

SPLICING TO THE LIMIT

ELIZABETH GOODE AND DENNIS PIXTON

ABSTRACT. We consider the result of a wet splicing procedure after the reaction has run to its completion, or limit, and we try to describe the molecules that will be present at this final stage. In language theoretic terms the splicing procedure is modeled as an H system, and the molecules that we want to consider correspond to a subset of the splicing language which we call the *limit language*. We give a number of examples, including one based on differential equations, and we propose a definition for the limit language. With this definition we prove that a language is regular if and only if it is the limit language of a reflexive and symmetric splicing system.

1. INTRODUCTION

Tom Head [4] invented the notion of a *splicing system* in 1987 in an attempt to model certain biochemical reactions involving cutting and reconnecting strands of double-sided DNA. Since then there have been many developments and extensions of the basic theory, but some of the basic questions have remained unanswered. For example, there is still no simple characterization of the languages defined by splicing systems.

The language defined by a splicing system is called the *splicing language*, and this corresponds to the set of molecular types that are created during the evolution of the system. In discussions several years ago Tom proposed analyzing the outcome of a splicing system in a way that is closer to what is actually observed in the laboratory. The idea is that certain molecules which appear during the biochemical reactions are *transient* in the sense that they are used up by the time the splicing experiment has “run to completion.” The molecules that are left at this stage can be termed the *limit molecules* of the system, and the corresponding formal language defined by the splicing system is called the *limit language*.

The first major problem is to properly define this limit language. One possibility is to try to use differential equations to model the set of molecules, and then to define the limit language in terms of the solutions. Although it is reasonably clear how to set up such equations, solving them is another matter, since the system of differential equations is non-linear and, in non-trivial cases, has infinitely many variables.

Another possibility is to try to avoid quantitative issues and to analyze the splicing process qualitatively, in the context of formal languages, but guided by our understanding of the actual behavior of the biochemical systems.

In this paper we present a number of examples from the second viewpoint, illustrating some of the features that we expect in a limit language. We also give some of the details of a simple example from the differential equations viewpoint, to illustrate the general approach.

We then propose a definition of limit language in the formal language setting. It is not clear that our definition covers all phenomena of limit behavior, but it is sufficient for the range of examples we present, and leads to some interesting questions. It is well known that splicing languages (based on finite splicing systems) are regular, and it is natural to ask whether the same is true for limit languages. We do not know the answer in general, but in the special case of a reflexive splicing system we can give a satisfying answer: The limit language is regular; and, moreover, every regular language occurs as a limit language.

We present this paper not as a definitive statement, but as a first step toward understanding the limit notion.

2. MOTIVATION

We review briefly the basic definitions; for more details see [6]. A *splicing rule* is a quadruple $r = (u_1, v_1; u_2, v_2)$ of strings, and it may be used to *splice* strings $x_1u_1v_1y_1$ and $x_2u_2v_2y_2$ to produce $x_1u_1v_2y_2$. A *splicing scheme* or *H scheme* is a pair $\sigma = (A, R)$ where A is the alphabet and R is a finite set of rules over A . A *splicing system* or *H system* is a pair (σ, I) where σ is a splicing scheme and $I \subset A^*$ is a finite set, the *initial language*. The *splicing language* defined by such a splicing system is the smallest language L that contains I and is closed under splicing using rules of σ . Usually L is written as $\sigma^*(I)$.

Associated to a rule $r = (u_1, v_1; u_2, v_2)$ are three other rules: its *symmetric twin* $\bar{r} = (u_2, v_2; u_1, v_1)$ and its *reflexive restrictions* $\dot{r} = (u_1, v_1; u_1, v_1)$ and $\ddot{r} = (u_2, v_2; u_2, v_2)$. A splicing scheme σ is called *symmetric* if it contains the symmetric twins of its rules, and it is called *reflexive* if it contains the reflexive restrictions of its rules.

We consider I as modeling an initial set of DNA molecule types, with σ modeling a test-tube environment in which several restriction enzymes and ligases act simultaneously. In this case the language L models the set of all molecule types which appear at any stage in this reaction. In this interpretation we can see that, in fact, representatives of all molecule types in L appear very quickly (although perhaps in very small concentrations) after the beginning of the reactions.

Here we want to consider a somewhat different scenario: We ask not for the set of all molecules that are created in this reaction, but for the set of molecules that will be present after the system has “reached equilibrium”. In other words, we expect that some of the molecular types will be used up in constructing different molecular types, so they will eventually disappear from the solution. We call the subset L_∞ of L corresponding to the remaining “final-state” molecules the *limit set* of the splicing system. In this section we consider several examples of such limit sets, without giving a formal definition.

2.1. Example. *Initial language $I = \{ab, cd\}$, rule $r = (a, b; c, d)$, together with its symmetric and reflexive versions. Then the splicing language is $L = \{ab, cd, ad, cb\}$ but the limit language is $L_\infty = \{ad, cb\}$.*

Discussion. One of the first wet-lab experiments validating the splicing model was described in [7]. This example is just the symbolic version of that experiment. The molecules ab and cd are spliced by rules r and \bar{r} to produce ad and bc ; they are also spliced to themselves by rules \dot{r} and \ddot{r} to regenerate themselves. The words ad and bc are “inert” in the sense that

they do not participate in any further splicings, and any inert words will obviously appear in the limit language. However, initial stocks of the initial strings ab and cd will eventually be used up (if they are present in equal numbers), so they will not appear in the limit language, and the limit language should be just $\{ad, cb\}$. We call the words that eventually disappear “transient”. Of course, if there are more molecules of ab than of cd present at the start then we should expect that there will also be molecules of ab left in the final state of the reaction.

Our definition of limit language will ignore this possibility of unbalanced numbers of molecules available for various reactions. Put simply, we will define the limit language to be the words that are predicted to appear if some amount of each initial molecule is present, and sufficient time has passed for the reaction to reach its equilibrium state, regardless of the balance of the reactants in a particular experimental run of the reaction. In the example above, one particular run of the reaction may yield molecules of ab at equilibrium, while another run may not yield molecules of ab at equilibrium, but rather molecules of cd . Both reactions, however, are predicted to yield molecules of ad and bc , and hence these molecules constitute the limit language. \square

Limit languages would be easy to analyze if they consisted only of inert strings. The following example involves just a slight modification to the previous example and provides a limit language without inert words.

2.2. Example. *Initial language $I = \{ab, cd\}$, rules $r_1 = (a, b; c, d)$ and $r_2 = (a, d; c, b)$, together with their symmetric twins and reflexive restrictions. Then there are no inert words and the splicing language and limit language are the same, $L = L_\infty = \{ab, cd, ad, cb\}$.*

Discussion. The molecules ab and cd are spliced by rules r_1 and its symmetric twin as in Example 2.1 producing ad and cb . These products in turn can be spliced using r_2 and its symmetric twin to produce ab and cd . As in Example 2.1 each molecule can regenerate itself using one of the reflexive rules. None of the strings are inert, nor do any of them disappear. Thus each molecule in L_∞ is considered to be in a reactive “steady-state” at equilibrium. Their distribution can be calculated using differential equations that model the chemical reactions for the restriction enzymes, ligase, and initial molecules involved. \square

The next example demonstrates the possibility of an infinite number of reactive steady state molecules at equilibrium.

2.3. Example. *Initial language $I = \{aa, aaa\}$, rule $r = (a, a; a, a)$. Then there are no inert words and the splicing language and the limit language are the same, $L = L_\infty = aa^+$.*

Discussion. It is easy to see that $L = aa^+$. Moreover, each string a^k in L is the result of splicing two other strings of L ; in fact, there are infinitely many pairs of strings in L which can be spliced to regenerate a^k . In effect, all copies of the symbol “ a ” are shuffled between the various molecular types a^k in L . We consider each word a^k of L to be in the limit language. This interpretation is buttressed by the example in Section 3, which calculates the limiting concentrations of the molecules in a very similar system and finds that all limiting concentrations are positive.

Our definition of limit language will avoid the detailed calculation of limiting distribution; in cases like this we will be content to note that any molecule a^k will be present in the limit. \square

The following illustrates a very different phenomenon: Molecules disappear by growing too big.

2.4. Example. *Initial language $I = \{abc\}$, splicing rule $r = (b, c; a, b)$. Then the splicing language is $L = ab^+c$ but the limit language is empty.*

Discussion. The calculation of the splicing language is straightforward; note that splicing $ab^k c$ and $ab^j c$ produces $ab^{k+j} c$. Hence abc is not the result of any splicing operation although it is used in constructing other molecules. Therefore all molecules of type abc will eventually be used up, so abc cannot appear in the limit language. But now, once all molecules of type abc have disappeared then there is no way to recreate $ab^2 c$ and $ab^3 c$ by splicing the remaining strings $ab^k c$, $k \geq 2$. Hence all molecules of types $ab^2 c$ or $ab^3 c$ will eventually be used up, so they cannot appear in the limit language. The remainder of L is analyzed similarly, using induction. \square

The H scheme in Example 2.4 is neither reflexive nor symmetric, so it is hard to justify this example as a model of an actual chemical process. In fact the next example shows that this phenomenon, in which molecules disappear by “converging to infinitely long molecules”, can also happen in reflexive and symmetric H systems. However, we shall see later (Corollary 5.2) that in the reflexive case the limit language is not empty unless the initial language is empty.

2.5. Example. *Initial language $I = \{abc\}$, splicing rule $r = (b, c; a, b)$ together with its symmetric twin and its reflexive restrictions. Then the splicing language is $L = ab^*c$ and the limit language is ac .*

Discussion. The calculation of the splicing language follows as in Example 2.4, and all molecules of type $ab^k c$ for $k > 0$ will again eventually disappear under splicing using r . The symmetric splice using \bar{r} generates the single inert string ac , which will eventually increase in quantity until it consumes virtually all of the a and c pieces. Thus this system involves an infinite set of transient strings and a finite limit language consisting of a single inert string.

This system could run as a dynamic wet lab experiment, and one would expect to be able to detect by gel electrophoresis an increasingly dark band indicating the presence of more and more ac over time. Over the same time frame we predict the strings of the type $ab^k c$ would get very long, and this would eventually cause these molecules to remain at the top of the gel. In theory, at the final state of the reaction, there would be one very long such molecule that consumed all of the b pieces. We might expect the dynamics of the system to make it increasingly difficult for molecules of $ab^k c$ to be formed as k gets very large. A wet splicing experiment designed to demonstrate the actual dynamics of such a system would be a logical step to test the accuracy of this model of limit languages. \square

It is essential for the dynamic models that are introduced in Section 3 that we use a more detailed version of splicing. The action of splicing two molecules represented as $w_1 =$

$x_1u_1v_1y_1$ and $w_2 = x_2u_2v_2y_2$ using the splicing rule $(u_1, v_1; u_2, v_2)$ to produce $z = x_1u_1v_2y_2$ is actually a three step process. A restriction enzyme first acts on w_1 , cutting it between u_1 and v_1 . The resulting fragments have “sticky ends” corresponding to the newly cut ends, in the sense that a number of the bonds between base pairs at the cut site are broken, so part of the cut end on one side is actually single stranded, with the complementary bases on the other fragment. Similarly the molecule w_2 is cut between u_2 and v_2 , generating two fragments with sticky ends. Finally, if the fragments corresponding to x_1u_1 and v_2y_2 meet in the presence of a ligase they will join at the sticky ends to form z . It is clear that any dynamic analysis of the situation must account for the various fragments, keeping track of sticky ends.

There are two primary techniques for this: cut-and-paste systems as defined by Pixton [9], and cutting/recombination systems as defined by Freund and his coworkers [2]. The two approaches are essentially equivalent and, in this context, differ only in how they handle the “sticky ends”.

We shall use cut-and-paste. In this formulation there are special symbols called “end markers” which appear at the ends of all strings in the system and are designed to encode the sticky ends, as well as the ends of “complete” molecules. The cutting and pasting rules are themselves encoded as strings with end markers. A cutting rule $\alpha z \beta$ encodes the cutting action $xzy \implies x\alpha + \beta y$, while a pasting rule $\alpha w \beta$ corresponds to the pasting action $x\alpha + \beta y \implies xwy$. Thus a splicing rule $(u_1, v_1; u_2, v_2)$ can be represented by cutting rules $\alpha_1 u_1 v_1 \beta_1$ and $\alpha_2 u_2 v_2 \beta_2$ and a pasting rule $\alpha_1 u_1 v_2 \beta_2$. In this case α_1 encodes the sticky end on the left half after cutting between u_1 and v_1 and β_2 corresponds to the sticky end on the right half after cutting between u_2 and v_2 , so the pasting rule reconstitutes $u_1 v_2$ when the sticky ends α_1 and β_2 are reattached. For details see [9].

2.6. Example. *This is a cut-and-paste version of Example 2.5. The set of endmarkers is $\{\alpha, \beta_1, \beta_2, \gamma, \delta\}$, the set of cutting rules is $C = \{\alpha ab\beta_1, \beta_2 bc\gamma\}$ and the set of pasting rules is $P = C \cup \{\beta_2 bb\beta_1, \alpha ac\gamma\}$, and the initial set is $\{\delta abc\delta\}$. Then the full splicing language is $L = \delta ab^* c\delta + \delta\alpha + \gamma\delta + \delta ab^* \beta_2 + \beta_1 b^* c\gamma + \beta_1 b^* \beta_2$, and the limit language is $L_\infty = \{\delta ac\delta\}$.*

Discussion. $\delta ac\delta$ is inert and is created from $\delta\alpha$ and $\gamma\delta$ by pasting. Thus the strings $\delta\alpha$ and $\gamma\delta$ will be eventually used up. These strings are obtained by cutting operations on the strings in $\delta ab^* c\delta + \delta ab^* \beta_2 + \beta_1 b^* c\gamma$, so these in turn are transient. The remaining strings are in $\beta_1 b^* \beta_2$, and these are never cut but grow in length under pasting using the rule $x\beta_2 + \beta_1 y \implies x b b y$. Hence these strings disappear in the same fashion as in Example 2.4. \square

One possibility that we have not yet discussed is that a DNA fragment may produce a circular string by *self pasting*, if both of its ends are sticky and there is an appropriate pasting rule. We use \hat{w} to indicate the circular version of the string w .

2.7. Example. *If we permit circular splicing in Example 2.6 then the limit language is $L_\infty = \{\delta ac\delta\} \cup \{\hat{b}^k : k \geq 2\}$.*

Discussion. The only strings that can circularize are in $\beta_1 b^+ \beta_2$, and if we self paste $\beta_1 b^j \beta_2$ using the pasting rule $\beta_2 bb\beta_1$ then we obtain \hat{b}^{j+2} . These circular strings cannot be cut, so they are inert. \square

Our final examples illustrate somewhat more complex dynamics.

2.8. Example. *Initial language $I = \{abb, cdd\}$ and splicing rules $r_1 = (ab, b; a, b)$, $r_2 = (c, d; a, b)$, $r_3 = (ad, d; a, d)$, together with their symmetric and reflexive counterparts. The splicing language is $L = ab^+ + ad^+ + cb^+ + cdd$ and the limit language is $L_\infty = ad^+ + cb^+$.*

Discussion. Rule r_1 is used to “pump” ab^+ from abb while rule \bar{r}_1 produces ab from two copies of abb . Rules r_2 and \bar{r}_2 splice copies of cdd and ab^k to produce cb^k and add . Rule r_3 generates add^+ from add , while \bar{r}_3 generates ad from two copies of add . The transient words cdd and ab^k are used up to produce add and cb^+ , and the inert words cb^+ are limit words. However all words of ad^+ are continually recreated from other words in ad^+ using rules r_3 , \bar{r}_3 and their reflexive counterparts (in the same manner as in Example 2.3), so these are also limit words. Thus, in this example there are infinitely many inert strings; infinitely many steady-state strings; and infinitely many transient strings.

It is not hard to extend this example to have several “levels” of transient behavior. \square

2.9. Example. *The splicing rules are*

$$r_1 = (a, b; c, d), \quad r_2 = (a, d; f, b), \quad r_3 = (a, e; a, e), \quad r_4 = (a, f; a, f)$$

together with their symmetric twins and reflexive restrictions, and the initial language I consists of

$$w_1 = cd, \quad w_2 = ad, \quad w_3 = afd, \quad w_4 = afbae, \quad w_5 = cbae, \quad w_6 = abae$$

Then the splicing language L is the same as I , but the limit language is $\{w_3, w_4, w_5, w_6\}$.

Discussion. We shall use a notation that will be made more precise later: For w and z in L we write $w \rightarrow z$ to mean that a splicing operation involving w produces z . It is easy to check that there is a cycle $w_2 \rightarrow w_3 \rightarrow w_4 \rightarrow w_5 \rightarrow w_6 \rightarrow w_2$ so all these words would seem to be “steady-state” words. On the other hand, w_1 can be generated only by splicing w_1 and w_1 using $\bar{r}_1 = (c, d; c, d)$, but w_1 also participates in splicing operations that lead to the w_2 – w_6 cycle. For example, w_6 and w_1 splice using r_1 to produce w_2 , so $w_1 \rightarrow w_2$.

Clearly then w_1 will be eventually used up, and so it should not be part of the limit language. We let $L_1 = L \setminus \{w_1\}$, and reconsider the limit possibilities. There are only two ways to produce w_2 by splicing: from w_2 and w_2 using $\bar{r}_1 = (a, b; a, b)$ or from w_6 and w_1 using r_1 . Since w_1 is not available in L_1 we see that the w_2 – w_6 cycle is broken. In fact, since w_2 is used in the splicing $w_2 \rightarrow w_3$, it must eventually disappear, so it is not part of the limit language. The remaining words do not form a cycle but they are connected as follows: $w_3 \rightarrow w_4 \rightarrow w_3$ using r_4 and $w_4 \rightarrow w_5 \rightarrow w_6 \rightarrow w_4$ using r_3 . Thus these words appear to be limit words; and, since the splicings in these cycles do not involve w_2 , they remain limit words after w_2 is removed.

With somewhat more work we can construct a splicing language in which the words w_k are replaced by infinite sublanguages W_k , but demonstrating the same phenomenon: The disappearance of a set of transients like W_1 can change the apparent limit behavior of the remaining strings. \square

3. DYNAMICS

We present here a dynamical model based on a very simple cut-and-paste system. The molecules all have the form F^j for $j > 0$, where F is a fixed fragment of DNA. We suppose that our system contains a restriction enzyme which cuts between two copies of F and a ligase which joins any two molecules at their ends. That is, the simplified cut-and-paste rules are just $(F^i, F^j) \iff F^{i+j}$. For simplicity we do not allow circular molecules.

The dynamics of our model will be parameterized by two rates, α and β . We interpret α as the probability that a given ordered pair A, B of molecules will paste to yield AB in a unit of time, and we interpret β as the probability that a cut operation will occur at a given site on a given molecule A in a unit of time.

Let $S_k = S_k(t)$ be the set of molecules of type F^k at time t , let $N_k = N_k(t)$ be the number of such molecules (or the concentration – i.e., the number per unit volume), and let $N = N(t) = \sum_{j=0}^{\infty} N_j$ be the total number of molecules. Consider the following four contributions to the rate of change of N_k in a small time interval of length Δt :

First, a certain number will be pasted to other molecules. A single molecule A of type F^k can be pasted to any other molecule B in two ways (yielding either AB or BA), so the probability of pasting A and B is $2\alpha\Delta t$. Since there are $N - 1$ other molecules the probability that A will be pasted to some other molecule is $2\alpha(N - 1)\Delta t$, and, since this operation removes A from S_k , the expected change in N_k due to pastings with other molecules is $-2\alpha(N - 1)N_k\Delta t$. Since N is very large we shall approximate this as $-2\alpha NN_k$. Note that this may seem to overestimate the decrease due to pastings when both A and B are in S_k , since both A and B will be considered as candidates for pasting; but this is correct, since if two elements of S_k are pasted then N_k decreases by 2, not by 1.

Of course pasting operations involving smaller molecules may produce new elements of S_k . For molecules in S_i and S_j where $i + j = k$ the same reasoning as above produces $\alpha N_i N_j \Delta t$ new molecules in S_k . There is no factor of 2 here since the molecule from S_i is considered to be pasted on the left of the one from S_j . Also, if $i = j$ then we actually have $\alpha(N_i - 1)N_i\Delta t$ because a molecule cannot paste to itself, and we approximate this as $\alpha N_i^2 \Delta t$. This is not as easy to justify as above, since N_i is not necessarily large; however, this approximation seems to be harmless. The total corresponding change in N_k is $\alpha \sum_{i+j=k} N_i N_j \Delta t$.

Third, a certain number of the molecules in S_k will be cut. Each molecule in S_k has $k - 1$ cutting sites, so there are $(k - 1)N_k$ cutting sites on the molecules of S_k , so we expect N_k to change by $-\beta(k - 1)N_k\Delta t$.

Finally, new molecules appear in S_k as a result of cutting operations on longer molecules, and, since Δt is a very small time interval, we do not consider multiple cuts on the same molecule. If A is a molecule in S_m then there are $m - 1$ different cutting sites on A , and the result of cutting at the j^{th} site is two fragments, one in S_j and the other in S_{m-j} . Hence, if $m > k$, exactly two molecules in S_k can be generated from A by cutting, and the total expected change in N_k due to cutting molecules in S_m is $2\beta N_m \Delta t$. Summing these gives a total expected change in N_k of $2\beta \sum_{m>k} N_m \Delta t$.

So we have the following basic system of equations:

$$(3.1) \quad N'_k = -2\alpha NN_k - \beta(k-1)N_k + \alpha \sum_{i+j=k} N_i N_j + 2\beta \sum_{m>k} N_m.$$

Define $M = \sum_k kN_k$. If μ is the mass of a single molecule in S_1 then $M\mu$ represents the total mass of DNA, so M should be a constant. We verify this from (3.1) as a consistency check:

We will ignore all convergence questions. We have $M' = \sum_k kN'_k$. Plugging in (3.1), we have four sums to consider, as follows:

$$\begin{aligned} -2\alpha \sum_k kNN_k &= -2\alpha MN, \\ \alpha \sum_k \sum_{i+j=k} kN_i N_j &= \alpha \sum_{i,j} (i+j)N_i N_j = 2\alpha MN, \\ -\beta \sum_k k(k-1)N_k &= -\beta \sum_m m(m-1)N_m, \\ 2\beta \sum_k \sum_{m>k} kN_m &= \beta \sum_m \left(2 \sum_{k<m} k \right) N_m = \beta \sum_m m(m-1)N_m. \end{aligned}$$

Adding these gives $M' = 0$.

Summing the equations (3.1) to get an equation for N yields four sums, which we treat similarly:

$$\begin{aligned} -2\alpha \sum_k NN_k &= -2\alpha N^2, \\ \alpha \sum_k \sum_{i+j=k} N_i N_j &= \alpha \sum_{i,j} N_i N_j = \alpha N^2, \\ -\beta \sum_k (k-1)N_k &= -\beta(M-N), \\ 2\beta \sum_k \sum_{m>k} N_m &= 2\beta \sum_m \left(\sum_{k<m} 1 \right) N_m = 2\beta(M-N). \end{aligned}$$

Hence

$$(3.2) \quad N' = -\alpha N^2 - \beta N + \beta M.$$

Equations of this form are solved explicitly in elementary differential equation texts; in this case the solution is

$$(3.3) \quad N = \bar{N} + \frac{\gamma C e^{-\gamma t}}{\alpha(1 - C e^{-\gamma t})},$$

where $\gamma = \sqrt{\beta^2 + 4\alpha\beta M}$, \bar{N} is the positive solution of

$$(3.4) \quad -\alpha\bar{N}^2 - \beta\bar{N} + \beta M = 0,$$

and C is determined by the initial conditions. From the solution (3.3) and a consideration of the direction field for the differential equation (3.2) it is clear that if $N(0) > 0$ then $N(t)$ is defined for all $t \geq 0$ and $N \rightarrow \bar{N}$ as $t \rightarrow \infty$.

We want to find the limiting values of the quantities N_k . First we have a simple result in differential equations.

3.1. Lemma. *Suppose a and b are continuous functions on $[t_0, \infty)$ and $a(t) \rightarrow \bar{a} > 0$ and $b(t) \rightarrow \bar{b}$ as $t \rightarrow \infty$. Then any solution of $Y' = -aY + b$ satisfies $\lim_{t \rightarrow \infty} Y(t) = \bar{b}/\bar{a}$.*

We omit the proof of the lemma, which uses standard comparison techniques for solutions of ordinary differential equations.

Now we replace $\sum_{m>k} N_m$ with $N - \sum_{i=1}^{k-1} N_i - N_k$ in (3.1) and rearrange to get

$$\begin{aligned} N'_k &= -2\alpha N N_k - \beta(k-1)N_k + \alpha \sum_{i+j=k} N_i N_j + 2\beta \left(N - \sum_{i=1}^{k-1} N_i - N_k \right) \\ &= -(2\alpha N + \beta(k+1))N_k + \alpha \sum_{i+j=k} N_i N_j + 2\beta \left(N - \sum_{i=1}^{k-1} N_i \right) \\ &= -a_k N_k + b_k. \end{aligned}$$

The point of this rearrangement is that the coefficients a_k and b_k only depend on the functions N_j for $j < k$ and on the known function N . Hence, if we solve the equations in sequence then the coefficients a_k and b_k can be treated as known functions of t .

It is convenient to write our results in terms of the asymptotic average molecular size $\bar{W} = M/\bar{N}$; using (3.4) we have $\beta\bar{W} = \alpha\bar{N} + \beta$. Clearly the coefficient a_k has the limit $\bar{a}_k = 2\alpha\bar{N} + \beta(k+1)$. We can find the limit \bar{N}_k of N_k from Lemma 3.1 once we know the limit \bar{b}_k of b_k . Using a routine but messy induction (which we omit) we show that $\bar{b}_k = \bar{a}_k \bar{N}_k$ where

$$\bar{N}_k = \lim_{t \rightarrow \infty} N_k(t) = \frac{\bar{N}}{\bar{W}} \left(\frac{\bar{W} - 1}{\bar{W}} \right)^{k-1}.$$

4. DEFINITIONS

We suppose we have a finite H scheme σ and an initial language I , defining the splicing language $L = \sigma^*(I)$. Given two words w, z in L we write $w \rightarrow_L z$ to mean that there is some word w' in L (possibly the same as w or z) so that either w, w' or w', w splices, using a rule of σ , to produce z . Then \rightarrow_L is a binary relation on L . As usual we define the transitive closure \rightarrow_L^+ and the reflexive transitive closure \rightarrow_L^* of \rightarrow_L . Precisely, $w \rightarrow_L^+ z$ means there is some finite sequence w_0, w_1, \dots, w_n of words of L so that $n \geq 1$, $w_0 = w$, $w_n = z$, and $w_k \rightarrow_L w_{k+1}$ for $0 \leq k < n$; and $w \rightarrow_L^* z$ means $w \rightarrow_L^+ z$ or $w = z$.

We say a word $w \in L$ is a *first-order limit* of L iff for any z in L for which $w \rightarrow_L^+ z$ we have $z \rightarrow_L^* w$. A word which is not a first-order limit is called *transient* in L ; in other words w is transient in L iff there is a word z in L so that $w \rightarrow_L^+ z$ but $z \rightarrow_L^* w$ is false. This notion of transience is meant to model the following: The splicing operations $w \rightarrow_L^+ z$ contributes some of the material of the molecule w to the molecule z , and this material is

never reassembled in a molecule of type w ; hence the material of w will eventually be used up.

We now define L_1 to be the set of first-order limit words of L , and we continue recursively to define the set L_k of k^{th} -order limits of L to be the set of first order limits of L_{k-1} . That is, we obtain L_k from L_{k-1} by deleting the words that are transient in L_{k-1} .

Finally we define the *limit language* as

$$L_\infty = \bigcap_{k=1}^{\infty} L_k.$$

The limit languages described informally in the examples in Section 2 all satisfy this definition. Example 2.9 shows the difference between limits of order 1 and order 2.

We will often use the following interpretation of the relation \rightarrow_L : We consider a directed graph G_L in which the vertices are the words of L , so that there is an edge from w to z if and only if $w \rightarrow_L z$. We call this the *splicing graph* of L , although it is not determined just by L but by the pair (L, σ) .

In this interpretation we can describe the limit language as follows: We start by determining the *strongly connected components* of G_L ; these are the maximal subgraphs C of the graph so that, for any vertices w, z of C , we have $w \rightarrow_L^* z$. Such a component is called a *terminal* component if there is no edge $w \rightarrow_L z$ with w in C and z not in C . The first order limit language L_1 consists of the vertices which lie in the terminal components, and the transient words in L are the vertices of the non-terminal components.

We then define a subgraph G_L^1 of G_L whose vertices are the vertices of the terminal components, so that there is an edge from w to z if and only if there is a splicing operation $(w, w') \implies z$ or $(w', w) \implies z$ using a rule of σ in which w' is in L_1 . Then, of course, L_∞ is the set of vertices of the intersection G_L^∞ of the chain of subgraphs constructed recursively by this process.

5. REGULARITY

We continue with the terminology of Section 4. It is well-known that the splicing language L is regular ([1], [8]), and it is natural to ask whether the limit language L_∞ is regular. We do not know the answer in general. However, we can give a satisfactory answer in the important special case of a *reflexive* splicing system. In Head's original formulation of splicing [4] both symmetry and reflexivity were understood, and there are good biochemical reasons to require that any H system that purports to model actual chemical reactions must be both symmetric and reflexive.

5.1. Theorem. *Suppose σ is a finite reflexive H scheme and I is a finite initial language. Then the limit language L_∞ is regular.*

Proof. First, let S be the collection of all sites of rules in σ . That is, the string s is in S if and only if there is a rule $(u_1, v_1; u_2, v_2)$ in σ with either $s = u_1v_1$ or $s = u_2v_2$. For each $s \in S$ we let $L_s = L \cap A^*sA^*$; that is, L_s consists of the words of L which contain s as a factor. We shall refer to a set of the form L_s as a *site class*. We also define $L_I = L \setminus A^*SA^*$. This is the set of *inert* words of L ; that is, the set of words which do not contain any sites.

Suppose $x, y \in L_s$. By reflexivity we may find a rule $r = (u, v; u, v)$ of σ so that $s = uv$. Write $x = x_1uvx_2$ and $y = y_1uvy_2$. Then splicing x and y using r produces $z = x_1uvy_2$, and splicing y and z using r produces y_1uvy_2 , or y . Hence we have $x \rightarrow_L z \rightarrow_L y$, and z is in L_s . We conclude that the subgraph of G_L with the elements of L_s as vertices forms a strongly connected subgraph of the vertices of G_L , and so it lies in one strongly connected component of G_L .

We conclude that each strongly connected component of G_L is either a singleton $\{w\}$ with $w \in L_I$ or a union of a collection of site classes. The first type of component is clearly terminal. Thus the set L_1 of first-order limit words is the union of L_I and some collection of site classes. Moreover, any site class L_s which appears in L_1 is still a strongly connected subset of the vertex set of G_L^1 . Hence we can use induction to extend this decomposition to see that each L_k is the union of L_I and a collection of site classes.

Since the sets $L_{k+1} \subset L_k$ and there are only finitely many site classes we see that the sequence L_k is eventually constant, so $L_\infty = L_k$ for all sufficiently large k . Hence L_∞ is the union of L_I and a collection of site classes. This is a finite union of sets, each of which is regular (using the regularity of L), so L_∞ is regular. \square

5.2. Corollary. *If I is not empty then L_∞ is not empty.*

Proof. This is clear from the proof, since there are a finite number of strongly connected components (aside from the inert words) at each stage, and so there are terminal components at each stage; and the recursive construction stops after finitely many steps. \square

It has been a difficult problem to determine the class of splicing languages as a subclass of the regular class, even if we restrict attention to reflexive splicing schemes; see [3]. Hence the following is rather unexpected.

5.3. Theorem. *Given any regular language K there is a finite reflexive and symmetric splicing scheme σ and a finite initial language I so that the limit language L_∞ is K .*

Proof. This is an easy consequence of a theorem of Head [5], which provides a finite reflexive and symmetric splicing scheme σ_0 and a finite initial language I so that $\sigma_0^*(I) = cK$, where c is some symbol not in the alphabet of K . In fact, every rule of σ_0 has the form $(cu, 1; cv, 1)$ for strings u and v .

Our only modification involves adding the rules $r_0 = (c, 1; 1, c)$, $\bar{r}_0 = (1, c; c, 1)$, $\dot{r}_0 = (c, 1; c, 1)$ and $\ddot{r}_0 = (1, c; 1, c)$ to σ_0 to define σ . This changes the splicing language to $L = c^*K$. To see this, note that if j and k are positive then c^jw and c^kz splice, using any of the new rules to produce c^mz , where, depending on the rule used, m ranges over the integers from 0 to $j + k$.

Now all words of $L \setminus K$ have the form $c^k w$ with $k > 0$, and these are transient since any such word splices with itself using \bar{r}_0 to produce the inert word $w \in K$. Hence the limit language is just K , the set of inert words. \square

6. CONCLUSION

We wrote this paper to introduce the notion of the limit language. This notion can be supported by qualitative reasoning, as in the examples in Section 2. However, the splicing

operation is not a realistic model of the actual chemical reactions; for this we need something like cut and paste operations to model the separate chemical actions of restriction enzymes and ligases. Thus Examples 2.6 and 2.7 give a better account of the molecules involved during the chemical reactions than Example 2.5.

We believe that a complete understanding of the limit language must wait until a suitable dynamical model has been developed and supported by experiment. It is not hard to formulate a generalization of the example in Section 3 to apply to arbitrary cut-and-paste systems, but it is much more difficult to solve such a generalization. We would hope eventually to derive regularity of the splicing language from the dynamical system, and then to define and analyze the limit language directly from the dynamical system.

Our preliminary definition of the limit language in Section 4 is simply an attempt to circumvent the considerable difficulties of the dynamical systems approach and see what we might expect for the structure of the limit language. We cannot yet prove that this definition coincides with the natural definition based on dynamical systems, but we believe that it will, probably with minor restrictions on the splicing scheme.

REFERENCES

- [1] K. Culik II and T. Harju. Splicing semigroups of dominoes and DNA. *Discrete Applied Mathematics*, 31:261–277, 1991.
- [2] R. Freund, E. Csuhaj-Varjú, and F. Wachtler. Test tube systems with cutting/recombination operations. In *Pacific Symposium on Biocomputing*, 1997.
- [3] Elizabeth Goode and Dennis Pixton. Recognizing splicing languages: syntactic monoids and simultaneous pumping. *to appear*, 2003.
- [4] T. Head. Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors. *Bulletin of Mathematical Biology*, 49(6):737–759, 1987.
- [5] T. Head. Splicing languages generated with one sided context. In G. Păun, editor, *Computing with Bio-Molecules. Theory and Experiments*, pages 158–181. Springer Verlag, Berlin, Heidelberg, New York, 1998.
- [6] T. Head, G. Păun, and D. Pixton. Generative mechanisms suggested by DNA recombination. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 2. Springer Verlag, Berlin, Heidelberg, New York, October 1996.
- [7] Elizabeth Laun and Kalluru J. Reddy. Wet splicing systems. In *Proceedings of the 3rd DIMACS Workshop on DNA Based Computers, held at the University of Pennsylvania, June 23 – 25, 1997*, pages 115–126, 1997.
- [8] Dennis Pixton. Regularity of splicing languages. *Discrete Applied Mathematics*, 69(1–2):101–124, August 1996.
- [9] Dennis Pixton. Splicing in abstract families of languages. *Theoretical Computer Science*, 234:135–166, 2000.

MATHEMATICS DEPARTMENT, TOWSON UNIVERSITY, TOWSON, MD 21252
E-mail address: egoode@towson.edu

DEPARTMENT OF MATHEMATICAL SCIENCES, BINGHAMTON UNIVERSITY, BINGHAMTON, NY 13902-6000