analysis. In Section 7 it is shown that general recombination may be regarded as dominant over other recombinational behaviors in determining the global structure of the language of potential DNA molecules. In Section 8 a brief summary of this article is given and related algorithmic questions are raised. At the end of this article are two Appendices. The first consists of the proofs of the Propositions discussed in Section 3 and the second gives a relevant algorithm.

It is intended that Section 3 should constitute a paradigm for relating the effect of other enzyme activities, such as those mentioned in Section 6, to the structure of the associated languages of potential DNA molecules. We suggest Salomaa (1985) and Hopcroft and Ullman (1979) as general references for formal language theory and Lewin (1983, 1987) and Watson *et al.* (1983) as general references for DNA behavior. Legerski and Robberson (1985) may be consulted regarding the action of ligase enzymes.

It is hoped that the line of research initiated here will interest biologists who have not previously found reason to be concerned with formal language theory. The situations modeled are usually ones in which the enzymes and the string operations they provide are known. Consequently the models usually apply directly only *in vitro* although some results may be suggestive for the *in vivo*

situation. The present article provides illustrations of two directions of development that may be of interest: 1. This type of work determines the set of possibilities for the outcome of *in vitro* activities. This is exemplified by the Theorem that concludes Section 3. Interpretation of this result provides a characterization of the set of possible DNA molecules that can arise in specified circumstances. Although the conditions appearing in this theorem use the special terminology developed here, the second condition of the theorem has a simple non-technical meaning that is made clear by the paragraph following Proposition 2 of Section 3. 2. The relationships between or among the transforming activities made possible by different enzymes or combinations of enzymes may be made precise or at least conceptually clarified. This is exemplified by the discussion in Section 7 of the relation between general recombination (which provides the required uniform splicing) and *all* other forms of recombination (known and unknown). The italicized statements in Section 7 are in one sense the most satisfying in this article for the following reason: With uniform splicing replaced by general recombination, these statements do not contain the technical terms introduced here. Thus they convey information that has now been established in a formal sense, but which has meaning independent of the formalism. The results discussed here are certainly only modest formal achievements. It is hoped that future developments will provide much stronger results following the two given paradigms and yield wholly new paradigms as well. Finally, it should not go unnoticed that these formal models and the related algorithms provide the basis for software that will allow computer experimentation with the recombination processes formalized.

2. *The Language Associated with a Set of DNA Molecules.* First we choose an appropriate alphabet of symbols in terms of which we can model each double-stranded DNA molecule as a string over this alphabet. Let A, C, G and T denote the four deoxyribonucleotides that incorporate adenine, cytocine, guanine and thymine respectively. Consider for discussion the small hypothetical molecule:

A G C T A T C C T G A C C A T G A A T C G C
T C G A T A G G A C T G G T A C T T A G C G.

We wish to regard this molecule as one single string, not as two strings that are bound together in parallel. (As yet formal language theory does not deal directly with linked pairs of strings.) Consequently our alphabet must consist of the four complex symbols:

A	C	G	T
T	G	C	A.

To save space we will denote these four elements of our alphabet by: [A/T], [C/G], [G/C] and [T/A], respectively. Thus our alphabet will be the set $D=\{[A/T], [C/G], [G/C], [T/A]\}$. Note that our alphabet is not the set {A, C, G, T} which would be appropriate for discussing single-stranded DNA.

By an *alphabet* in the general sense of formal language theory we mean a finite, non-empty set. Let A be any such alphabet. Then A^* denotes the set of all finite strings of symbols that can be constructed using symbols in A. The *length* of a string x in A^* is the number of symbol occurrences in x. A^* is understood to contain a string of length zero, i.e. a string in which no symbol occurs. This string is denoted with the numeral 1 and is called the *null string*. A subset of A^* is called a *language* over A. Formal language theory is the theory of languages over alphabets in this abstract sense. A^* has a natural binary operation, called *concatenation*, under which it is closed: from strings x and y in A^* having lengths m and n, respectively, we may form a string of length $m+n$ by attaching a copy of y onto the right end of a copy of x. Then xy denotes the resulting string. When D denotes the four-letter alphabet for DNA, D^* denotes the set of all conceivable (primary structures of) DNA molecules.

By an *involution* of A^* we mean a function $f: A^* \to A^*$ for which $f(f(x))=x$ and $f(xy)=f(y)f(x)$ hold for all x and y in A^*. An involution is necessarily a one to one mapping of A^* onto A^*. Consequently $f(a)$ must lie in A for every a in A. Moreover, f is determined uniquely by the values it takes on A since $f(a_1a_2 \ldots a_{n-1}a_n)=f(a_n)f(a_{n-1}) \ldots f(a_2)f(a_1)$ for any $a_1, \ldots, a_n$ in A. It is convenient to use the alternate notation for involutions obtained by defining $x'=f(x)$ for each x in A^*. Then we have the identities: $x''=x$ and $(xy)'=y'x'$. The DNA alphabet $D=\{[A/T], [C/G], [G/C], [T/A]\}$ has the natural involution: $[A/T]'=[T/A]$, $[C/G]'=[G/C]$, $[G/C]'=[C/G]$, $[T/A]'=[A/T]$. When dealing with DNA we will use the extension of this function $'$ to the whole of D^*. Thus for the small, hypothetical DNA molecule $x=$[G/C] [A/T] [C/G] [A/T] we have $x'=$[T/A] [G/C] [T/A] [C/G].

In modeling DNA molecules as strings, x and x' must be regarded as alternate and equivalent models for the same DNA molecule: DNA molecules exist in three-dimensional space and may be rotated around freely. Strings are conceived of as lying in a line and having a left to right orientation. *To compensate for this difference it is necessary and sufficient to keep the equivalence of x and x' as models in mind systematically.* It may seem that another alternate model for the molecule $x=$[G/C] [A/T] [C/G] [A/T] would be $y=$[A/T] [C/G] [A/T] [G/C]. But y and x are not models of the same molecule: recall that successive nucleotides in DNA are held together by phosphodiester bonds which introduce a form of directionality. Each of the two strands has so-called 5′ and 3′ ends. At each end of a double-stranded DNA molecule one strand has an open 5′ end and the other has an open 3′ end. Thus the two strands are oppositely oriented. In properly formed DNA the bonds join a 5′ end only to a

3′ end. We will use the usual convention of denoting our DNA molecules on a typed line with the top strand having an open 5′ end at the left and consequently an open 3′ end at the right. When this point is clear it allows one to see why strings x and y represent different molecules and confirms that x and x' represent the same molecule: The open 5′ ends of both x and x' are on a G and a T in the example above. The open 5′ ends of y are on an A and a C.

The use of the involution concept allows an especially clear discussion of palindromes. Let A be any alphabet and let $'$ be an involution. When no involution is explicitly given the involution defined for all a in A by $a' = a$ may be assumed. The following definition of a palindrome then coincides with the use of this term in formal language theory (where we may take $a' = a$ for all a in A) and in molecular biology (where we understand the effect of $'$ on the special alphabet D as defined above): a string x of even length is a palindrome if $x = yy'$ for a string y in A^*. A string x of odd length is a palindrome if $x = yay'$ for a symbol a in A and a string y in A^*. In the even length case the condition $x = yy'$ can be replaced by the simpler condition $x = x'$.

DNA exists not only as linear molecules but also as circular molecules. Conventional formal language theory deals only with linear strings. Consequently, since we wish to take a conservative approach to the problem of relating formal language theory to the study of DNA in this initial work, we will ignore circular DNA here. Closure of sets of DNA molecules under recombinational events such as the ligation required in Section 3 or the excision/insertion events as mentioned in Section 6 calls for a unified formal treatment of linear and circular DNA. We therefore look forward to the development in the future of an expanded theory of formal languages that will deal with linear and circular strings in a unified fashion. The theme of analysis introduced here provides motivation for such a theory. Note that at the dawn of this century Axel Thue considered circular as well as linear strings. Moreover, circular strings have continued as objects of study in the theory of circular codes. However, what we are suggesting here is that in the future a unified theory of mutually interacting circular and linear strings might be developed. *The present article deals only with linear double-stranded DNA in which no mismatched pairs occur.*

3. Languages Generated by Restriction Enzymes. A restriction enzyme cuts DNA molecules in a very specific way. We illustrate this by discussing the enzyme *Eco*RI. Let

...N N N N N N N G A A T T C N N N N N N N...
...N N N N N N N C T T A A G N N N N N N N...

be a DNA molecule where N is a variable used to denote any arbitrary deoxyribonucleotide. *Eco*RI operates only at six-term sequences of exactly the

form shown in the center above. The effect is to cut the molecule into two pieces:

```
...NNNNNNNG          AATTCNNNNNNN...
...NNNNNNNCTTAA          GNNNNNNN....
```

Such staggered strands of DNA spontaneously reassociate if the ends are in the neighborhood of each other. (To maintain the reassociation a ligase must be available to seal the two bonds in the phosphodiester backbone at the G–A nucleotides. Throughout this article we assume the presence of an appropriate ligase wherever needed. Whenever small molecules are needed for such purposes as supplying energy they are also assumed present.) Suppose now that each of two different DNA molecules is cut by *Eco*RI into two such pieces as illustrated above. After reassociation the original two molecules may be reformed, but it is also possible that two new hybrid molecules will be formed by the left half of the first of the original molecules reassociating with the right half of the second and the right half of the first with the left half of the second. Let us move to a more convenient notation:

Let c=[G/C] and x=[A/T] [A/T] [T/A] [T/A]. If *Eco*RI is added to a collection of copies of two molecules of the form $ucxc'v$ and $pcxc'q$ (where u, v, p and q are arbitrary DNA segments) then after reassociation we may expect to find, in addition to molecules of the original forms, molecules of the forms $ucxc'q$ and $pcxc'v$. Many restriction enzymes are available and their behavior with respect to DNA molecules is quite similar to that of *Eco*RI, although their cleavage sites vary in sequence and some have more than one cleavage site. Appendix A of Watson *et al.* (1983) contains convenient tables of restriction enzymes and their associated sites or cleavage recognition sequences. With each restriction enzyme we associate a triple that we will call the *cleavage pattern of the enzyme*. This pattern will show the cleavage sequence plus the points of cleavage. For example, the cleavage pattern of *Eco*RI is ([G/C], [A/T] [A/T] [T/A] T/A], [C/G]). Notice from the diagram above illustrating the effect of cleavage by *Eco*RI that the single-stranded tails (called cohesive ends) each terminate with an A having an open 5′ end. Such single-stranded tails are called 5′ *overhangs*.

Consider three more enzymes, their cleavage sites, the nature of the cohesive ends they form, and our nōtation for the cleavage patterns they provide:

*Taq*I	*Sci*NI	*Hha*I
...NNTCGANN...	...NNGCGCNN...	...NNGCGCNN...
...NNAGCTNN...	...NNCGCGNN...	...NNCGCGNN...
...NNT CGANN...	...NNG CGCNN...	...NNGCG CNN...
...NNAGC TNN...	...NNCGC GNN...	...NNC GCGNN...
([T/A], [C/G] [G/C], [A/T])	([G/C], [C/G] [G/C], [C/G])	([G/C], [C/G] [G/C], [C/G]).

An important aspect of the cleavage process is not encoded in our cleavage

patterns: The fragments produced by *Taq*I and *Sci*NI can form hybrid molecules but the fragments produced by *Hha*I cannot recombine with the fragments produced by the other two. The single-stranded tails produced by the first two are 5′ overhangs, but for *Hha*I the single-stranded tails are 3′ overhangs. To build this last distinction into our formal model we will group our cleavage patterns into two sets: those associated with 5′ overhangs and those associated with 3′ overhangs. [It is typical but not universal for cleavage sites to be palindromes, as is the case for each of the four enzymes illustrated. Note that since $ucxc'v$ represents the same molecule as $(ucxc'v)' = v'c''x'c'u' = v'cxc'u'$, *Eco*RI may act on two copies of $ucxc'v$ to yield the long palindromic molecules $ucxc'u'$ and $v'cxc'v$].

There are also restriction enzymes for which the cleavages produced are not staggered. These are said to create blunt ends. For example, *Alu*I has the cleavage pattern ([A/T] [G/C], 1, [C/G] [T/A]) which indicates that it cleaves as illustrated:

...NNNAGCTNNN... becomes ...NNNAG and CTNNN...
...NNNTCGANNN... ...NNNTC GANNN....

The ligases currently in use rejoin such blunt ends and consequently mixed splicing of blunt end segments occurs. Ligation of blunt ends and cleavages that produce blunt ends raise some minor questions about the model we are developing. These are best ignored until Section 6.

The behavior of DNA with respect to restriction enzymes suggests the following intuitively presented questions. Suppose we are given a finite set M of DNA molecules and a finite set N of restriction enzymes (nucleases); suppose also that as many copies as desired of any DNA molecule that is given or produced are always available. What is the nature of the language consisting of all DNA molecules that can arise through the action of the restriction enzymes in N (followed by reassociation) on the set M of DNA molecules and on any further DNA molecules produced? In particular, which molecules can potentially arise? We set down a definition to allow a formal treatment of these questions:

Definition. A *splicing system* $S = (A, I, B, C)$ consists of a finite *alphabet* A, a finite set I of *initial strings* in A^*, and finite sets B and C of triples (c, x, d) with c, x and d in A^*. Each such triple in B or C is called a *pattern*. For each such triple the string cxd is called a *site* and the string x is called a *crossing*. Patterns in B are called *left patterns* and patterns in C are called *right patterns*. The *language* $L = L(S)$ *generated by* S consists of the strings in I and all strings that can be obtained by adjoining to L $ucxfq$ and $pexdv$ whenever $ucxdv$ and $pexfq$ are in L and (c, x, d) and (e, x, f) are patterns of the same hand. A language L is a *splicing language* if there exists a splicing system S for which $L = L(S)$.

This definition allows a formal treatment of the intuitively stated question

above concerning the language formed by the interaction of a finite set M of DNA molecules with a finite set N of restriction enzymes: The splicing system associated with the intuitive question is $S=(D, I\cup I', B, C)$, where D is our usual four-symbol alphabet for duplex DNA, I is the set of strings over D that represent the primary structures of the molecules in M, B consists of the set of all patterns associated with the enzymes in the set N that either produce 5′ overhangs or produce blunt ends, and C consists of the patterns associated with enzymes in N that produce 3′ overhangs. For example, if *Eco*RI is in N then it contributes the pattern ([G/C], [A/T] [A/T] [T/A] [T/A], [C/G]) to B. If *Hha*I is in N then it contributes ([G/C], [C/G] [G/C], [C/G]) to C. If *Alu*I is in N then it contributes ([A/T] [G/C], 1, [C/G] [T/A]) to B. If *Hgi*AI is in N then it contributes the following four patterns to C: ([G/C], [A/T] [G/C] [C/G] [A/T], [C/G]), ([G/C], [A/T] [G/C] [C/G] [T/A], [C/G]), ([G/C], [T/A] [G/C] [C/G] [A/T], [C/G]), and ([G/C], [T/A] [G/C] [C/G] [T/A], [C/G]) as can be seen by consulting Appendix A of Watson *et al.* (1983).

Certain features of B and C need to be pointed out and the use of I' needs explanation; we have seen that x and x' in D^* must be interpreted as models of the same molecule. The convenient way to incorporate this identity of interpretations into our formalism is to adjoin $I'=\{w \text{ in } A^*: w' \text{ in } I\}$ to the set of initial strings. For splicing systems derived from models of the situations we are discusssing, C and D will also have an involutional property: If (c, x, d) is a pattern for an enzyme then (d', x', c') must also be a pattern of the same hand for this enzyme. [This can be seen from elementary physical considerations or by the following formal considerations: Suppose (c, x, d) is a pattern and we have $ud'x'c'v$. Then $ud'x'c'v$ models the same molecule as $(ud'x'c'v)'=v'cxdu'$. Thus cleavage at the site $d'x'c'$ via the pattern (d', x', c') is the same process as cleavage at the site cxd via the pattern (c, x, d)]. Consequently, for the splicing system $S=(D, I\cup I', B, C)$ derived from a set M of DNA molecules and a set N of restriction enzymes, we will automatically have $B=B'$ and $C=C'$ where for any set T of triples we define $T'=\{(d', x', c'):(c, x, d) \text{ in } T\}$.

The present study is focused on splicing systems for which the property of being the crossing of a site is passed from certain segments to subsegments (persists) when a splicing operation is performed:

Definition. Let $S=(A, I, B, C)$ be a splicing system. Then S is *persistent* if for each pair of strings $ucxdv$, and $pexfq$, in A^* with (c, x, d) and (e, x, f) patterns of the same hand: If y is a subsegment of ucx (respectively xfq) that is the crossing of a site in $ucxdv$ (respectively $pexfq$) then this *same* subsegment y of $ucxfq$ contains an occurrence of the crossing of a site in $ucxfq$.

Note that whether a splicing system is persistent or not is independent of the set I of initial strings. To clarify the concept of persistence we give immediately two elementary examples. A crucial example of a non-persistent system that has a chemical interpretation is given in Section 4.

1. Let $S=(\{c, d, x\}, I(\text{unspecified}), \{(c, x, d), (d, x, c)\}, \phi)$. Consider the strings cxd, dxc and the string cxc obtained from these by splicing. Here x is a subsequent of cx that is the crossing of a site in cxd. This same subsequent x does not contain an occurrence of the crossing of any site in cxc. Thus S is not persistent. If B and/or C is enlarged so that $B \cup C$ also contains either the set $\{(c, x, c), (d, x, d)\}$ or the set $\{(1, x, 1)\}$ then S becomes persistent.
2. Let $S=(\{c, d, x, y\}, I(\text{unspecified}), \phi, \{(d, x, y), (cx, y, d), (c, x, y)\})$. Consider the strings dxy, $cxyd$ and the string $dxyd$ obtained from these by splicing. Here y is a subsegment of xyd that is the crossing of a site in $cxyd$. This same subsegment y of $dxyd$ does not contain an occurrence of the crossing of any site in $dxyd$. Thus S is not persistent.

When restriction enzymes are chosen from Appendix A of Watson *et al.* (1983) the associated splicing system models are frequently persistent. If a single enzyme is chosen the result is persistent in all cases including those such as *Hgi*AI that have multiple cleavage sites. When several enzymes are chosen the resulting system will still often be persistent. Examples 3 and 4 of Section 4 show that this is not always the case.

It is important that an algorithmic procedure be available for deciding whether an arbitrarily given splicing system is persistent. Such a procedure is given here in Appendix B. The following special class of splicing systems will also play a major role in this study.

Definition. A *null context* splicing system is a splicing system $S=(A, I, B, C)$ for which each cleavage pattern in B and each in C has the form $(1, x, 1)$.

Observe that each null context system is persistent.

Let $S=(A, I, B, C)$ be a null context splicing system. Let X be the set of crossings of B and Y the set of crossings of C. The association of each crossing x in X with its unique pattern $(1, x, 1)$ in B provides a one to one correspondence between X and B. We have a similar one to one correspondence between Y and C. Consequently we will adopt the lighter notation $S=(A, I, X, Y]$ for null context systems, where X and Y are the sets of crossings for B and C, respectively. We will continue to regard a null context system as a special case of a persistent system. In this notation the enzyme *Eco*RII, which has two cleavage sites both with 5′ overhangs, would provide $B=\{(1,$ [C/G] [C/G] [A/T] [G/C] [G/C], 1), (1, [C/G] [C/G] [T/A] [G/C] [G/C], 1)$\}$ or $X=\{$[C/G] [C/G] [A/T] [G/C] [G/C], [C/G] [C/G] [T/A] [G/C] [G/C]$\}$. Both C and Y would be empty.

M. P. Schutzenberger (1975) has already introduced a concept intimately associated with splicing into the mathematical literature. We paraphrase what we need of his concept:

Definition (Schutzenberger). With respect to a language over A, a string c in A^* is a *constant* if, whenever ucv and pcq are in the language, ucq and pcv are also in the language.

Note that when a string c is constant for a language L in A^* then all strings in A^*cA^* are also constant. In specifying a splicing system $S=(A, I, B, C)$ we are demanding that the sites be constants for the language being generated. Consequently when either B or C is not empty the set of constants is always infinite since it contains $\cup\{A^*cxdA^*:(c, x, d)$ in B or $C\}$. Note that a string s may be a constant for a splicing language even when no site occurs in s: let L be an arbitrary language and let s be a string for which there are no strings x and y for which xsy is in L. Consulting the definition of a constant reveals that s is a constant with respect to L in this vacuous sense. In Example 1 of Section 4 a specific string is given which is constant in this vacuous sense.

With each splicing system $S=(A, I, B, C)$ for which I is not empty we associate three non-negative integers $M(S)$, $LR(S)$ and $RR(S)$ as follows: For each s in I, let $L(s)$ be the length of the longest subsegment of s which does not contain the crossing of any occurrence of a cleavage site in s. Let $M(S)=1+\max\{L(s):s$ in $I\}$. Let the *left radius* of S be $LR(S)=\max\{$length $c:(c, x, d)$ in B or $C\}$ and the *right radius* of S be $RR(S)=\max\{$length $d:(c, x, d)$ in B or $C\}$.

PROPOSITION 1. *For the language $L(S)$ generated by a persistent splicing system S, every string of length $P=LR(S)+M(S)+RR(S)$ is constant.* (See Appendix A for proof.)

If the requirement that S be persistent is dropped from Proposition 1 the resulting statement is false. This is shown by Example 4 given in Section 4.

For P a positive integer, A^P is the set of all strings over the alphabet A that have length exactly P. For each pair of sets X and Y, $X\setminus Y=\{x$ in $X:x$ not in $Y\}$. With each language L and each positive integer P we associate the following four sets:

$S_P=L\setminus A^PA^*$.

$L_P=\{u$ in A^P: there is a v in A^* for which uv is in $L\}$.

$R_P=\{v$ in A^P: there is a u in A^* for which uv is in $L\}$.

$M_P=\{w$ in A^P: uwv is in L for some u, v in $A^*\}$.

Thus S_P is the set of strings in L of length less than P; L_P (respectively R_P) is the set of left (respectively right) end segments of length P of strings in L; and M_P is the set of segments of length P of strings in L. From these definitions P_P and R_P must be subsets of M_P. Apparently:

$$L\subseteq S_P\cup[(L_PA^*\cap A^*R_P)\setminus A^*(A^P\setminus M_P)A^*] \tag{1}$$

holds for *every* language L.

The following concept traces back to McNaughton and Papert (1971) but the present version of the definition follows A. De Luca and A. Restivo (1980):

Definition. A language L is *P-strictly locally testable* if equality holds for L in the containment relation (1) displayed above. A language is *strictly locally testable* if it is P-strictly locally testable for some P.

The following theorem is fundamental for our investigation of the structure of splicing languages. A converse form of this theorem is stated later.

THEOREM (De Luca and Restivo, 1980). *If for a language L over A all the strings in A^P are constants then L is $(P+1)$-strictly locally testable.*

From this Theorem and Proposition 1 we have immediately:

PROPOSITION 2. *For each persistent splicing system $S=(A, I, B, C)$, $L=L(S)$ is P-strictly locally testable for $P=LR(S)+M(S)+RR(S)+1$, i.e. for this value of P:*

$$L=S_P \cup [L_P A^* \cap A^* R_P) \backslash A^* (A^P \backslash M_P) A^*]. \tag{2}$$

This equation gives a clear structural description of $L=L(S)$. Moreover, when S_P, L_P, R_P and M_P are known, (2) provides us with a transparent procedure for deciding whether an arbitrarily given string over A lies in the language (i.e. in the DNA interpretation, whether a given DNA sequence can arise from a given set of DNA molecules in the presence of a given set of restriction enzymes). Let us elaborate on the significance of (2). Suppose that the four finite lists of strings S_P, L_P, R_P and M_P have been formed and consider the right side of (2). On the left of the union sign is the set of strings in L of length less than P. To the right is the set of strings in L of length P or more. The term $(L_P A^* \cap A^* R_P)$ demands that a string in L of length P or more must begin with an acceptable segment of length P and terminate with an acceptable segment of length P. Since M_P is the set of segments of length P that are acceptable for strings in L, the set $(A^P \backslash M_P)$ is the set of segments of length P that are unacceptable for strings in L. Consequently $A^*(A^P \backslash M_P)A^*$ is the set of all strings over A that contain at least one subsegment *unacceptable* for strings in L. Thus $(L_P A^* \cap A^* R_P) \backslash A^* (A^P \backslash M_P) A^*$ explicitly represents the set of strings of length P or more that initiate and terminate with acceptable segments of length P and contain no unacceptable segment of length P. Suppose now that a string s over A is given and we wish to determine whether it is in L. If the length of s is less than P we merely examine the list S_P to see if s is in this list. If s has length exactly P we merely check to see if s lies in both L_P and R_P. If s has length greater than P we examine the leftmost subsegment of length P to see if it lies in

L_P. We then examine the next subsegment of length P (i.e. symbol occurrences 2 through $P+1$) to see if it lies in M_P. We continue in the same manner with the next subsegment of length P and then the next and the next until we arrive at the rightmost subsegment of length P which we examine for membership in R_P. The nature of this test for determining whether long strings lie in L explains the choice of the term P-strictly locally testable: the test is based on collecting purely local information about the string.

Proposition 2 and the discussion above demonstrate the *conceptual simplicity* of the structure of the persistent splicing languages. Unfortunately practical limitations exist for many simple examples derived from DNA and restriction enzymes because $M(S)$, and therefore P, may be in the thousands. On the other hand in many such cases the structure of the language is again made transparent by encoding judiciously chosen (sometimes very long) subsegments of the initial strings as single symbols. It is hoped that the brief comments in Example 2 in Section 4 clarify our meaning here.

The expression (2) shows that strictly locally testable languages are regular and are therefore recognizable by finite automata. Within automata theory it is easily verified that a P-strictly locally testable language can be recognized by a finite deterministic automaton having $2|A|^P$ or fewer states, where $|A|$ denotes the number of symbols in the alphabet A of the language.

From Proposition 1 we know that for every persistent splicing language L there is a positive integer P for which every string of length P is constant. This suggests the possibility of using as crossings in a splicing system the set of all the strings of a fixed common length. The investigation of such systems provides the result that the persistent splicing languages are *precisely* the strictly locally testable languages and lays a convenient background for remarks about general recombination in Section 5 and related matters in Section 7.

Definition. A *uniform splicing system* is a null context splicing system $S=(A, I, X, X)$ for which there is a positive integer P such that $X=A^P$. A language L is a *uniform splicing language* if there is a uniform splicing system S for which $L=L(S)$.

Observe that for $X=A^P$, the languages generated by the following three splicing systems are identical: (A, I, X, X), (A, I, X, ϕ) and (A, I, ϕ, X). Consequently for a uniform splicing system (A, I, X, X) we lighten the notation to (A, I, X).

Let L be a language with respect to which all strings of length P are constant. By the Theorem of De Luca and Restivo we know that L is $(P+1)$-strictly locally testable and therefore that it is regular and is recognized by a finite deterministic automaton. Any such automaton can be reduced to a similar automaton that recognizes L and has the smallest number of states. Any two such minimal automata that recognize L are isomorphic and therefore have the same number of states.

PROPOSITION 3. *Let L be a language over A with respect to which all strings in A^P are constant. Let N be the number of states in a minimal automaton recognizing L. Then $L = L(S)$ for the uniform splicing system $S = (A, I, A^P)$ where I is the set of all those strings in L of length less than $P + 2N$.* (See Appendix A for proof.)

The following result is a converse form of the Theorem by the same authors stated above:

THEOREM (De Luca and Restivo, 1980). *For a P-strictly locally testable language L over an alphabet A, all strings in A^P are constants.*

The following theorem tells us that the uniform splicing systems do not generate languages that have special properties not shared by all persistent splicing languages. In fact, it tells us that every persistent splicing language is generated by some uniform splicing system.

THEOREM. *The following conditions on a language L over an alphabet A are equivalent:*

(i) *L is a persistent splicing language;*
(ii) *L is a strictly locally testable language;*
(iii) *the set of constants for L contains A^P for some P;*
(iv) *L is a uniform splicing language.*

Proof. (i) implies (ii) by Proposition 2. (ii) and (iii) are equivalent by the two Theorems of De Luca and Restivo. (iii) implies (iv) by Proposition 3. That (iv) implies (i) is immediate from the fact that a uniform splicing system is a special case of a persistent splicing system. ■

The class of persistent splicing systems is the broadest class of splicing systems for which we are able at this time to do two things: (i) give a complete determination of the structure of the associated languages (see the Theorem above); and (ii) give a procedure for deciding whether any given splicing system is persistent (see Appendix B).

In Section 4 an example is given of a splicing language that is not strictly locally testable.

4. Examples.

Example 1. Let $S = (A, I, B, C)$ where $A = \{c, x\}$, $I = \{cxcxc\}$, $B = \{(c, x, c)\}$ and $C = \phi$. Since there is only one cleavage site, cxc, it is easily confirmed that S is persistent. [In order to provide a biochemical interpretation of this example, consider the restriction enzyme *Cla*I which has the cleavage pattern ([A/T][T/A], [C/G][G/C], [A/T][T/A]) and leaves 5′ overhangs on its fragments. We may regard the element c in our alphabet as a

higher order symbol for the two-symbol string [A/T] [T/A] and the element x as a higher order symbol for [C/G] [G/C]. With this interpretation the language $L = L(S)$ generated by our example represents all molecules that can potentially arise from the action of *Cla*I (and an appropriate ligase, as always) on (copies of) the molecule [A/T] [T/A] [C/G] [G/C] [A/T] [T/A] [C/G] [G/C] [A/T] [T/A]. Regardless of this interpretation we deal with the two-symbol alphabet for which, for example, $cxcxc$ has length 5 (not 10).] This example has been chosen to be simple enough that a convenient algebraic expression can be given for the language: the initial string can be written both as $ucxc$ and $cxcv$ where $u = cx$ and $v = xc$. Since the pattern (c, x, c) is in B and both $ucxc$ and $cxcv$ are in L, the two strings $ucxcv$ and cxc must also be in L. Since $ucxcv = ucxcxc = u^2cxc$ and $cxcv$ are in L, u^2cxcv must also be in L. Since $u^2cxcv = u^2cxcxc = u^3cxc$ and $cxcv$ are in L, u^3cxcv must also be in L. This process can be continued indefinitely. Thus L contains $\{(cx)^n cxc : n \geq 0\}$. Since this latter set is apparently closed with respect to the operation provided by the single pattern (c, x, c), $L = \{(cx)^n cxc : n \geq 0\}$.

For this system the only crossing is x. The length of the longest subsegment of the initial string that does not contain the crossing of a site occurring in the initial string is 1. (There are three such segments each being an occurrence of $a\,c$.) Thus $M(s) = 1 + 1 = 2$. The radii are $LR(S) = 1$ and $RR(S) = 1$. From Proposition 2 it follows that every string of length 4 is a constant for L.

That strings of length 4 are constants is a consequence of the fact that every segment of length 4 of every word in L contains a site. The string xcx of length 3 contains no site, so the 4 is in a weak sense sharp. However strings can be constant for reasons that are less obvious than that they contain sites. All strings except 1 and c are constant for L: Since cxc and cxc are in L and c is not in L, c is not a constant for L. Likewise since L contains $1cxc$ and $cxc1$ but not 1, 1 is not a constant for L. However, on examining the expression for the complete language L, we verify from the definition of a constant that all other strings in A^* are constant. The string cc, for example, vacuously satisfies the condition for being a constant since it does not occur in any string in the language. *Thus the relation between sites and constants is not as simple as one might think.* [This distinction can be given an interesting biochemical interpretation: A string s may be constant because it contains an occurrence of a site of a restriction enzyme that is present, but a string t may be constant merely because of the exact global nature of the language of possible molecules (as was the case of cc in the present example). Suppose that a new enzyme is added (that does not conflict with the previous enzymes). In general the language of possible molecules will now be different. The string s will still be a constant because it contains the site of the restriction enzyme that is still present. However, t may no longer be a constant.] Since we have seen that all strings of length 2 are constant for L, the first Theorem of De Luca and Restivo

in Section 3 asserts that L is 3-strictly locally testable. Examination of the expression for L is not only 3-strictly locally testable but 2-strictly locally testable where $S_2=\phi$, $L_2=\{cx\}$, $R_2=\{xc\}$, and $M_2=\{cx, xc\}$.

From the expression for L the minimal automaton recognizing L is easily constructed and found to have $N=5$ states. Proposition 3 now asserts that $L=L(U)$ for the uniform splicing system $U=(\{c, x\}, J, \{cc, cx, xc, xx\})$ where J consists of all strings in L of length $<P+2N=2+2(5)=12$. Thus $J=\{cxc, cxcxc, cxcxcxc, cxcxcxcxc, cxcxcxcxcxc\}$. Examination of L confirms that $J=\{cxcxc\}$ is adequate.

Example 2. Suppose we have (copies of) the single molecule $d^{150}xd^{750}xd^{250}$ where d denotes [C/G] [G/C], x denotes the cleavage site of *Eco*RI which is [G/C] [A/T] [A/T] [T/A] [T/A] [C/G] and the exponents, 150, 750 and 250 denote repetitions of d. In contrast with the first example the site occurrences do not overlap here. As cleavages and ligations take place it is clear from an inspection of the initial molecule and the given site that site occurrences of future molecules will not overlap. Let us encode the powers of d as follows for our convenience: $a=d^{150}$, $b=d^{750}$, $c=d^{250}$. To keep this example short we have chosen d to be a palindrome and therefore a, b and c are also palindromes. On the other hand our initial molecule, which is now modeled by the string $axbxc$, is not a palindrome. However $(axbxc)'=c'x'b'x'a'=cxbxa$. We let S be the null context system $S=(\{a, b, c, x\}, \{axbxc, cxbxa\}, \{x\})$. Then $L=L(S)$ is the resulting model for the set of possible DNA molecules. Here $M(S)=2$, $LR(S)=0$, and $RR(S)=0$. A simple expression for L is again available: $L=\{a(xb)^nxa: n\geq 0\}\cup\{a(xb)^nxc: n\geq 0\}\cup\{c(xb)^nxa: n\geq 0\}\cup\{c(xb)^nxc: n\geq 0\}$.

Example 3. Let $S=(D, I, B, C)$ where D is our usual four-letter alphabet, I may go unspecified, B consists of the patterns provided by *Eco*RI, *Taq*I, *Sci*NI and *Alu*I and C contains the single pattern provided by *Hha*I. (These five patterns were given in Section 3.) By a careful comparison of the sequences occurring in these five sites it is found that this system is not persistent, but that it becomes persistent on deleting either the pattern for *Taq*I or the pattern for *Sci*NI. These two patterns are in conflict. Consider the strings [T/A] [C/G] [G/C] [A/T] and [G/C] [C/G] [G/C] [C/G] and the string [T/A] [C/G] [G/C] [C/G] obtained from the first two by splicing. Let y be the segment [C/G] [G/C] which is the crossing of the site of the first pattern (also of the second). Observe that this segment in the third string does not contain the crossing of any site in that string. With either of these two patterns deleted, no further conflicts are possible.

Example 4. *A splicing language that is not strictly locally testable.* We shorten notation: for [A/T], [C/G], [G/C] and [T/A] we will write only A, C,

G and T with the understanding that these single symbols now abbreviate the elements of our alphabet D. Thus TAG is short for [T/A] [A/T] [G/C]. Let $S=(D, I, B, \phi)$ where $I=\{$AGATCT GCGCGT GGATCC, ACGCGT GGATCT GCGCGC$\}$ and $B=\{$(A, GATC, T), (G, GATC, C), (A, CGCG, T), (G, CGCG, C)$\}$. Note that there are only two initial strings. Symbols are being kept in groups of six for easy reading. The cleavage patterns have been chosen for chemical realism: They are, in order, the cleavage patterns of *Bgl*II, *Bam*HI, *Mlu*I and *Bse*PI. These four all leave 5′ overhangs.

Careful attempts to pattern match will reveal that no sites exist in the initial strings except at the ends and that the crossings occurring in the first string are incompatible with the crossings occurring in the second. With this clear it may then be seen that $L=L(S)$ falls naturally in two separate parts: splicing the first string with itself gives AGATCT GCGCGT GGATCT GCGCGT GGATCC which again has sites only at the ends. Continuing in this way yields $\{$AGATCT (GCGCGT GGATCT)^n GCGCGT GGATCC: $n\geq 0\}$. Splicing the second with itself gives ACGCGT GGATCT GCGCGT GGATCT GCGCGC which again has sites only at the ends. Continuing in this way yields $\{$ACGCGT GGATCT (GCGCGT GGATCT)^n GCGCGC: $n\geq 0\}$. The language L is the union of these two sets, one produced from each initial string. Observe that all the strings in the first set begin with AG and end with CC, whereas all the strings in the second set begin with AC and end with GC. It follows that none of the strings in the infinite set $\{$(GCGCGT GGATCT)^n: $n\geq 0\}$ is a constant since each such string occurs as a segment of a string in each of the two sets. (If such a string were constant a string beginning with AG and ending with GC would lie in L.) We have shown that for this splicing language there is no positive integer P for which all strings of length P are constant. For this reason, or direct examination, the language is not strictly locally testable. Note, however, that it is the union of two strictly locally testable languages. (This S is not persistent.)

5. General Recombination. Suppose that two DNA molecules have the form uxv and pxq where the common subsegment x is sufficiently long. Then in the presence of appropriate enzymes (in *E.Coli* this would include *Rec*A and others) these molecules can recombine to yield the two molecules uxq and pxv. Unlike the related activity allowed by restriction enzymes, this does not depend on any sequence specificity of x at all but only on x being sufficiently long. Thus there seems to be agreement that there is a sufficiently large positive integer P for which if x has length P then this process, called general recombination, can take place about x.

Suppose that an initial set M of DNA molecules is given and that enzyme activity supporting general recombination is present. Suppose, as we have previously, that as many copies as may be desired of any DNA molecule given

or produced are always available. Let L be the language that consists of the strings that represent the primary structures of all DNA molecules that may potentially arise by general recombination acting on the molecules in M and on any further molecules produced. Since for an appropriate P all strings of length P are constant, the first Theorem of De Luca and Restivo applies and gives the conclusion that L is $(P+1)$-strictly locally testable.

We note that recombination can take place between molecules ux_1v and px_2q not only when $x_1 = x_2$ and this common segment is long enough but also when x_1 and x_2 are not equal but sufficiently similar (homologous). Such splicings will produce imperfect molecules containing one or more mismatched pairs. As we consider the languages being modeled to consist only of those double-stranded molecules that do not contain mismatched pairs (and since we are not assuming the presence of repair enzymes), we may disregard such defective products. Nevertheless we do consider that this limits the significance of the remarks we have made here concerning general recombination.

6. Modeling Questions. The fact that the ligases in use can concatenate the blunt ends of DNA molecules (as mentioned in Section 3) is worth a special clarifying comment. If we have an initial set I of DNA molecules, a ligase, and *no* restriction enzymes at all then, in view of the blunt end ligation capability, the appropriate formal model of the language generated would be I^*. We did not wish to build such a universal closure under concatenation into our splicing language model. To avoid the resulting dissonance between our modeling process via splicing systems and this blunt end ligation capability we prefer to assume that in all situations in which we are using splicing systems to model the effect of restriction enzymes our initial DNA molecules have ends that are chemically modified to prevent such ligation. This restores I itself, rather than I^*, as the appropriate model in the trivial situation described above and has the correct effect when enzymes are present.

The adjustment of the previous paragraph provides an adequate background for one further question concerning our modeling procedure. As mentioned in Section 3 we have intended our splicing system model to cover the action of such enzymes as *Alu*I which cleaves leaving blunt ends. We have stated that our language is taken as a model of the set of all well-formed, linear double-stranded DNA molecules that can potentially arise. However the result of a blunt-end cut in such a molecule is two separate molecules that are again well-formed, linear and fully double-stranded. An examination of the definition of a splicing system and its associated language will confirm that we have not in general provided for the two fragments produced by a blunt end cleavage to be covered by our model. The splicing system formalism implicitly requires that all fragments resulting from cleavage must be reconnected with companion fragments before being included in the language generated. To avoid this

dissonance between our modeling process and the possible presence of blunt end cleaving enzymes we make the following modification: The set of molecules being modeled consists of the well-formed, linear, fully double-stranded DNA molecules both ends of which possess the chemical modification assumed of the initial molecules.

In the remainder of this section we consider additional types of recombinant behavior and suggest that previously existing formalisms of language theory may be appropriate for modeling the generative power of these processes. Our comments here also serve to provide a background for Section 7. The languages generated by persistent splicing systems proved to be regular. The following considerations suggest that if we extend our modeling to cover these additional processes the resulting languages may cease to be regular.

Certain segments of DNA are known to move into and out of DNA molecules or to be replicated from one DNA molecule into another (or into another location within the original molecule). Among such segments are the *insertion sequences* and the *transposons* which we shall discuss first. For brevity we will discuss both under the single title of insertion sequence:

Suppose that we have two DNA molecules *uiv* and *ptq* where u, i, v, p, t and q denote subsegments of these molecules. Assume that the segment i is an insertion sequence. To assume this means that for certain (generally short) sequences called *targets*, an enzyme (or enzymes) exist that may cause a replication of i to be formed adjacent to a target sequence in a DNA molecule. We will make this formally clear by supposing that the t in *ptq* is a target for the i in *uiv*. Then the enzyme can act on *ptq* and *uiv* to convert *ptq* into *ptitq* while *uiv* remains unchanged. Note that the target t has also been replicated and that a copy of t now lies on each side of i in the new longer molecule *ptitq*. We will imitate this sort of process with a simple language generating formalism. We use as an alphabet the three symbols: t, (, and). Suppose that the three term sequence (t) is an insertion sequence with the one term string t as target. Let us suppose that we have initially the single string $t(t)t$. The effect of the insertion feature of (t) can be expressed by the production rule $t \rightarrow t(t)t$. By means of this production rule the initial string generates the language of all non-empty, well-formed strings of parentheses with an initial t adjoined, a terminal t adjoined and a t inserted between each pair of parentheses of either hand. This is a context free language that is not regular. [Here we have tacitly assumed that the target for an insertion sequence is unique. It is commonly supposed that for each insertion sequence there is a positive integer P (often between 4 and 9) for which each subsequence of length P may serve as a target.]

There are DNA sequences that have the ability (in the presence of appropriate enzymes) to insert into other DNA sequences and also to *leave* DNA sequences in which they appear. Let such sequences be called *excision sequences*. Such behavior could be represented in formal language theory in a

natural way in terms of *Thue systems* and the associated *congruential* languages which Book (1983, 1985) has recently surveyed. Such languages are not in general regular.

7. Uniform Splicing in the Presence of Further Enzymes. Suppose that we have a finite set M of DNA molecules, enzymes that guarantee uniform splicing about all strings of length P for some positive integer P (perhaps, for example, enzymes allowing general recombination), and various other unspecified enzymes that allow the activities of various insertion sequences, various excision sequences, and possibly other processes not yet described. Let L be the language of all possible sequences that can be formed in this setting. Can anything be said about the nature of L? Yes: regardless of the vagaries of the hypothesis, all strings of length P must be constant for L. Thus the first Theorem of De Luca and Restivo applies and tells us that L is $(P+1)$-strictly locally testable. Then L can be recognized by a finite deterministic automaton having $2|D|^{P+1} = 2(4^{P+1})$ or fewer states. Consequently, by Proposition 3, L is (or can be modeled as) the splicing language of $S = (D, I, D^P)$ where D is the usual four-symbol alphabet and where I consists of those strings in L of length less than $P+2(2(4^{P+1})) = P+4^{P+2}$. Thus *when enzymes are present that provide uniform splicing, they completely override all other enzymes in determining the global character of the language*, i.e. they insure strict local testability regardless of what other enzymes may be present. Also, since $L = L(S)$, it is seen that *after a fixed finite number of applications of the various recombination processes allowed by the total set of enzymes (enough to produce the set I) only the enzymes that provide uniform splicing about segments of length P are required to produce all further possible molecules.*

8. Summary and Algorithmic Questions. In this article we have illustrated what we mean by a formal characterization of the generative capacity of specified enzymatic activities operating on specified sets of double-stranded DNA molecules. This was done by concentrating on the action of restriction enzymes together with an appropriate ligase. This led to the development of the new generative formalism here called a splicing system. Examples of specific splicing systems modeling specific sets of restriction enzymes were given. Three special subclasses of splicing systems were defined and illustrated by DNA examples: persistent systems, null-context systems and uniform systems. The Theorem that closes Section 3 gave a full characterization of the persistent splicing languages by showing that they are precisely the strictly locally testable languages. The special structural clarity of this class of languages was delineated in Proposition 2 and the paragraph that follows the statement of that proposition. In Section 5 (with a reference to Section 7) it was shown that

the generative power of general recombination can be modeled by means of a uniform splicing system as long as we consider only the molecules produced that are purely double-stranded with no mismatched pairs. In Section 7 it was shown that in the presence of general recombination no matter what additional recombinant processes are enabled by the presence of further enzymes (assuming the additional ones do not block general recombination) the resulting language will be strictly locally testable.

On the basis of this paper further analyses of the generative power of sets of enzymes will be carried out in terms of the formalisms that have been introduced here. Thus the present article is intended to provide the link between future formal work and the imagery of molecular biology from which this work will derive and to which it will refer. Much of this will deal with the presentation of algorithms, some of which are needed to put the results of the present article on a constructive basis.

When a splicing system S is given the three integers $LR(S)$, $M(S)$, and $RR(S)$ defined in Section 3 are immediately computable and consequently so is the P defined by $P = LR(S) + M(S) + RR(S) + 1$ in Proposition 2. Consequently when S is persistent we know a value of P for which the language $L = L(S)$ is P-strictly locally testable. However, the present article does not give a means of calculating the four finite sets S_P, L_P, R_P and M_P that are required if equation (2) in Proposition 2 is to be a constructive expression for L. In joint work in progress now algorithmic solutions are sought for this and other related problems. *Beyond this lies the problem of characterizing the splicing languages that are not persistent.*

I thank Donald L. Robberson for reading and correcting from a biological perspective portions of an early version of this paper. I thank my colleague Ronald Gatterdam for his continued interest in this topic and for many provocative conversations. I thank also both Antonio Restivo and Aldo De Luca for helpful guidance to their results that are so fundamental to this work. Any errors in the current version are wholly my own.

APPENDIX A

Proofs of the propositions of Section 3. Let $S = (A, I, B, C)$ *be a splicing system. To provide a convenient background for the proof of Proposition* 1 *we partition the language* $L = L(S)$ into an infinite pair-wise disjoint family of finite subsets $I_0, I_1, I_2, \ldots, I_n, I_{n+1}, \ldots$:

Let $I_0 = I$. Let $I_1 = \{s \text{ in } A^* \backslash I_0$: there exist strings $ucxdv$ and $pexfq$ in I_0 and patterns (c, x, d) and (e, x, f) of the same hand, for which $s = ucxfq\}$. Suppose that $I_0, I_1, \ldots, I_n$ have been defined. Define $I_{n+1} = \{s \text{ in } A^* \backslash (I_0 \cup \ldots \cup I_n)$: there exist strings $ucxdv$ and $pexfq$ in $I_0 \cup \ldots \cup I_n$ and patterns (c, x, d) and (e, x, f) of the same hand, for which $s = ucxfq\}$. We have $L = \cup\{I_j : j \geq 0\}$ and I_m is disjoint from I_n when m and n are distinct.

Recall that $M(S)=1+\max\{L(s): s \text{ in } I\}$ where $L(s)$ is the length of the longest subsegment of s which does not contain the crossing of any occurrence of a site in s.

LEMMA. *Let $S=(A, I, B, C)$ be a persistent splicing system. Then each subsegment of length $M=M(S)$ of each string z in $L(S)$ contains the crossing of an occurrence in z of a site.*

Proof. From the definition of $M(S)$ the assertion holds for z in $I=I_0$. Suppose that the Lemma holds for all z in I_n for $n \leq k$, where k is any non-negative integer and let s be a string in I_{k+1}. Then there are strings $ucxdv$ and $pexfq$ in $\cup\{I_j: 0 \leq j \leq k\}$ with $s=ucxfq$ and patterns (c, x, d) and (e, x, f) of the same hand. Suppose that y is a subsegment of s having length M. Then at least one of the following three cases holds: (i) (respectively (ii)) y is contained in the exhibited occurrence of ucx (respectively xfq), hence in $ucxdv$ (respectively $pexfq$), and therefore by the induction hypothesis contains a crossing x_1 of an occurrence of a site in $ucxdv$ (respectively $pexfq$). By the hypothesis of persistence this occurrence of x_1 in y contains an occurrence of a crossing x_2 of a site in s. (iii) y contains the exhibited occurrence of x which by the hypothesis of persistence must contain the crossing of an occurrence of a site in s. Thus in any case the assertion of the Lemma holds for all z in I_{k+1}. By finite induction the Lemma holds for all z in $\cup\{I_n: n \geq 0\}=L(S)$. ■

Recall that the left radius (respectively right radius) of $S=(A, I, B, C)$ is $LR(S)=\max\{\text{length } c: (c, x, d) \text{ in } B \text{ or in } C\}$ (respectively $RR(S)=\max\{\text{length } d: (c, x, d) \text{ in } B \text{ or in } C\}$).

PROPOSITION 1. *For $S=(A, I, B, C)$ persistent, every string of length $LR(S)+M(S)+RR(S)$ is constant with respect to $L(S)$.*

Proof. Let s be a string of length $LR(S)+M(S)+RR(S)$. Case 1: Suppose that there is a string usv in $L(S)$. Let $s=ywz$ where length $y=LL(S)$, length $w=M(S)$, and length $z=LR(S)$. By the Lemma, w contains the crossing x of an occurrence in usv of a site cxd. By the choices of the lengths of y and z this occurrence of cxd lies in s. Since s contains a site, s is a constant relative to $L(S)$. Case 2: Suppose that s does not occur as a subsegment of any string in $L(S)$. Then s vacuously satisfies the definition of a constant relative to $L(S)$. ■

The proof of Proposition 2 was complete in Section 3.

PROPOSITION 3. *Let L be a language over A with respect to which all strings in A^P are constants. Let N be the number of states in a minimal automaton recognizing L. Then $L=L(S)$ for the uniform splicing system $S=(A, I, A^P)$ where I is the set of all those strings in L of length less than $P+2N$.*

Proof. ($L(S) \subseteq L$:) Since $I \subseteq L$ and all the strings in A^P are constants for L the desired inclusion follows.

($L \subseteq L(S)$:) Suppose this inclusion fails. Then among the strings in $L \setminus L(S)$ there is at least one string, s, for which no string in $L \setminus L(S)$ is shorter than s. This proof will be complete once we have demonstrated the contradiction that s is in $L(S)$. Let M be the length of s. Since s is in $L \setminus L(S)$, s is not in I and consequently $M \geq P+2N$. Let $s=a(1) \ldots a(M)$ with the $a(i)$ in A. Let $u=a(1) \ldots a(N)$, $x=a(N+1) \ldots a(M-N)$ and $v=a(M-N+1) \ldots a(M)$. Since length $u=N=$ length v and $M \geq P+2N$ it follows that the length of x is at least P and consequently that x *is a constant for* L. Let $Q(0), Q(1), \ldots, Q(M)$ be the sequence of states that the minimal automaton for L passes through as it reads s. Thus $Q(0)$ and $Q(M)$ are initial and final states, respectively.

In the list of $N+1$ states $Q(0), \ldots, Q(N)$ there must be a repetition. Let $Q(j)=Q(k)$ where $0 \leq j < k \leq N$. Let $u_1=a(1) \ldots a(j)a(k+1) \ldots a(N)$. Observe that u_1xv is in L since as it is read into the automaton for L the sequence of states passed through is $Q(0) \ldots Q(j)Q(k+1) \ldots Q(M)$.

In the list of $N+1$ states $Q(M-N), \ldots, Q(M)$ there must be a repetition. Let $Q(m)=Q(n)$ where $(M-N) \leq m < n \leq M$. Let $v_1=a(M-N+1) \ldots a(m)a(n+1) \ldots a(M)$. Observe that uxv_1 is in L since as it is read into the automaton for L the sequence of states passed through is $Q(0) \ldots Q(m)Q(n+1) \ldots Q(M)$.

We now produce the desired contradiction: Since uxv_1 and u_1xv are in L and x is a constant for L, $s=uxv$ is in L. ■

APPENDIX B

Recognizing persistence. Fundamental for the present article is the fact that there is an algorithmic procedure for deciding whether any given splicing system $S=(A, I, B, C)$ is persistent. We present one such algorithm here. The presentation given is chosen for maximum clarity of the algorithm and its validity. Efficiency has not been considered and no proof of validity is given:

By the *site length*, $SL(S)$, of S we mean the length of the longest site in S.

Consider all ordered pairs of strings s_1cxds_2 and s_3exfs_4 for which:

(i) (c, x, d) and (e, x, f) are both in B or both in C;
(ii) length $s_1cx=$ length $s_3ex=SL(S)-1$; and
(iii) length $xds_2=$ length $xfs_4=SL(S)-1$.

For each such pair there are two procedures to carry out:

First procedure. Find all patterns (g, y, hk) for which:

(i) (g, y, hk) is in either B or C;
(ii) gyh is a suffix of s_1cx; and
(iii) k is a prefix of ds_2.

If there is no such pattern begin the second procedure.
If there are such patterns then *every such pattern must be treated as follows*:
Decide whether there is a factorization $y=y_1y_2y_3$ for which there is a pattern (p, y_2, q) for which:

(i) (p, y_2, q) is in either B or C;
(ii) py_2y_3h is a suffix of s_1cx;
(iii) q is a prefix of y_3hfs_4.

If there is no such pattern, S is not persistent and we may stop.

Second procedure. Decide whether there is a pattern (gh, y, k) for which:

(i) (gh, y, k) is in either B or C;
(ii) hyk is a prefix of xds_2; and
(iii) g is a suffix of s_1c.

If there is no such pattern begin the treatment of the next pair of strings.
If there are such patterns then *every such pattern must be treated as follows*:
Decide whether there is a factorization $y=y_1y_2y_3$ for which there is a pattern (p, y_2, q) for which:

(i) (p, y_2, q) is either in B or C;
(ii) hy_1y_2q is a prefix of xfs_4; and
(iii) p is a suffix of s_1chy_1.

If there is no such pattern, S is not persistent and we may stop.
If all the original pairs of strings have been treated completely and no stop order has been encountered then S is persistent. ■

LITERATURE

Book, R. 1983. "Thue Systems and the Church-Rosser Property: Replacement Systems, Specification of Formal Languages, and Presentations of Monoids." In: L. Cummings (Ed.), *Combinatorics on Words*, pp. 1–38. New York: Academic Press.

——— 1985. "Thue Systems and Word Problems." In: J. P. Jouannand (Ed.), *Rewriting Systems and Applications*, Lecture Notes on Computer Science 202, pp. 63–94. Springer.

Brendel, V. and H. G. Busse. 1984. "Genome Structure Described by Formal Languages." *Nucleic Acids Res.* **12**, 2561–2568.

De Luca, A. and A. Restivo. 1980. "A Characterization of Strictly Locally Testable Languages and Its Application to Subsemigroups of a Free Semigroup." *Information and Control* **44**, 300–319.

Eberling, W. and M. A. Jimenez-Montano. 1980. "On Grammars, Complexity, and Information Measures of Biological Macromolecules." *Mathematical Biosciences* **52**, 53–72.

Jimenez-Montano, M. A. 1984. "On the Syntactic Structure of Protein Sequences and the Concept of Grammar Complexity." *Bull. math. Biol.* **46**, 641–659.

Hopcroft, J. E. and J. D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation.* Reading, MA: Addison–Wesley.

Landau, G. M. and U. Vishkin. 1986. "Introducing Efficient Parallelism into Approximate String Matching and a New Serial Algorithm." *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, pp.* 220–230.

———, ——— *and R. Nussinov.* 1986. "An Efficient String Matching Algorithm with k Differences for Nucleotide and Amino Acid Sequences." *Nucleic Acids Res.* **14**, 31–46.

Legerski, R. J. and D. L. Robberson. 1985. "Analysis and Optimization of Recombinant DNA Joining Reactions." *J. mol. Biol.* **181**, 297–312.

Lewin, B. 1983. *Genes.* New York: Wiley.

———. 1987. *Genes III.* New York: Wiley.

Martinez, H. M. (Ed.) 1984. "Mathematical and Computational Problems in the Analysis of Molecular Sequences.' *Bull. math. Biol.* (Special Issue Honoring M. O. Dayhoff) **46**(4).

McNaughton, R. and S. Papert. 1971. *Counter-Free Automata.* Cambridge MA: M.I.T. Press.

Salomaa, A. 1985. *Computation and Automata.* Cambridge: Cambridge University Press.

Schutzenberger, M. P. 1975. "Sur Certaines Operations de Fermeture dans les Langages Rationnels." *Symposia Math.* **15**, 245–253.

Watson, J. D., Tooze, J. and Kurtz, D. T. 1983. *Recombinant DNA: A Short Course.* New York: Freeman.

RECEIVED 1.9.87
REVISED 7.17.87