ELSEVIER

Theoretical Computer Science 234 (2000) 135–166

Theoretical Computer Science

www.elsevier.com/locate/tcs

Dennis Pixton [1]

Department of Mathematical Sciences, Binghamton University, P.O. Box 6000, Binghamton, NY 13902-6000, USA

Received June 1997; revised November 1997
Communicated by A. Salomaa

Abstract

We show that the iterated splicing operation determined by a regular H scheme (with some necessary restrictions) preserves membership in any full abstract family of languages. This involves translation of an H scheme into two alternative forms. The first form, which is closely related to the underlying biochemical operations, uses cutting and pasting rather than splicing. The second form uses matrices of languages, and in this formulation the splicing operation is translated into standard formal language operations (concatenation and quotient). Moreover, in the matrix formulation the splicing language itself may be expressed in terms of standard formal language operations, and this provides an algorithm for calculating the splicing language. As an application we use the cutting and pasting approach to extend the closure result to circular strings. © 2000 Elsevier Science B.V. All rights reserved.

1. Introduction

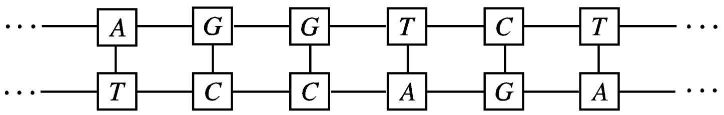
Tom Head [6, 7], in an analysis of certain biochemical processes involving DNA, introduced a new operation on strings called *splicing*. Since then his basic idea has been formalized in terms of generative mechanisms for formal languages called *splicing systems* or, more recently, *H systems*. These systems are now fairly well understood abstractly, and have been proposed as a theoretical foundation for some versions of the new field of DNA computing (see [1] or [10]).

As a generative mechanism an H system transforms an *initial language* into a *splicing language*. The object of this note is to improve on several earlier results showing that, under certain restrictions, the transformation from an initial language to a splicing language preserves membership in any full AFL (abstract family of languages; see Section 2 for a definition). The main improvement, besides substantial simplification, is an

E-mail address: dennis@math.binghamton.edu (D. Pixton)
[1] Research partially supported by NSF grant #CCR-9509831.

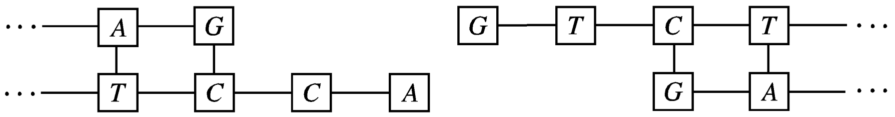
explicit representation of the splicing language in terms of the original using only operations familiar from elementary formal language theory. Our technique involves conversion of the splicing formulation into a form, using *matrices* of languages, which allows us to use simple algebraic techniques. Another feature of our technique is the translation of the standard formulation of H systems, under certain restrictions, into a form involving *cutting and pasting* which seems to be a better model of operations on DNA than the standard formulation. In fact, we shall show that suitably restricted H systems, cut and paste H systems, and matrix H systems can be freely translated into each other.

Before describing the mathematics we summarize, in non-technical terms, the biochemical operations that motivated Head's definition of splicing; more details may be found in his papers. A single molecule of "double-stranded" DNA may be visualized as a long "ladder", as follows:



The blocks A, T, G, C can only occur vertically in the pairs $[A/T]$, $[T/A]$, $[C/G]$, and $[G/C]$, so the DNA molecule may be considered a string over the alphabet consisting of these four pairs. (Actually, since the molecule is free to rotate in three dimensional space, it cannot be represented just as a string. Our model may be extended to respect the symmetries determined by such rotations; see [7] and [16].)

Splicing of DNA molecules occurs in two steps. First, a type of chemical called a "restriction enzyme" may cut the DNA molecule at a specific substring, called a "site", leaving two pieces with a damaged ladder structure at the cut end. The site and the actual shape of the cut ends are specific to each enzyme. For example, a particular enzyme may cut the string above at the site $[G/C][G/C][T/A][C/G]$, producing the following two pieces:



In this representation note that the middle of the site has been split into dangling single strands; the remaining unsplit parts of the site are called the left and right "contexts" of the site.

Since restriction enzymes work locally they can still operate on the pieces above, cutting them at appropriate sites as long as the sites do not touch the single stranded ends. Another type of chemical called a "ligase" mediates the reconnection of fragments by rejoining properly matched single-stranded ends. Thus in the presence of a ligase, two fragments as shown above can rejoin. This is not very interesting if these fragments are the result of the original cut, since the new molecule will match the original; rather the interest in splicing comes from the possibility that the two fragments rejoined by the ligase came from different types of DNA molecules, or even different fragments of the same molecule, so that the resulting double-stranded molecules will not be the same as the originals.

Head's translation of this process into formal language theory is the following. A site is a pair (u, u') of strings, and a splicing rule is a pair (u, u') and (v', v) of sites. Two strings may be spliced according to this rule if they can be factored as $xuu'y$ and $x'v'vy$, in which case the result of the splicing operation is $xuvy$. In other words, the first string is cut between u and u', the second is cut between v' and v, and the left fragment of the first string is glued to the right fragment of the second. A set of such rules is an *H scheme*, and an *H system* consists of an H scheme together with an initial language L. The eventual object of study is the *splicing language* generated by the H system; this is the smallest language containing L and invariant under splicing operations using the rules in the H scheme.

Gheorghe Păun noticed in [12] that such a splicing rule could be encoded as a single string $u\#u'\$v'\#v$ using the two auxiliary symbols # and \$, so that the set R of all applicable splicing rules is itself a language. This has become the standard for representing H schemes. The obvious question has the general form: What can you say about the splicing language if you restrict the languages R and L in some way?

The first such problem was solved by Culik and Harju in [2], who answered one of Head's original questions by showing that if the initial language is regular and the rule set is finite then the splicing language is regular. Culik and Harju used their theory of dominoes; algorithmic approaches to this result have been investigated by Gatterdam [5] and Kim [11].

Pixton [16] simplified Culik and Harju's proof, translating to the language of automata, and generalized the result to cover regular initial languages and certain (but not all) cases of regular rule sets. Moreover, with the same types of restricted regular rule sets, Pixton [15] showed that full AFL's are invariant under splicing. On the other hand, Păun [14] showed that for regular rule sets and regular initial sets the resulting splicing language could be essentially *any* RE language. These and related results are summarized in [8].

Păun's result shows that we must place some restriction on regular rule sets in order to prove closure. We can describe this restriction as follows. When two strings are spliced using the splicing rule $u\#u'\$v'\#v$ the substrings u' and v' do not appear in the final result, but the strings u and v do appear. For this reason we call u' and v' the *invisible* sites, and we call u and v the *visible* sites. Our restriction on rule sets is that there be only finitely many visible sites. This restriction can be interpreted in terms of the discussion of DNA splicing above as the requirement that the lengths of the single strands at cut ends, plus the attached context, should be bounded. (Pixton's results in [16, 15] essentially incorporated this restriction, but disguised under a different formulation of splicing. This is discussed at the end of Section 3.)

Our first alternate formulation of splicing is based on two operations, cutting and pasting, which are closer in spirit to biochemical operations on DNA than the single operation of splicing. Similar formulations were used implicitly in [15] and explicitly by Freund et al. in [3]. We call this alternate formulation a CPH scheme (for "cutting and pasting" H scheme). The cutting and pasting operations are facilitated by a finite number of special symbols, not in the original alphabet, called *end markers* and used only to terminate strings. Thus, a string which starts and ends with end markers but otherwise uses only the original alphabet is called an *end marked* string, and a language consisting only of end marked strings is called an *end marked language*. Cut operations are determined by an end marked language C of cutting sites. In a cut operation a site in C (minus the end markers) is removed from the original string leaving two fragments, and the end markers are affixed to the fragments at the "cut" ends. Pasting operations are determined by another end marked language P of pasting strings. The end markers on a pasting string determine the fragments which may be pasted, and when the fragments are joined the pasting string is inserted between them (deleting all markers except at the ends of the final string). In terms of DNA splicing the end markers introduced by cutting encode the dangling single strands at the cut ends (plus the attached context), and the pasting strings can be interpreted as the result of rejoining these single strands as part of the ligase operation.

Just as for H systems, we can define a corresponding splicing language for CPH systems. We say a CPH scheme is regular iff the sets C and P are regular. In Section 3 we show how to translate between regular H systems with finitely many visible sites and regular CPH systems and between their corresponding splicing languages.

We then reformulate CPH systems in terms of matrices of languages. A *matrix of languages* X is a finite rectangular array of languages, X_{ij}. The translation between a language with end markers and a matrix of languages is straightforward: If the end markers are enumerated as ε_i for $1 \leqslant i \leqslant n$ then an end marked language L corresponds to the matrix X where $X_{ij} = \{s : \varepsilon_i s \varepsilon_j \in L\}$. The data in a CPH scheme is just a pair of end marked languages, corresponding to the cutting sites and the pasting strings. These translate immediately into matrices: B and D, derived from the cutting sites after some modification; and P, corresponding to the pasting strings. We call such a triplet of matrices an MH scheme (for "matrix H scheme"). An initial end marked language for the CPH scheme translates directly to an initial matrix for the MH scheme.

In this formulation we do not need to introduce an analogue of the splicing operation on strings; rather, we use algebraic operations on matrices of languages. Specifically, we need the matrix operations corresponding to concatenation, left and right quotient, Kleene closure, and union (which we generally write using "$+$" rather than "$\cup$" to emphasize the algebraic structure). Here we are, in effect, treating such matrices as matrices of formal power series with boolean coefficients, in the sense of [17]. Definitions and basic properties of these operations are in Section 4. Now given an MH system we define the *splicing matrix* of the MH system as the smallest matrix solution S of the *splicing equation*:

$$S = L + B^{-1}S + SD^{-1} + SPS.$$

In Section 5 we show how to translate between CPH systems and MH systems and between splicing languages and splicing matrices.

Next, in Section 6, we show how to solve the splicing equation. In fact we can give an explicit formula for the solution in terms of matrix operations on the original data for the MH system. In general the solution is somewhat complicated, but to give the flavor of the general solution we exhibit the solution here in an important special case. If P is the identity matrix I (so $I_{jj} = \{1\}$ and otherwise $I_{jk} = \emptyset$) and B and D both contain I then

$$S = \bar{L}^{+} \quad \text{where } \bar{L} = ((S^{-1}B)^{*})^{-1}L((DS^{-1})^{*})^{-1}.$$

Notice that this is not quite an explicit formula for S, since the solution is in terms of $\bar{L}$, which is itself defined in terms of S. However, there is a simple algorithm at this stage for determining S; this is described, along with some sample calculations, in Section 7. Moreover, even without this algorithm, the form of the solution readily demonstrates that if $P = I$, B and D are regular, and the initial matrix L is in a full AFL $\mathscr{F}$ (meaning that each component of L is in $\mathscr{F}$) then S is in $\mathscr{F}$. Together with translations between the various splicing models, this provides the proof of our main result:

Closure Theorem. *Suppose $\mathscr{F}$ is a full AFL. The splicing language determined by a regular H system with finitely many visible sites and an initial language in $\mathscr{F}$ is in $\mathscr{F}$. The splicing language determined by a regular CPH system with an initial language in $\mathscr{F}$ is in $\mathscr{F}$. The splicing matrix determined by a regular MH system with an initial matrix in $\mathscr{F}$ is in $\mathscr{F}$.*

CPH systems and MH systems are generally easier to work with than H systems, and are in many ways more natural. As an illustration of this we consider another aspect of DNA splicing, following Head [7]. Since ligases act locally on two ends of DNA molecules and DNA molecules are somewhat flexible it is quite possible that *circular* molecules of DNA will be produced, even if the original molecules were all linear. Similarly, since restriction enzymes act locally it is quite possible that they will cut circular molecules of DNA, producing linear molecules. To model this we can consider a mixture of both circular and linear strings, and modify the definitions of the splicing, cutting, and pasting operations. We need certain further restrictions on H schemes (but not on CPH schemes) when dealing with circular strings. We also must require a further closure property for the AFL $\mathscr{F}$ (which is satisfied, at least, by the regular and context-free families). The necessary definitions are given in Section 8, where we prove:

Circular Closure Theorem. *Suppose $\mathscr{F}$ is a full AFL which is closed under cyclic closure.*

1. *If σ is a finite, symmetric and reflexive H scheme and M is a language of linear and circular strings in $\mathscr{F}$ then the splicing language generated by σ and M is in $\mathscr{F}$.*

2. *If κ is a regular CPH scheme and M is a language of end marked linear strings and circular strings in $\mathscr{F}$ then the splicing language generated by κ and M is in $\mathscr{F}$.*

In fact, most of the the proof of this theorem is the reduction from an H system to a CPH system. The proof in the CPH case is quite simple (and obvious): Just cut all the circular strings, let the language evolve strictly as a linear system so that the linear Closure Theorem can be applied, and then reconnect any linear strings that can produce circular strings. This should be contrasted with the argument in [16] for circular splicing (in the regular case) which is much more difficult than the corresponding proof in the linear case.

2. Preliminaries

We summarize here our notation and conventions from formal language theory.

By an *alphabet* A we shall always mean a non-empty finite set. We write 1 for the empty word in A^*. For languages L and M we write the *left and right quotients* of L by M as $M^{-1}L$ and LM^{-1}. We have $y \in M^{-1}L$ iff there is $z \in L$ and $x \in M$ with $z = xy$, with a similar definition for LM^{-1}. Since $M_1^{-1}(LM_2^{-1}) = (M_1^{-1}L)M_2^{-1}$ we can write the double quotient unambiguously as $M_1^{-1}LM_2^{-1}$.

As is customary we shall confuse singleton sets with their contents, writing, for example, $a^{-1}L$ or a^* instead of $\{a\}^{-1}L$ or $\{a\}^*$.

The notion of a *full abstract family of languages*, or *full AFL*, is defined in [9]. By this we mean a family of languages $\mathscr{F}$ with the following closure properties:
- The regular languages are in $\mathscr{F}$.
- If L and M are in $\mathscr{F}$ then LM, $L \cup M$ and L^* are in $\mathscr{F}$.
- If L is in $\mathscr{F}$ and R is regular then $L \cap R$, $R^{-1}L$ and LR^{-1} are in $\mathscr{F}$.
- If $L \subset A^*$ and $M \subset B^*$ are in $\mathscr{F}$ and $h : A^* \to B^*$ is a homomorphism then $h(L)$ and $h^{-1}(M)$ are in $\mathscr{F}$.

The most familiar full AFLs are the regular and context-free families.

We shall occasionally need the automaton formulation of regularity. For this we consider a finite directed graph G with a labeling function λ from the edges of G into $A \cup \{1\}$. Then λ extends to a map from the set of finite paths in G to A^*. If I and T are specified sets of vertices in G then an *accepting path* is a path in G from a vertex in I to a vertex in T. We define $L(G,I,T)$ to be the set of all words $\lambda(p)$ where p is an accepting path. It is well known that a language is regular if and only if it can be written as $L(G,I,T)$ for some G, I and T.

We shall need the following simple order-theoretic fact frequently.

Suppose $(\mathscr{S}, \leqslant)$ is a partially ordered set in which every countable set has a supremum. We write the supremum of $\{X, Y\}$ as $X \vee Y$ and we write the supremum of $\{X_p : p \geqslant 0\}$ as $\bigvee_{p=0}^{\infty} X_p$. We say an order-preserving function $\psi : \mathscr{S} \to \mathscr{S}$ is *continuous* iff for every non-decreasing sequence X_p we have $\psi(\bigvee_{p=0}^{\infty} X_p) = \bigvee_{p=0}^{\infty} \psi(X_p)$.

If $\phi : \mathscr{S} \to \mathscr{S}$ satisfies $\phi(X) \geqslant X$ for all X we define $\phi^0(X) = X$; $\phi^{n+1}(X) = \phi(\phi^n(X))$ for $n \geqslant 0$; and $\phi^*(X) = \bigvee_{n=0}^{\infty} \phi^n(X)$.

Lemma 2.1. *Suppose $(\mathscr{P}, \leqslant)$ is a partially ordered set in which every countable set has a supremum. Suppose $\psi : \mathscr{P} \to \mathscr{P}$ is order-preserving and continuous, and define ϕ by $\phi(X) = X \vee \psi(X)$. For $L \in \mathscr{P}$ define $S = \phi^*(L)$. Then S is the smallest solution of each of the following*:
1. $S \geqslant L \vee \phi(S)$.
2. $S = L \vee \phi(S)$.
3. $S \geqslant L \vee \psi(S)$.
4. $S = L \vee \psi(S)$.

Proof. Note that $\psi(S) = \psi(\bigvee_{p=0}^{\infty} \phi^p(L)) = \bigvee_{p=0}^{\infty} \psi(\phi^p(L))$ by continuity. A trivial induction shows that $L \vee \bigvee_{p=0}^{n} \psi(\phi^p(L)) = \phi^{n+1}(L)$. Hence, $L \vee \psi(S) \geqslant L \vee \bigvee_{p=0}^{n} \psi(\phi^p(L)) = \phi^{n+1}(L)$ for all n so $L \vee \psi(S) \geqslant \bigvee_{n=1}^{\infty} \phi^n(L) = S$. Conversely, $S \geqslant \phi^{n+1}(L) \geqslant \psi(\phi^n(L))$ for all n so $S \geqslant \bigvee_{p=0}^{\infty} \psi(\phi^p(L)) = \psi(S)$, and also $S \geqslant L$, so $S \geqslant L \vee \psi(S)$. Therefore $S = L \vee \psi(S)$, and we have verified that S satisfies part 4. It is trivial to check that the equation in part 4 implies parts 3, 2 and 1.

On the other hand, suppose S' satisfies part 3. Then $L \leqslant S'$ and for any $X \leqslant S'$ we have $\phi(X) \leqslant \phi(S') \leqslant S'$. Thus by induction we obtain $\phi^p(L) \leqslant S'$ for all p, so $S = \phi^*(L) \leqslant S'$. That is, S is the *smallest* solution of part 3, and hence also the smallest solution of part 4.

Next, for any X, the statements $X \geqslant L \vee \psi(X)$ and $X \geqslant L \vee X \vee \psi(X) = L \vee \phi(X)$ are equivalent, so it follows that S is also the smallest solution of part 1. As above we finally conclude that S is the smallest solution of part 2. $\square$

A common situation in which Lemma 2.1 applies occurs when $\mathscr{P}$ is a power set $\mathscr{P}(Z)$ ordered by inclusion and ψ is defined by $\psi(X) = F(X)$ where $F : Z \to X$ is a function. We need a somewhat more general situation, where ψ is not determined by a function but by a relation:

Lemma 2.2. *Suppose Z is a set and R is a subset of $\bigcup_{n \geqslant 0} Z^{n+1}$. Define $\psi : \mathscr{P}(\mathscr{Z}) \to \mathscr{P}(\mathscr{Z})$ so that $x \in \psi(X)$ iff there is $n \geqslant 0$ and there are $x_1, \ldots, x_n$ in X with $(x_1, \ldots, x_n, x) \in R$. Then ψ is order-preserving and continuous.*

Proof. *ψ is order preserving*: If $X \subset Y \subset Z$ and $x \in \psi(X)$ then $(x_1, \ldots, x_n, x) \in R$ for some $n \geqslant 0$ and $x_1, \ldots, x_n \in X$, and so $x_1, \ldots, x_n$ are in Y, implying $x \in \psi(Y)$.

ψ is continuous: Suppose $X_0 \subset X_1 \subset \ldots \subset Z$ and let $X = \bigcup_{p=0}^{\infty} X_p$. Since $X_p \subset X$ for all p and ψ is order-preserving we have $\psi(X_p) \subset \psi(X)$ for all p, so $\bigcup_{p=0}^{\infty} \psi(X_p) \subset \psi(X)$. On the other hand, if $x \in \psi(X)$ then there are $x_1, \ldots, x_n$ in X with $(x_1, \ldots, x_n, x) \in R$. For each j choose p_j so that $x_j \in X_{p_j}$, and let p be the maximum of $p_1, \ldots, p_n$. Then $X_{p_j} \subset X_p$ for all j, so $x_1, \ldots, x_n$ are in X_p and $(x_1, \ldots, x_n, x) \in R$, so $x \in \psi(X_p)$. This completes the proof of the opposite inclusion $\psi(X) \subset \bigcup_{p=0}^{\infty} \psi(X_p)$, as needed. $\square$

3. Splicing by cutting and pasting

We fix an alphabet A and two symbols # and \$ not in A. Following Păun [12] we define a *splicing rule* to be a string in $A^*\#A^*\$A^*\#A^*$. If $r = u\#u'\$v'\#v$ is a splicing rule we say u and v are the *visible sites* of r, uu' and $v'v$ are the *left* and *right sites*, and u' and v' are the *invisible sites*. Given this rule r and a pair of strings $w = xuu'y'$ and $w' = x'v'vy$, we say the string $z = xuvy$ is the result of *splicing* w *and* w' *using the rule* r. (More precisely, z is determined not by w and w' but by the indicated factorizations. Similar comments apply to similar definitions in the rest of the paper.)

An *H scheme* is a pair $\sigma = (A, R)$ where $R \subset A^*\#A^*\$A^*\#A^*$ is a set of splicing rules. We say σ is finite or regular iff R is a finite or regular language. If $L \subset A^*$ we define $\sigma(L)$ as the union of L and the set of all strings that can be obtained by splicing two strings of L using a rule in R. Clearly Lemmas 2.2 and 2.1 apply here. The language $\sigma^*(L)$ is the *splicing language* defined by the H scheme σ and the initial language L.

Now suppose we have a finite set E of *end markers* disjoint from A. An element of EA^*E is called an *end marked string*, and a set of end marked strings is called an *end marked language*. As a notational convention we shall employ exclusively Greek letters to refer to elements of E. Given $c = \delta u \varepsilon$ and a string $z = \alpha xuy\beta$, we say the strings $w = \alpha x \delta$ and $w' = \varepsilon y \beta$ are the results of *cutting* z *at the cutting site* c. On the other hand, given $p = \delta v \varepsilon$ and strings $w = \alpha x \delta$ and $w' = \varepsilon y \beta$, we say the string $z = \alpha xvy\beta$ is the result of *pasting* w *and* w' *using the pasting string* p. Note that cutting and pasting operations preserve EA^*E.

A *CPH scheme* (CP for "cut and paste") is a quadruple $\kappa = (A, E, C, P)$ where C and P are end marked languages. We say κ is finite or regular iff both C and P are finite or regular. If L is an end marked language and κ is a CPH scheme we define $\kappa(L)$ as the union of L and the set of all strings that can be obtained by cutting a string of L at a site in C or by pasting two strings of L using a string in P. Again Lemmas 2.2 and 2.1 apply to κ.

We shall need the following observation about iterated cutting:

Lemma 3.1. *Suppose $\kappa = (A, E, C, \emptyset)$ is a CPH scheme with no pasting operations, and suppose $L \subset EA^*E$. A string $\xi z \eta$ is in $\kappa^*(L)$ if and only if there is a string $\xi' xsztyη'$ in L so that*
1. *either $\xi' xs = \xi$ or there is a cutting site $\xi'' s \xi$ in C; and*
2. *either $ty\eta' = \eta$ or there is a cutting site $\eta t \eta''$ in C.*

In particular, $\kappa^*(L) = \kappa^2(L)$.

Proof. Strings $\xi z \eta$ satisfying the indicated property are clearly in $\kappa^2(L)$ and it is easy to check that the set of all such strings is closed under all cut operations. □

In the rest of this section we shall show how to convert between regular H schemes with finitely many visible sites and regular CPH schemes.

For the first translation we start with a regular H scheme $\sigma = (A, R)$ with finitely many visible sites, and we shall construct a CPH scheme in which each splicing operation can be replaced by two cuts and a paste. We use end markers at the cut operations to record enough information about the splicing rule so that we can then decide which paste operation to perform. We also mark the initial strings with a special end marker, so that we can pick out the final splicing results from the mixture of fragments and fully spliced strings formed by the CPH system.

Before giving the details of the translation we prove the following, which is the key to translating finitely many visible sites to a finite number of end markers.

Lemma 3.2. *Suppose $\sigma = (A, R)$ is a regular H scheme with finitely many visible sites. Then there are finitely many non-empty regular sets $R_1, \dots, R_N$ (with $N \geqslant 0$) whose union is R, satisfying the following: If $u\#u'\$v'\#v$ and $\bar{u}\#\bar{u}'\$\bar{v}'\#\bar{v}$ are in R_k then $u = \bar{u}, v = \bar{v}$, and $u\#u'\$\bar{v}'\#\bar{v}$ is also in R_k.*

Proof. If R is finite we can just use singleton sets for the R_k.

In general, choose an automaton representation $R = L(G, I, T)$, as in Section 2. We let E_0 be the set of edges in G which are labeled with \$. Now consider triples $w = (u, e, v)$ where u and v are in A^* and $e \in E_0$. For such w we let R_w be the set of rules of the form $u\#u'\$v'\#v$ which are labels of accepting paths in G which pass through e. Clearly, each R_w is a regular set and each rule in R lies in some R_w.

Suppose $r = u\#u'\$v'\#v$ and $\bar{r} = u\#\bar{u}'\$\bar{v}'\#v$ are both in R_w. If we follow an accepting path for r from I to the edge e and then continue from e to T along an accepting path for $\bar{r}$ then we obtain an accepting path labeled by $r' = u\#u'\$\bar{v}'\#v$, so r' is in R_w.

Since R has only finitely many visible sites there are only finitely many triples $w = (u, e, v)$ with $R_w \neq \emptyset$. These sets, relabeled as $R_1, \dots, R_N$, have the required properties. □

Now we fix a collection $\{R_k\}$ as given by the lemma, and we let u_k and v_k be the common visible sites shared by the rules in R_k. For each k we select markers δ_k and ε_k, and we let $E = \{\delta_k, \varepsilon_k : 1 \leqslant k \leqslant N\} \cup \{\nabla\}$, where ∇ is one additional marker. We define $C_R = \{\delta_k u_k u' \delta_k : 1 \leqslant k \leqslant N \text{and for some } v', u_k\#u'\$v'\#v_k \in R_k\}$. This is easily seen to be regular. We define C_L as the similar set of strings of the form $\varepsilon_k v' v_k \varepsilon_k$, and we set $C = C_L \cup C_R$. We finish by defining P as the finite set $\{\delta_k u_k v_k \varepsilon_k : 1 \leqslant k \leqslant N\}$. This defines the CPH scheme $\kappa = (A, E, C, P)$. The following says that κ generates the same splicing language as σ, if we are willing to identify A^* with $\nabla A^* \nabla$:

Proposition 3.3. *Suppose $\sigma = (A, R)$ is a regular H scheme with only finitely many visible sites, and define $\kappa = (A, E, C, P)$ as above. Then κ is a regular CPH scheme, finite if σ is finite, and, for any $L \subset A^*$, $\sigma^*(L) = \nabla^{-1}\kappa^*(\nabla L \nabla)\nabla^{-1}$.*

Proof. To show $\sigma^*(L) \subset S' = \nabla^{-1}\kappa^*(\nabla L \nabla)\nabla^{-1}$ we use Lemma 2.1: we need $S' \supset L$, which is obvious, and $\sigma(S') \subset S'$. So suppose $x = x_1 u_k u' x_2$ and $y = y_1 v' v_k y_2$ are in S' and can be spliced using $u_k\#u'\$v'\#v_k \in R_k$ to produce $z = x_1 u_k v_k y_2$. Since x and y are in S' we have $\nabla x \nabla$ and $\nabla y \nabla$ in $\kappa^*(\nabla L \nabla)$. Then the fragments $\nabla x_1 \delta_k$ and $\varepsilon_k y_2 \nabla$ obtained by cutting these strings at the sites $\delta_k u_k u' \delta_k$ and $\varepsilon_k v' v_k \varepsilon_k$ are in $\kappa^*(\nabla L \nabla)$. Hence, the result $\nabla x_1 u_k v_k y_2 \nabla = \nabla z \nabla$ of pasting these fragments using the pasting string $\delta_k u_k v_k \varepsilon_k$ is also in $\kappa^*(\nabla L \nabla)$. Therefore z is in $\nabla^{-1}\kappa^*(\nabla L \nabla)\nabla^{-1} = S'$.

For the proof of the opposite inclusion let κ_c be the CPH scheme $(A, E, C, \emptyset)$. We shall first show that $\kappa^*(\nabla L \nabla) \subset S'' = \kappa_c^*(\nabla \sigma^*(L) \nabla)$. Again, since $S'' \supset \nabla L \nabla$, this is just a matter of showing that $\kappa(S'') \subset S''$. Clearly, S'' is closed under cut operations, so suppose that $x' = \xi x \delta_k$ and $y' = \varepsilon_k y \eta$ in S'' are pasted to yield $z = \xi x u_k v_k y \eta$. By Lemma 3.1 we can find $\nabla \bar{x} \nabla = \nabla x_1 s x u_k u' x_2 \nabla$ in $\nabla \sigma^*(L) \nabla$ with either $\xi = \nabla$ and $x_1 s = 1$ or $\xi s \xi \in C$, and with $\delta_k u_k u' \delta_k$ in C. Corresponding to this cutting site there is a rule $u_k \# u' \$ v' \# v_k$ in R_k. Similarly there is $\nabla \bar{y} \nabla = \nabla y_1 \bar{v}' v_k y t y_2 \nabla$ in $\nabla \sigma^*(L) \nabla$ with either $\eta = \nabla$ and $t y_2 = 1$ or $\eta t \eta \in C$, and with $\varepsilon_k \bar{v}' v_k \varepsilon_k$ in C, with the corresponding rule $u_k \# \bar{u}' \$ \bar{v}' \# v_k$ also in R_k. By Lemma 3.2 the rule $u_k \# u' \$ \bar{v}' \# v_k$ is in R_k, and splicing $\bar{x}$ and $\bar{y}$ using this rule shows that $\nabla x_1 s x u_k v_k y t y_2 \nabla$ is in $\nabla \sigma^*(L) \nabla$. Applying Lemma 3.1 now shows that $\xi x u_k v_k y \eta = z$ is in $\kappa_c^*(\nabla \sigma^*(L) \nabla) = S''$, finishing the inclusion argument.

Finally, we notice that any cut operation introduces an end marker different from ∇, and so $(\nabla A^* \nabla) \cap \kappa_c^*(\nabla \sigma^*(L) \nabla) = \nabla \sigma^*(L) \nabla$. Hence $(\nabla A^* \nabla) \cap \kappa^*(\nabla L \nabla) \subset \nabla \sigma^*(L) \nabla$, and this implies $\nabla^{-1}\kappa^*(\nabla L \nabla)\nabla^{-1} \subset \sigma^*(L)$, finishing the proof of Proposition 3.3. $\square$

Now we want to translate a regular CPH scheme $\kappa = (A, E, C, P)$ into a splicing scheme. As an intermediate step (which will be useful later) we first translate it into a CPH scheme in which P is very simple.

We start the construction by enumerating E as $\{\varepsilon_1, \ldots, \varepsilon_n\}$. We then select two new sets of symbols, $\Delta = \{\delta_1, \ldots, \delta_n\}$ and $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$, and we set $\bar{E} = \Delta \cup \Gamma$. We let h be the homomorphism from $(A \cup \bar{E})^*$ to $(A \cup E)^*$ defined by $h(\delta_i) = h(\gamma_i) = \varepsilon_i$ and $h(a) = a$ for $a \in A$. Notice that h is a bijection when restricted to either $\Delta A^* \Gamma$ or $\Gamma A^* \Delta$.

We define $\bar{C} = h^{-1}(C) \cap (\Gamma A^* \Delta)$ and $\bar{P} = \{\eta\eta : \eta \in \bar{E}\}$. Our new CPH scheme is $\bar{\kappa} = (A, \bar{E}, \bar{C}, \bar{P})$. If $L \subset EA^*E$ is an initial language then we define $\bar{L} = h^{-1}(L) \cap (\Delta A^* \Gamma)$. Of course we must incorporate the pasting strings of the original scheme; we do this by essentially adding P to $\bar{L}$. However we must take some care, since there may be strings in P which are never actually used in generating $\kappa(L)$, and if we add such strings to $\bar{L}$ we may introduce spurious strings in the splicing language generated by $\bar{\kappa}$.

To be precise we introduce the following terminology. Suppose $\zeta \in \bar{E}$. Then we say that Z is a *left extension* of ζ iff $Z \in \Delta A^*$ and either $Z = \zeta$ (so $\zeta \in \Delta$) or $\zeta \in \Gamma$ and $h(Z\zeta) \in \kappa^*(L)$. Similarly, Z is a *right extension* of ζ iff $Z \in A^*\Gamma$ and either $Z = \zeta$ (so $\zeta \in \Gamma$) or $\zeta \in \Delta$ and $h(\zeta Z) \in \kappa^*(L)$. If $z = \zeta_1 w \zeta_2 \in \bar{E}A^*\bar{E}$ then we say $Z_1 w Z_2$ is an *extension* of z iff Z_1 is a left extension of ζ_1 and Z_2 is a right extension of ζ_2.

We define $\bar{P}_L$ as the set of all strings in $h^{-1}(P) \cap (\Gamma A^* \Delta)$ which have extensions.

Proposition 3.4. *Suppose $\kappa = (A, E, C, P)$ is a CPH scheme and $L \subset EA^*E$. If h, $\bar{\kappa} = (A, \bar{E}, \bar{C}, \bar{P})$, $\bar{L}$ and $\bar{P}_L$ are defined as above then $\kappa^*(L) = h(\bar{\kappa}^*(\bar{L} \cup \bar{P}_L) \cap (\Delta A^* \Gamma))$. If κ is finite or regular then so are $\bar{\kappa}$ and $\bar{P}_L$.*

Proof. Let Γ_0 be the set of markers in Γ which have left extensions, and let Δ_0 be the set of markers in Δ which have right extensions. These are finite and $\bar{P}_L = h^{-1}(P) \cap (\Gamma_0 A^* \Delta_0)$, so $\bar{P}_L$ is regular if P is regular. Since h is a bijection on $\Gamma A^* \Delta$ it is clear that $\bar{P}_L$ is finite if P is finite.

Now let S' be the set of strings $w \in \bar{E}A^*\bar{E}$ satisfying the following two conditions: w has an extension, and each extension W of w satisfies $h(W) \in \kappa^*(L)$. We claim that $\bar{\kappa}^*(\bar{L} \cup \bar{P}_L) \subset S'$. By Lemma 2.1 we only have to show that $S' \supset \bar{L} \cup \bar{P}_L$ and that $\bar{\kappa}(S') \subset S'$.

For the first part, note that any string in $\Delta A^* \Gamma$ has a unique extension, namely itself. From this it follows that $h(S' \cap \Delta A^* \Gamma) = \kappa^*(L)$, so $\bar{L} \subset S'$. Further, any string in $\bar{P}_L$ has an extension by definition. Moreover, suppose $W = Z_1 v Z_2$ is any extension of the element $w = \gamma_j v \delta_k$ of $\bar{P}_L$. Writing $Z_1 = \delta_i X_1$ and $Z_2 = X_2 \gamma_\ell$, we have both $h(Z_1 \gamma_j) = \varepsilon_i X_1 \varepsilon_j$ and $h(\delta_k Z_2) = \varepsilon_k X_2 \varepsilon_\ell$ in $\kappa^*(L)$. Also, $h(w) = \varepsilon_j v \varepsilon_k \in P$ so we can paste $h(Z_1\gamma_j)$ and $h(\delta_k Z_2)$ using $h(w)$ to show that $h(W) = \varepsilon_i X_1 v X_2 \varepsilon_\ell$ is in $\kappa^*(L)$. Hence $\bar{P}_L \subset S'$.

For the second part we have to consider both cutting and splicing of elements of S' using $\bar{\kappa}$. First suppose $z = \zeta_1 x u y \zeta_2$ is in S' and $\gamma_j u \delta_k$ is in $\bar{C}$, generating $x' = \zeta_1 x \gamma_j$ and $y' = \delta_k y \zeta_2$ in $\bar{\kappa}(S')$. We give the argument that x' is in S'; a symmetric argument works for y'. First, since z is in S' it has an extension, $Z_1 x u y Z_2$, so $Z_1 x \gamma_j$ is an extension of x'. Moreover, if $Z_1' x \gamma_j$ is any extension of x' then $Z_1' x u y Z_2$ is an extension of z, so $h(Z_1' x u y Z_2)$ is in $\kappa^*(L)$. But then $h(Z_1' x \gamma_j) = h(Z_1') x \varepsilon_j$ is obtained from $h(Z_1' x u y Z_2)$ by cutting at $\varepsilon_j u \varepsilon_k \in C$, so $h(Z_1' x \gamma_j)$ is in $\kappa^*(L)$.

To complete the proof of the claim we suppose $x' = \zeta_1 x \eta$ and $y' = \eta y \zeta_2$ are in S' and are pasted using $\eta\eta \in \bar{P}$ to produce $z = \zeta_1 x y \zeta_2 \in \bar{\kappa}(S')$. We shall show that z is in S' assuming $\eta = \gamma_k \in \Gamma$; the case that η is in Δ is handled symmetrically. First take extensions $Z_1 x \gamma_k$ and $Y y Z_2$ for x' and y'. Then $Z_1 x y Z_2$ is an extension of z. Moreover if $Z_1' x y Z_2'$ is any extension of z then $Z_1' x \gamma_k$ is an extension of x', so $h(Z_1' x \gamma_k)$ is in $\kappa^*(L)$. Hence $Z_1' x$ is a left extension of $\eta = \gamma_k$, so $(Z_1' x) y Z_2'$ is an extension of y'. Therefore $h(Z_1' x y Z_2')$ is in $\kappa^*(L)$.

Now we have established $\bar{\kappa}^*(\bar{L} \cup \bar{P}_L) \subset S'$. As noted above, $h(S' \cap (\Delta A^* \Gamma)) = \kappa^*(L)$, so we have $h(\bar{\kappa}^*(\bar{L} \cup \bar{P}_L) \cap (\Delta A^* \Gamma)) \subset \kappa^*(L)$.

Now we consider the opposite inclusion, $S'' = h(\bar{\kappa}^*(\bar{L} \cup \bar{P}_L) \cap \Delta A^* \Gamma) \supset \kappa^*(L)$. Since $L \subset S''$ we only have to show that $\kappa(S'') \subset S''$. Cutting operations in κ translate immediately to cutting operations in $\bar{\kappa}$, and it is then clear that the results of cutting strings of S'' at sites in C are in S''. Now suppose that $x' = \delta_i x \gamma_j$ and $y' = \delta_k x \gamma_\ell$ are in $\bar{\kappa}^*(\bar{L} \cup \bar{P}_L)$, and $h(x') = \varepsilon_i x \varepsilon_j$ and $h(y') = \varepsilon_k y \varepsilon_\ell$ in S'' are pasted using $p = \varepsilon_j v \varepsilon_k \in P$ to produce $z'' = \varepsilon_i x v y \varepsilon_\ell$ in $\kappa(S'')$. We have already shown that $S'' \subset \kappa^*(L)$, so $h(x')$ and $h(y')$ are in $\kappa^*(L)$. Hence $(\delta_i x) v (y \gamma_\ell)$ is an extension of $p' = \gamma_j v \delta_k$. Hence p' is in $\bar{P}_L$, and we can paste x', p' and y' using $\gamma_j \gamma_j$ and $\delta_k \delta_k$ to demonstrate that $z' = \delta_i x v y \gamma_\ell$ is in $\bar{\kappa}^*(\bar{L} \cup \bar{P}_L)$. Hence $z'' = h(z')$ is in S''.

We have now proved the two necessary inclusions, finishing the proof. □

Remark. In order to determine $\bar{P}_L$ it is necessary to determine $\kappa^*(L)$. In calculations this can be incorporated as part of an iterative procedure for finding $\bar{\kappa}^*(L)$. See the remark at the end of Section 7. Alternatively, one can translate to matrix terms and use Proposition 6.6.

We now complete the cycle, showing how to convert the result of Proposition 3.4 into a splicing scheme.

We start with a CPH scheme $\kappa = (A, E, C, P)$ with $P = \{\eta\eta : \eta \in E\}$ and we define an H scheme $\sigma = (A', R)$ as follows. First, the alphabet A' is $A \cup E \cup \{\nabla\}$ where ∇ is a new symbol, used to mark "catalyst" strings. Then we define R as the union of the following three sets of rules:

$$R_1 = \{a\#\zeta\$\zeta\#b : \zeta \in E;\ a,\ b \in A \cup E\},$$

$$R_2 = \{a\#ub\$\nabla\#\delta : \delta u\varepsilon \in C;\ a,\ b \in A \cup E\},$$

$$R_3 = \{\varepsilon\#\nabla\$au\#b : \delta u\varepsilon \in C;\ a,\ b \in A \cup E\}.$$

The symbols a and b in these rules are present only to avoid unwanted splicings with the catalyst strings and technical problems arising from empty cutting sites.

The following shows that σ determines the same splicing language as κ, modulo a finite number of catalyst strings.

Proposition 3.5. *Suppose $\kappa = (A, E, C, P)$ is a CPH scheme with $P = \{\eta\eta : \eta \in E\}$ and $L \subset EA^*E$. Let $\sigma = (A', R)$ be the H scheme described above and let F be the finite set $\nabla E \cup E\nabla$. Then $\kappa^*(L) = \sigma^*(L \cup F) \cap EA^*E = \sigma^*(L \cup F) \setminus F$. Also, if κ is finite or regular then so is σ.*

Proof. It is easy to check that σ is finite or regular if κ is.

We start by showing that $S' = \sigma^*(L \cup F) \cap EA^*E \supset \kappa^*(L)$. Since $S' \supset L$ we only have to show that $\kappa(S') \subset S'$. First suppose $x' = \xi x\zeta$ and $y' = \zeta y\eta$ are in S', so the result of a pasting operation gives $z = \xi xy\eta \in \kappa(S')$. But x' and y' may be spliced using the splicing rule $a\#\zeta\$\zeta\#b$ (where a is the last symbol in ξx and b is the first symbol in $y\eta$), and z is the result of this splicing, so $z \in S'$. Second, suppose $z = \xi xuy\eta$ is in S' and $\delta u\varepsilon$ is in C. Then $x' = \xi x\delta$ and $y' = \varepsilon y\eta$ are the results of cutting z at the site $\delta u\varepsilon$, so they are in $\kappa(S')$. Consider the splicing rule $a\#ub\$\nabla\#\varepsilon$; this applies to the pair z and $\nabla\varepsilon$ (where a is the last symbol in ξx and b is the first symbol in $y\eta$) and the result of splicing x and $\nabla\varepsilon$ using this rule is x'. Hence x' is in S'. A symmetric argument shows that y' is also in S'.

Next we show that $S'' = \kappa^*(L) \cup F \supset \sigma^*(L \cup F)$. Again $S'' \supset L \cup F$ so we only have to show that $\sigma(S'') \subset S''$. Suppose x' and y' in S'' may be spliced to produce $z \in \sigma(S'')$. We have to consider the different possible rules that can apply.

First, suppose the rule used is $a\#\zeta\$\zeta\#b \in R_1$. Then the left and right sites specify two symbols, neither of which is ∇, so neither x' nor y' can be in F. Hence, $x' \in \kappa^*(L) \subset EA^*E$ contains the substring $a\zeta$ so $x' = x''a\zeta$. Similarly, $y' = \zeta by''$. Moreover, these are the only locations of the sites $a\zeta$ in x' and ζb in y'. Hence, the result z of splicing x' and y' is $x''aby''$, which is the result of pasting x' and y' using $\zeta\zeta$, so z is in $\kappa^*(L) \subset S''$.

Next, suppose the rule used is $a\#ub\$\nabla\#\delta \in R_2$. Then the only string in S'' that contains the site $\nabla\delta$ is $\nabla\delta$ itself, so $y' = \nabla\delta$. Since x' contains the site aub we have $x' = \alpha x_1 u x_2 \beta$ where a is the last symbol in αx_1 and b is the first symbol in $x_2\beta$. Thus, the result z of splicing x' and y' is $\alpha x_1 \delta$. But $\delta u \varepsilon$ is in C, and z is also one of the results of cutting x' at the site $\delta u \varepsilon$, so z is in $\kappa^*(L) \subset S''$.

The case of a rule in R_3 is symmetric to the case of a rule in R_2.

We have now shown $\kappa^*(L) \cup F \supset \sigma^*(L \cup F)$ and $\sigma^*(L \cup F) \cap EA^*E \supset \kappa^*(L)$. We also have $F \subset \sigma^*(L \cup F)$, $\kappa^*(L) \subset EA^*E$ and $F \cap (EA^*E) = \emptyset$. From these and some elementary set manipulations we derive $\kappa^*(L) = \sigma^*(L \cup F) \cap EA^*E = \sigma^*(L \cup F) \setminus F$. □

Remark. We briefly show how the "triplet" formulation of [16, 15] fits in the current analysis. In this formulation a splicing rule is a string of the form $u \diamond w \diamond v$ where $\diamond$ is a symbol not in the alphabet A, and the corresponding splicing operation is $(xux', y'vy) \Longrightarrow xwy$. A *triplet H scheme* is a pair $\tau = (A, T)$ where T is a set of such rules. Any regular H scheme σ with finitely many visible sites can be translated into a regular triplet scheme; the translation maps $u\#u'\$v'\#v$ to $uu' \diamond uv \diamond v'v$. Finiteness of the possible values for uv is needed to preserve regularity, and $\sigma^*(L) = \tau^*(L)$ is obvious. In the other direction we indicate how to translate a regular triplet scheme $\tau = (A, T)$ into a regular CPH scheme. We choose an automaton representation $T = L(G, I, F)$; by eliminating unnecessary edges in G we may assume that the set of edges labeled by $\diamond$ is the disjoint union of two sets E_1 and E_2, so that any accepting path in G has edges e in E_1 and f in E_2, with e before f. We choose end markers δ_e for $e \in E_1$ and ε_f for $f \in E_2$, together with two more markers ∇ and $\bar{\nabla}$. For a rule $u \diamond w \diamond v$ represented by a path in G passing through e and f we add $\delta_e u \bar{\nabla}$ and $\bar{\nabla} v \varepsilon_f$ to C and $\delta_e w \varepsilon_f$ to P. This defines the CPH scheme κ, and $\tau^*(L) = \nabla^{-1}\kappa^*(\nabla L \nabla)\nabla^{-1}$ follows by an argument similar to the proof of Proposition 3.3.

4. Matrices of languages

By an *$m \times n$ matrix of languages X over A* we mean a rectangular array $[X_{ik}]$ of languages in A^*, for $1 \leqslant i \leqslant m$, $1 \leqslant k \leqslant n$. We say such a matrix X is finite or regular iff each X_{ik} is finite or regular. If $\mathscr{F}$ is an abstract family of languages then we say that X is in $\mathscr{F}$ iff each X_{ik} is in $\mathscr{F}$.

Square matrices of languages are just end marked languages in disguise: Suppose E is a set of end markers, enumerated as $\{\varepsilon_1, \dots, \varepsilon_n\}$. If $X \subset EA^*E$ then we define the *matrix form of X* to be the matrix $X^\dagger$ defined by $X^\dagger_{jk} = \varepsilon_j^{-1} X \varepsilon_k^{-1} \subset A^*$. Conversely, if X is a $n \times n$ matrix of languages over A then we define the *end marked form of X* to be the language $X^\dagger = \bigcup_{jk} \varepsilon_j X_{jk} \varepsilon_k$. Clearly, for any full AFL $\mathscr{F}$, X is in $\mathscr{F}$ if and only if $X^\dagger$ is in $\mathscr{F}$.

The advantage of using matrices rather than end marked languages is that various operations on languages are simpler to describe and manipulate using matrix terminology. In the following we assume that all matrices are square, of size $n \times n$.

If X and Y are matrices of languages we define $X \leqslant Y$ to mean $X_{ik} \subset Y_{ik}$ for all i, k. Similarly, we define the union and intersection of a family of matrices componentwise. To emphasize the algebraic nature of some of our calculations we usually write $X \cup Y$ as $X + Y$ and $\bigcup_{p=0}^{\infty} X_p$ as $\sum_{p=0}^{\infty} X_p$. We define concatenation and right and left quotients by

- $(XY)_{ik} = \bigcup_j X_{ij} Y_{jk}$,
- $(X^{-1}Z)_{jk} = \bigcup_i (X_{ij})^{-1} Z_{ik}$,
- $(ZY^{-1})_{ij} = \bigcup_k Z_{ik}(Y_{jk})^{-1}$.

These definitions may be rewritten as follows:

- If $Z = XY$ then $z \in Z_{ik}$ iff $z = xy$ for some j, $x \in X_{ij}$, $y \in Y_{jk}$.
- If $Y = X^{-1}Z$ then $y \in Y_{jk}$ iff $z = xy$ for some i, $x \in X_{ij}$, $z \in Z_{ik}$.
- If $X = ZY^{-1}$ then $x \in X_{ij}$ iff $z = xy$ for some k, $y \in Y_{jk}$, $z \in Z_{ik}$.

The zero matrix O is defined by $O_{ik} = \emptyset$, and the identity matrix I is defined by $I_{ii} = \{1\}$ and $I_{ik} = \emptyset$ for $i \neq k$. The powers X^p are defined by iterated concatenation, with $X^0 = I$, and then X^* is defined as $\sum_{p \geqslant 0} X^p$. To save a level of parentheses later we introduce the following notation:

$$X^{-p}Y = (X^p)^{-1}Y, \quad X^{-*}Y = (X^*)^{-1}Y,$$
$$YX^{-p} = Y(X^p)^{-1}, \quad YX^{-*} = Y(X^*)^{-1}.$$

We shall need to perform various computations with these operations. We summarize next some basic rules:

Lemma 4.1. *For $n \times n$ matrices of languages:*
1. $X(YZ) = (XY)Z$, $X(\sum_p Y_p) = \sum_p XY_p$.
2. $X^{-1}(\sum_p Y_p) = \sum_p X^{-1}Y_p$, $(\sum_p X_p)^{-1}Y = \sum_p X_p^{-1}Y$.
3. *If* $Y \leqslant Z$ *then* $XY \leqslant XZ$, $X^{-1}Y \leqslant X^{-1}Z$, *and* $Y^{-1}X \leqslant Z^{-1}X$.
4. $X^{-1}(Y^{-1}Z) = (YX)^{-1}Z$, $(ZX^{-1})Y^{-1} = Z(YX)^{-1}$, $(X^{-1}Z)Y^{-1} = X^{-1}(ZY^{-1})$.
5. $X^{-1}(YZ) = (X^{-1}Y)Z + (Y^{-1}X)^{-1}Z$.

The sums involved can be either finite or infinite.

Proof. All of these are analogues of the corresponding rules for operations on languages, and the proofs are very similar. As an example we give the argument for part 5.

First, $w \in (X^{-1}(YZ))_{jk}$ iff $v = xw$ for some i, $x \in X_{ij}$, $v \in (YZ)_{ik}$, and $v \in (YZ)_{ik}$ iff $v = yz$ for some l, $y \in Y_{il}$, $z \in Z_{lk}$. Since $xw = yz$ we have either $x = yz_1$, $z_1 w = z$ or $y = xw_1$, $w_1 z = w$. In the first case we have $z_1 \in (Y_{il})^{-1} X_{ij} \subset (Y^{-1}X)_{lj}$, so $w \in ((Y^{-1}X)_{lj})^{-1} Z_{lk} \subset ((Y^{-1}X)^{-1}Z)_{jk}$. In the second case, $w_1 \in (X_{ij})^{-1} Y_{il} \subset (X^{-1}Y)_{jl}$, so $w \in (X^{-1}Y)_{jl} Z_{lk} \subset ((X^{-1}Y)Z)_{jk}$.

For the opposite inclusion, suppose first that $w \in ((Y^{-1}X)^{-1}Z)_{jk}$. Then $z = z_1 w$ for some l, $z_1 \in (Y^{-1}X)_{lj}$, $z \in Z_{lk}$. Hence $x = yz_1$ for some i, $y \in Y_{il}$, $x \in X_{ij}$. Therefore

$xw = yz_1w = yz$ and $yz \in Y_{il}Z_{lk} \subset (YZ)_{ik}$ so $w \in (X_{ij})^{-1}(YZ)_{ik} \subset (X^{-1}(YZ))_{jk}$. In the second case, $w \in ((X^{-1}Y)Z)_{jk}$ so $w = w_1z$ for some l, $w_1 \in (X^{-1}Y)_{jl}$, $z \in Z_{lk}$. Hence $y = xw_1$ for some i, $y \in Y_{il}$, $x \in X_{ij}$. Therefore, $xw = xw_1z = yz$ and again $yz \in Y_{il}Z_{lk} \subset (YZ)_{ik}$ so $w \in (X_{ij})^{-1}(YZ)_{ik} \subset (X^{-1}(YZ))_{jk}$. $\square$

We shall use Lemma 4.1 many times, usually without attribution.

Remark. In Lemma 4.1, as well as in similar situations in the rest of the paper, we generally do not mention explicitly that there are "right-handed" versions of the results, obtained by interchanging the operations of left concatenation and left quotient with right concatenation and right quotient.

Note that $X^{-1}Z$ has nothing to do with matrix inversion. For computations it is simplest to consider $X^{-1}Y$ as the result of multiplying Y by the matrix X^{-1} defined by $(X^{-1})_{jk} = (X_{kj})^{-1}$. Of course, this does not make sense, since the inverse of a language is not a language. It does, however, make sense if one regards the inverse of a language L as the *operator* $M \mapsto L^{-1}M$, and then X^{-1} may be understood as a matrix of such operators. There is one special case in which this is particularly easy to interpret. Namely, if the language L is either $\emptyset$ or 1 then the operation $M \mapsto L^{-1}M$ coincides with the operation $M \mapsto LM$. To formulate the matrix analogue of this observation we say a matrix P is a 0–1 matrix iff each entry is either $\emptyset$ or 1.

Lemma 4.2. *Suppose P is a* 0–1 *matrix. Then, for any B and Z*:
1. $P^{-1}Z = P^{\mathrm{T}}Z$, *where P^{T} is the transpose of P.*
2. $B^{-1}(ZP) = (B^{-1}Z)P$.

Proof. Part 1 follows from the preceding discussion. For part 2 set $Q = P^{\mathrm{T}}$ and apply the right-hand version of part 1 twice:

$$B^{-1}(ZP) = B^{-1}(ZQ^{-1}) = (B^{-1}Z)Q^{-1} = (B^{-1}Z)P. \qquad \square$$

We also need the matrix analogues of the closure properties of a full AFL:

Lemma 4.3. *Suppose $\mathscr{F}$ is a full AFL.*
1. *If X and Y are matrices in $\mathscr{F}$ then $X + Y$, XY, and X^* are in $\mathscr{F}$.*
2. *If X is in $\mathscr{F}$ and Z is regular then $X \cap Z$ is in $\mathscr{F}$.*
3. *If Z is in $\mathscr{F}$ and X and Y are regular then $X^{-1}Z$ and ZY^{-1} are in $\mathscr{F}$, and if Z is regular and X and Y are arbitrary then $X^{-1}Z$ and ZY^{-1} are regular.*

Proof. These are immediate from the corresponding results for languages except, perhaps, for X^*. Take a set $E = \{\varepsilon_1, \ldots, \varepsilon_n\}$ of end markers so we can consider $X^\dagger$. We let $D = \{\varepsilon_i\varepsilon_i 1 \leqslant i \leqslant n\}$. Then $S_{ik} = (X^\dagger)^* \cap (\varepsilon_i(A^*D)^*A^*\varepsilon_k)$ is in $\mathscr{F}$. If h is the homomorphism from $(A \cup E)^*$ to A^* defined by erasing the markers in E then $h(S_{ik}) = (X^*)_{ik}$, so $(X^*)_{ik}$ is in $\mathscr{F}$. $\square$

5. Splicing via matrices

We now reinterpret splicing in matrix terms. An *MH scheme* (for "matrix H scheme") is a quadruple $\mu = (A, B, D, P)$ where A is an alphabet and B, D and P are $n \times n$ matrices of languages over A. If necessary we shall refer to μ as an $n \times n$ MH scheme, but usually n will be clear from the context. We say that μ is regular iff B, D and P are all regular matrices. If X is a matrix of languages then we define

$$\mu(X) = X + B^{-1}X + XD^{-1} + XPX.$$

We call B and D the *left* and *right cutting matrices*, and P the *pasting matrix*.

It follows from Lemma 4.1 that $X \mapsto B^{-1}X + XD^{-1} + XPX$ is order-preserving and continuous, so Lemma 2.1 applies to μ. Thus, the splicing matrix $\mu^*(L)$ is the smallest matrix of languages S satisfying the equation

$$S = L + B^{-1}S + SD^{-1} + SPS. \tag{5.1}$$

We shall refer to this equation as the *splicing equation* for μ and L.

In the following we use the *universal matrix U* defined by $U_{ik} = A^*$ for all i, k.

Proposition 5.1. *Suppose $\kappa = (A, E, C, P)$ is a CPH scheme. Define $B = UC^\dagger$, $D = C^\dagger U$ and $\mu = (A, B, D, P^\dagger)$. Then μ is regular if κ is regular, and, if L is any end marked language, $(\kappa^*(L))^\dagger = \mu^*(L^\dagger)$.*

Proof. Suppose X is any matrix of languages. Notice that $y \in (B^{-1}X)_{jk}$ iff $x = by$ for some i, $b \in B_{ij}$, $x \in X_{ik}$. Also, $b \in B_{ij}$ iff $b = wu$ for some l, $w \in U_{il}$, $u \in C^\dagger_{lj}$. That is, $w \in A^*$, $\varepsilon_i wuy\varepsilon_k = \varepsilon_i x \varepsilon_k$ is in $X^\dagger$, and $\varepsilon_l u \varepsilon_j \in C^\dagger$. Hence $y \in (B^{-1}X)_{jk}$ iff $\varepsilon_j y \varepsilon_k$ is the "left" fragment resulting from a cut operation on some $\varepsilon_i x \varepsilon_k \in X^\dagger$ at some cutting site $\varepsilon_l u \varepsilon_j \in C^\dagger$. Together with a similar analysis for XD^{-1}, we see that $B^{-1}X + XD^{-1}$ is the matrix form of the set of strings resulting from single cut operations on $X^\dagger$ using the cutting sites in $C^\dagger$. An even simpler calculation shows that XPX is the matrix form of the set of strings resulting from single pasting operations on pairs of strings in $X^\dagger$ using the pasting strings in $P^\dagger$. Hence $\mu(X) = (\kappa(X^\dagger))^\dagger$, from which the proposition easily follows. □

Next we translate an $n \times n$ MH scheme $\mu = (A, B, D, P)$ to a CPH scheme. We start with a set of end markers $E = \{\varepsilon_1, \ldots, \varepsilon_n\}$ which will define the end marked forms of various matrices. We add two more symbols $\{\nabla, \bar{\nabla}\}$ to E to form E', and we add $2n$ more symbols p_i, q_i, $1 \leqslant i \leqslant n$ to A to form A'.

Notice that, as a cutting operation, $X \mapsto B^{-1}X$ acts by matching and deleting an entire prefix of an element of X_{ik}, and different components of B apply to different components of X. However, a cutting operation in a CPH scheme cannot take into account the end markers. The new *alphabet* symbols p_i and q_j are used as alternative markers for the ends of strings. Since these are alphabet symbols the cutting sites can refer to them.

The set of pasting strings P' consists of $P^\dagger$ plus the finite set of strings of the form $\bar{\nabla} p_j \varepsilon_j$ or $\varepsilon_j q_j \bar{\nabla}$, which are used to replace end markers with the alphabetic markers p_j and q_j. Finally, we place in C' all strings of the form $\nabla p_i b \varepsilon_k$ where $b \in B_{ik}$ or $\varepsilon_i d q_k \nabla$ where $d \in D_{ik}$.

Proposition 5.2. *Suppose $\mu = (A,B,D,P)$ is an MH scheme and $\kappa = (A',E',C',P')$ is defined as above. Then κ is a regular CPH scheme if μ is regular. If L is any matrix of languages and $L' = L^\dagger \cup \{\nabla\bar{\nabla},\ \bar{\nabla}\nabla,\ \nabla\nabla\}$ then $\mu^*(L) = (\kappa^*(L') \cap EA^*E)^\dagger$.*

Proof. Regularity of κ is clear.

Let $S' = (\kappa^*(L') \cap EA^*E)^\dagger$. As usual, in order to establish $\mu^*(L) \leqslant S'$ we only have to show that $\mu(S') \leqslant S'$. Take $z \in (\mu(S'))_{jk} = (S' + B^{-1}S' + S'D^{-1} + S'PS')_{jk}$; we need to consider three possibilities.

First consider $z \in (B^{-1}S')_{jk}$. That is, for some i, $w = bz$ with $w \in S'_{ik}$ and $b \in B_{ij}$. So $w' = \varepsilon_i w \varepsilon_k$ is in $\kappa^*(L')$ and can be pasted with $\nabla\bar{\nabla} \in L'$ using $\bar{\nabla} p_i \varepsilon_i \in P'$ to produce $w'' = \nabla p_i w \varepsilon_k = \nabla p_i b z \varepsilon_k$ in $\kappa^*(L')$. Now we can cut w'' at the cutting site $\nabla p_i b \varepsilon_j \in C'$ to produce $\nabla\nabla$ and $z' = \varepsilon_j z \varepsilon_k$ in $\kappa^*(L')$. Hence $z \in S'_{jk}$.

The case $z \in (S'D^{-1})_{jk}$ is handled similarly, and the last case, $z \in (S'PS')_{jk}$, is a simple matter of translating between matrix and end marked forms. So we have established the first inclusion, $\mu^*(L) \leqslant S'$.

For the reverse inclusion, we define the language S'' as the set of all strings of the form $s_j x t_k$ where $x \in \mu^*(L)_{jk}$, $s_j = \varepsilon_j$ or ∇p_j, and $t_k = \varepsilon_k$ or $q_k \nabla$, together with the three strings $\nabla\bar{\nabla}$, $\bar{\nabla}\nabla$ and $\nabla\nabla$. We shall show $\kappa^*(L') \subset S''$. Since $L' \subset S''$ we only need to show $\kappa(S'') \subset S''$.

First consider the effect of cutting a string $z' \in S''$ at a site $\nabla p_i b \varepsilon_j \in C'$, with $b \in B_{ij}$. Then z' must contain the symbol p_i so the only possible form for z' is $z' = \nabla p_i b y t_k$ and the results of cutting are $\nabla\nabla \in S''$ and $y' = \varepsilon_j y t_k$. Then $by \in \mu^*(L)_{ik}$ and $b \in B_{ij}$ so $y \in (B^{-1}\mu^*(L))_{jk} \subset \mu^*(L)_{jk}$. Hence $y' = \varepsilon_j y t_k$ is in S''. Together with the symmetric argument for cutting at sites $\varepsilon_i d q_k \nabla$ this shows that S'' is closed under cutting.

Now consider the possibilities for pasting two strings x' and y' in S'' using a pasting string $p' \in P'$. If $p' = \bar{\nabla} p_j \varepsilon_j$ then $x' = \nabla\bar{\nabla}$, since this is the only string in S'' ending with $\bar{\nabla}$, and $y' = \varepsilon_j y t_k$ with $y \in \mu^*(L)_{jk}$. Then the result of pasting is $\nabla p_j y t_k$, which is still in S''. A similar argument holds if $p' = \varepsilon_j q_j \bar{\nabla}$. Finally consider $p' = \varepsilon_j v \varepsilon_k \in P^\dagger$. Then $x' = s_i x \varepsilon_j$ and $y' = \varepsilon_k y t_\ell$ with $x \in \mu^*(L)_{ij}$ and $y \in \mu^*(L)_{k\ell}$. But then the result of pasting x' and y', $s_i x v y t_\ell$, is in S'' since $xvy \in (\mu^*(L)P\mu^*(L))_{i\ell} \subset \mu^*(L)_{i\ell}$. Therefore S'' is also closed under pasting, so $\kappa(S'') \subset S''$.

Now we finish the proof, since we have already shown $\mu^*(L) \leqslant S'$, and clearly $S'' \cap EA^*E = (\mu^*(L))^\dagger$, so $\kappa^*(L') \subset S''$ implies $\kappa^*(L') \cap EA^*E \subset (\mu^*(L))^\dagger$, and hence $S' \leqslant \mu^*(L)$. $\square$

The translation from a CPH scheme to an MH scheme introduces the matrices $B = UC^\dagger$ and $D = C^\dagger U$ which are (usually) infinite even if C is finite. Thus a translation to matrix terms and back to splicing will lose information about finiteness of the original scheme. We now give an alternate construction which recaptures a finite CPH scheme in this case.

We assume that $\mu = (A,B,D,P)$ is an MH scheme and that there are matrices $\tilde{B}$ and $\tilde{D}$ such that $B = U\tilde{B}$ and $D = \tilde{D}U$. We construct a CPH scheme as follows. First we choose a set $E = \{\varepsilon_1,\dots,\varepsilon_n\}$ of end markers to set up the correspondence between matrices and end marked languages in EA^*E. We add to E one more symbol, ∇, to form $\tilde{E}$. We now form $\tilde{C}$ as the set of all strings $\nabla b\varepsilon_k$ where $b \in \tilde{B}_{jk}$ and $\varepsilon_j d\nabla$ where $b \in \tilde{D}_{jk}$.

Proposition 5.3. *Suppose $\mu = (A,B,D,P)$ is an MH scheme, and suppose there are matrices $\tilde{B}$ and $\tilde{D}$ such that $B = U\tilde{B}$ and $D = \tilde{D}U$. Define $\tilde{E}$ and $\tilde{C}$ as above, and let $\kappa = (A,\tilde{E},\tilde{C},P^\dagger)$. Then, for any matrix of languages L, $\mu^*(L) = (\kappa^*(L^\dagger) \cap EA^*E)^\dagger$. Moreover, κ is a regular CPH scheme if μ is regular, and κ is a finite CPH scheme if P, $\tilde{B}$, and $\tilde{D}$ are finite.*

Proof. The idea here is just that the unwanted result of each cut operation will be marked with ∇ and strings containing ∇ will not be accepted in the final language. The details are similar to (and simpler than) those in several previous proofs and are left to the reader. □

6. Solving the splicing equation

We shall now show how to solve the splicing equation (5.1). We first consider a trivial special case, with no cutting:

Proposition 6.1. *If $\tilde{\mu} = (A,O,O,P)$ then, for any L, $\tilde{\mu}^*(L) = (LP)^*L = L(PL)^*$.*

Since $\tilde{\mu}(Z) = Z + ZPZ$ this is easily proved by induction. Note that this just says that the strings in the splicing language are obtained by pasting together a sequence of strings from the initial language.

We shall need the following calculation when solving more general forms of the splicing equation.

Lemma 6.2. *Let $\tilde{\mu} = (A,O,O,P)$, with P a 0-1 matrix, and suppose Z and B are matrices, with $I \leqslant B$. Define $\tilde{S} = \tilde{\mu}(Z)$ and $\tilde{B}_0 = B + (\tilde{S}P)^{-1}B$. Then*

$$\tilde{B}_0^{-1}\tilde{\mu}^*(Z) \leqslant \tilde{\mu}^*(\tilde{B}_0^{-1}Z).$$

Proof. We start with

$$\tilde{B}_0^{-1}\tilde{\mu}(Z) = \tilde{B}_0^{-1}Z + \tilde{B}_0^{-1}(ZPZ) = \tilde{B}_0^{-1}Z + \tilde{B}_0^{-1}(ZP)Z + ((ZP)^{-1}\tilde{B}_0)^{-1}Z, \tag{6.1}$$

using Lemma 4.1.5. The last term becomes

$$\begin{aligned}((ZP)^{-1}\tilde{B}_0) &= (ZP)^{-1}B + (ZP)^{-1}((\tilde{S}P)^{-1}B) = (ZP)^{-1}B + (\tilde{S}PZP)^{-1}B\\ &\leqslant (\tilde{S}P)^{-1}B \leqslant \tilde{B}_0,\end{aligned}$$

using $Z \leqslant \tilde{S}$ and $\tilde{S}P\tilde{S}P = (\tilde{S}P\tilde{S})P \leqslant \tilde{S}P$. Thus (6.1) simplifies to

$$\tilde{B}_0^{-1}\tilde{\mu}(Z) \leqslant \tilde{B}_0^{-1}Z + \tilde{B}_0^{-1}(ZP)Z + \tilde{B}_0^{-1}Z = \tilde{B}_0^{-1}Z + \tilde{B}_0^{-1}(ZP)Z. \tag{6.2}$$

Now we apply Lemma 4.2.2 to $\tilde{B}_0^{-1}(ZP)$ to continue the calculation:

$$\tilde{B}_0^{-1}\tilde{\mu}(Z) \leqslant \tilde{B}_0^{-1}Z + (\tilde{B}_0^{-1}Z)PZ \leqslant \tilde{B}_0^{-1}Z + (\tilde{B}_0^{-1}Z)P(\tilde{B}_0^{-1}Z) = \tilde{\mu}(\tilde{B}_0^{-1}Z),$$

where $Z \leqslant \tilde{B}_0^{-1}Z$ follows from $I \leqslant \tilde{B}_0$. Writing $Z_p = \tilde{\mu}^p(Z)$, we still have $\tilde{S} = \tilde{\mu}^*(Z_p)$ so we can apply the preceding with Z replaced with Z_p to get

$$\tilde{B}_0^{-1}Z_{p+1} = \tilde{B}_0^{-1}(\tilde{\mu}^{p+1}(Z)) = \tilde{B}_0^{-1}(\tilde{\mu}(Z_p)) \leqslant \tilde{\mu}(\tilde{B}_0^{-1}Z_p),$$

and from this $\tilde{B}_0^{-1}\tilde{\mu}^*(Z) \leqslant \tilde{\mu}^*(\tilde{B}_0^{-1}Z)$ follows by induction. □

The following is a more general solution of the splicing equation. We allow cutting, but we require that all pasting strings be empty words:

Theorem 6.3. *Suppose* $\mu = (A,B,D,P)$ *with* P *a* 0–1 *matrix and let* $S = \mu^*(L)$. *Define* $B_0 = (I + SP)^{-1}(I + B)$, $\bar{B} = B_0^*$, $D_0 = (I + D)(I + PS)^{-1}$, $\bar{D} = D_0^*$ *and* $\bar{L} = \bar{B}^{-1}L\bar{D}^{-1}$. *Then* $S = (\bar{L}P)^*\bar{L}$.

Remark. The idea is that, as in the simplest case of Proposition 6.1, a string in the splicing language is obtained by pasting together a sequence of strings from a modified initial language. The modified initial language consists of the results of cutting operations on the initial language, where the cutting matrices B and D have been replaced by "enhanced" cutting matrices $\bar{B}$ and $\bar{D}$.

Proof. If we replace B by $I + B$ and D by $I + D$ we do not change $\mu(Z) = Z + B^{-1}Z + ZD^{-1} + ZPZ$, so we shall assume $I \leqslant B$ and $I \leqslant D$. Thus $B_0 = B + (SP)^{-1}B$ and $D_0 = D + D(PS)^{-1}$. We define $\tilde{\mu} = (A,O,O,P)$ and $\tilde{S} = \tilde{\mu}^*(\bar{L}) = (\bar{L}P)^*\bar{L}$, by Proposition 6.1. We must show $S = \tilde{S}$.

To show $\tilde{S} \leqslant S$ we need $\bar{L} \leqslant S$ and $\tilde{\mu}(S) \leqslant S$. The second inequality is obvious, since $\tilde{\mu}(Z) \leqslant \mu(Z)$ for all Z. For the first we need $\bar{B}^{-1}S \leqslant S$. To prove this, start with $((SP)^{-1}B)^{-1}S \leqslant B^{-1}(SPS) \leqslant S$, using Lemma 4.1.5 and $B^{-1}S + SPS \leqslant \mu(S) = S$. Hence $B_0^{-1}S = B^{-1}S + ((SP)^{-1}B)^{-1}S \leqslant S$, again using $B^{-1}S \leqslant S$. Now $\bar{B}^{-1}S \leqslant S$ follows by iteration. Similarly, $S\bar{D}^{-1} \leqslant S$. Therefore $\bar{L} = \bar{B}^{-1}L\bar{D}^{-1} \leqslant \bar{B}^{-1}S\bar{D}^{-1} \leqslant S$.

To show $S \leqslant \tilde{S}$ we need $L \leqslant \tilde{S}$ and $\mu(\tilde{S}) \leqslant \tilde{S}$. Since $I \leqslant \bar{B}$ and $I \leqslant \bar{D}$ we have $L \leqslant \bar{L} \leqslant \tilde{S}$. Consider $\mu(\tilde{S}) = \tilde{S} + B^{-1}\tilde{S} + \tilde{S}D^{-1} + \tilde{S}P\tilde{S}$. We have $\tilde{S}P\tilde{S} \leqslant \tilde{\mu}(\tilde{S}) = \tilde{S}$, so we only need $B^{-1}\tilde{S} \leqslant \tilde{S}$ and $\tilde{S}D^{-1} \leqslant \tilde{S}$. We give the details for B; the argument for D is similar.

Let $\tilde{B}_0 = (\tilde{S}P)^{-1}B$ as in Lemma 6.2. We have already shown that $\tilde{S} \leqslant S$, so $\tilde{B}_0 \leqslant B_0$. Hence, by Lemma 6.2, $B^{-1}\tilde{S} = B^{-1}\tilde{\mu}^*(\bar{L}) \leqslant \bar{\tilde{B}}_0^{-1}\tilde{\mu}^*(\bar{L}) \leqslant \tilde{\mu}^*(\bar{\tilde{B}}_0^{-1}\bar{L}) \leqslant \tilde{\mu}^*(\bar{B}_0^{-1}\bar{L})$. But now $\bar{B}_0^{-1}\bar{L} \leqslant \bar{B}^{-1}\bar{L} = \bar{B}^{-2}L\bar{D}^{-1} = \bar{B}^{-1}L\bar{D}^{-1} = \bar{L}$, since $\bar{B}^2 = \bar{B}$. Thus, we have shown $B^{-1}\tilde{S} \leqslant \tilde{\mu}^*(B_0^{-1}\bar{L}) \leqslant \tilde{\mu}^*(\bar{L}) = \tilde{S}$, as required. □

From this solution in the 0–1 case we can now deduce our main theorem, the Closure Theorem from Section 1. We restate it more precisely here:

Theorem 6.4 (Closure Theorem). *Suppose $\mathscr{F}$ is a full AFL.*
1. *If $\sigma = (A, R)$ is a regular splicing scheme with finitely many visible sites and $L \subset A^*$ is in $\mathscr{F}$ then $\sigma^*(L)$ is in $\mathscr{F}$.*
2. *If $\kappa = (A, E, C, P)$ is a regular CPH scheme and $L \subset EA^*E$ is in $\mathscr{F}$ then $\kappa^*(L)$ is in $\mathscr{F}$.*
3. *If $\mu = (A, B, D, P)$ is a regular $n \times n$ MH scheme and L is an $n \times n$ matrix of languages in $\mathscr{F}$ then $\mu^*(L)$ is in $\mathscr{F}$.*

Proof. Propositions 3.3, 3.4, 3.5, 5.1 and 5.2 all transform one type of splicing scheme and initial object (language or matrix) into another. They all preserve regularity of the scheme. They all operate by transforming the initial object, operating on the transformed object by the new scheme, and then transforming the new splicing object back to the original splicing object. All these transformations require only simple language or matrix operations which preserve any given full AFL $\mathscr{F}$.

Therefore, as far as the validity of the Closure Theorem goes we can apply any of these transformations, and just prove closure for the transformed system. We choose to transform the given scheme and initial object into the special type of CPH scheme described in Proposition 3.4, and then apply Proposition 5.1 to produce a regular MH scheme $\mu' = (A', B', D', P')$ and a matrix L' in $\mathscr{F}$. In the special CPH scheme produced by Proposition 3.4 the set of pasting strings is $\{\eta\eta : \eta \in E\}$, and so the resulting matrix P' is the identity matrix I.

Thus, using the notation of Theorem 6.3 with $\mu = (A, B, D, P)$ replaced with $\mu' = (A', B', D', I)$, we can write $S' = \mu'^*(L') = \bar{L}^+$.

To finish we use the closure properties of Lemma 4.3. No matter what S' is, B_0 and D_0 are quotients of the regular matrices $I + B'$, respectively $I + D'$, so they are regular. Hence $\bar{B}$ and $\bar{D}$ are regular, so $\bar{L}$ is in $\mathscr{F}$ since $\mathscr{F}$ is closed under quotient by regular matrices. Since $\mathscr{F}$ is closed under concatenation and Kleene closure we conclude that $S' = \bar{L}\bar{L}^*$ is in $\mathscr{F}$. □

An examination of the proof shows that we can make two improvements. First, the pasting language P is added to the splicing language in Proposition 3.4, so we only need to require that P be in $\mathscr{F}$, not that it be regular. Second, since the quotient of a regular language by anything is still regular, we do not need to require that B and D (or C) be regular if $\mathscr{F}$ is the regular family. Thus we have

Addendum 6.5. *Theorem* 6.4 *remains true if we replace the assumption that P is regular with the assumption that P is in $\mathscr{F}$. It also remains true if $\mathscr{F}$ is the regular family and we remove the assumptions of regularity on R, C, B and D.*

The transformations in the proof of the Closure Theorem lead to a simple form for the splicing matrix but they generally increase the size of the system considerably. For example, suppose we start with an $n \times n$ MH scheme. Applying Proposition 5.2 produces a CPH system with $n+2$ end markers and $2n$ additional alphabet symbols. Then Proposition 3.4 transforms this into a CPH scheme with $2n+4$ end markers. Thus, the MH system used in the proof of the Closure Theorem uses $(2n+4) \times (2n+4)$ matrices over an alphabet with $2n$ additional symbols. The following is a more direct (and simpler) translation from an MH system to an MH system with a 0–1 pasting matrix:

Proposition 6.6. *Suppose $\mu = (A,B,D,P)$ is an $n \times n$ MH scheme and L is an $n \times n$ matrix. Using $n \times n$ blocks, define the $(2n) \times (2n)$ matrices*

$$\hat{B} = \begin{bmatrix} B & O \\ O & O \end{bmatrix}, \quad \hat{D} = \begin{bmatrix} O & O \\ O & D \end{bmatrix}, \quad \hat{L} = \begin{bmatrix} O & L \\ P & O \end{bmatrix}.$$

Further define $\hat{\mu} = (A,\hat{B},\hat{D},I)$ and

$$\begin{bmatrix} \bar{W} & \bar{X} \\ \bar{Y} & \bar{Z} \end{bmatrix} = \hat{\mu}^*(\hat{L}).$$

Then $\mu^*(L) = \bar{X}$.

This provides the "general" solution of the splicing equation:

Corollary 6.7. *With $\bar{L}$ defined by Theorem* 6.3 *using $\hat{\mu}$ and $\hat{L}$,*

$$\mu^*(L) = \begin{bmatrix} I & O \end{bmatrix} \bar{L}^+ \begin{bmatrix} O \\ I \end{bmatrix}.$$

Proof of Proposition 6.6. First calculate

$$\hat{\mu}\begin{bmatrix} W & X \\ Y & Z \end{bmatrix} = \begin{bmatrix} W' & X' \\ Y' & Z' \end{bmatrix}$$

$$= \begin{bmatrix} W + B^{-1}W + W^2 + XY & X + B^{-1}X + XD^{-1} + WX + XZ \\ Y + YW + ZY & Z + ZD^{-1} + YX + Z^2 \end{bmatrix}.$$

We first apply this with $W = \bar{W}$, etc. Since $\bar{Y} \geqslant P$ we have $\bar{W} = \bar{W}' \geqslant \bar{X}\bar{Y} \geqslant \bar{X}P$, so $\bar{W}\bar{X} \geqslant \bar{X}P\bar{X}$. Hence $\bar{X} = \bar{X}' \geqslant \bar{X} + B^{-1}\bar{X} + \bar{X}D^{-1} + \bar{X}P\bar{X} = \mu(\bar{X})$. From this and $\bar{X} \geqslant L$ we conclude $\bar{X} \geqslant \mu^*(L)$.

For the opposite inclusion let $S = \mu^*(L)$ and assume

$$X \leqslant S,\ \ WS \leqslant S,\ \ SZ \leqslant S,\ \text{and}\ SYS \leqslant S.$$

Then the same is true for X', etc.,

$$\begin{aligned} X' &= X + B^{-1}X + XD^{-1} + WX + XZ \leqslant \mu(X) + WS + SZ \leqslant S,\\ W'S &= WS + (B^{-1}W)S + W(WS) + XYS\\ &\leqslant S + B^{-1}(WS) + WS + SYS \leqslant S + B^{-1}S \leqslant S,\\ SZ' &= SZ + S(ZD^{-1}) + SYX + (SZ)Z\\ &\leqslant S + (SZ)D^{-1} + SYS + SZ \leqslant S + SD^{-1} \leqslant S,\\ SY'S &= SYS + SY(WS) + (SZ)YS \leqslant SYS \leqslant S. \end{aligned}$$

Now a trivial induction using $\hat{\mu}^p(\hat{L})$ shows that $\bar{X} \leqslant S$. □

7. Calculations and examples

Gatterdam [4] (corrected in [5]) and Păun [13] considered the problem of realizing a given regular language as the splicing language $\sigma^*(L)$ of a finite H scheme σ and a finite initial language L. Not all regular languages can be so represented. However, any regular language may be represented as the image of such a $\sigma^*(L)$ under a *coding* (a homomorphism that maps each symbol to a single symbol).

For CPH and MH schemes we have a simpler construction, which is well known in the study of sofic systems. A similar representation of a regular language in matrix terms is in [17].

Proposition 7.1. *Suppose Q is a regular language over the alphabet A.*
1. *There are a finite CPH scheme $\kappa = (A, E, \emptyset, P)$ with $P = \{\eta\eta : \eta \in E\}$, a finite language $L \subset EA^*E$, and two symbols ε_1 and ε_n in E so that $Q = \varepsilon_1^{-1}\kappa^*(L)\varepsilon_n^{-1}$.*
2. *There are a finite $n \times n$ MH scheme $\mu = (A, O, O, I)$ and a finite matrix L so that $Q = \mu^*(L)_{1n} = (L^+)_{1n}$.*

Proof. Choose an automaton representation $Q = L(G, I, T)$. We may assume I and T are singletons by adding, if necessary, extra vertices and edges labeled by 1 between these new vertices and the original initial and terminal vertices. Let $E = \{\varepsilon_1, \dots, \varepsilon_n\}$ be the set of vertices in G, indexed so $I = \{\varepsilon_1\}$ and $T = \{\varepsilon_n\}$. This determines κ. The initial language L consists of all strings $\varepsilon_j z \varepsilon_k$ such that there is an edge in G from ε_j to ε_k labeled by z. It is then obvious that, for $j \neq k$, $\varepsilon_j^{-1}\kappa^*(L)\varepsilon_k^{-1} = L(G, \{\varepsilon_j\}, \{\varepsilon_k\})$. (If $j = k$ then $L(G, \{\varepsilon_j\}, \{\varepsilon_j\})$ contains 1, but we do not necessarily have $\varepsilon_j\varepsilon_j$ in $\kappa^*(L)$.)

The MH scheme μ is the one generated from κ by Proposition 5.1, and then part 2 is an immediate translation of part 1. In this context the matrix L becomes the incidence matrix for the graph G (with coefficients in A^*). □

Remark. Conversely, suppose $\bar{L}$ is an $n \times n$ matrix. Then $\bar{L}$ defines a graph G with n vertices v_j, so that there is an edge from v_j to v_k iff $\bar{L}_{jk} \neq \emptyset$. We then label each such edge with the language $\bar{L}_{jk}$, and we label any path in G with the concatenation of the languages at the edges. In this way we may consider G as a "generalized" automaton, so that the language $L(G, \{v_j\}, \{v_k\})$ is just the jk entry in $\bar{L}^*$. In case $\bar{L}$ is finite (or, in fact, regular) this generalized automaton may be easily converted to a standard automaton. Thus, for example, if we start with a finite H scheme σ and a finite initial language L then we can consider the proof of Theorem 6.4 as producing a matrix $\bar{L}$ so that $S = \bar{L}^+$ has one off-diagonal entry, say S_{jk}, equal to $\sigma^*(L)$. In this case we may interpret $\bar{L}$, with the initial and terminal vertices v_j and v_k, as an automaton recognizing $\sigma^*(L)$.

We want to explain the computational content of the solution of the splicing equation in Section 6. The basic idea is that a regular matrix has only finitely many quotients. To use this idea explicitly we introduce some notation: For a regular language L we define $\|L\|$ to be the minimum number of vertices of an automaton representing L as $L(G,I,T)$. For an $n \times n$ regular matrix X we let $\|X\|$ be the sum of $\|X_{jk}\|$, $1 \leqslant j,k \leqslant n$.

Lemma 7.2. *Suppose X is an $n \times n$ regular matrix of languages and Y_m and Z_m are increasing sequences of matrices. Then there are at most $n\|X\| + 1$ distinct matrices in each of the sequences $Y_m^{-1}X$ and XZ_m^{-1}, and there are at most $2n\|X\| + 1$ distinct matrices in the sequence $Y_m^{-1}XZ_m^{-1}$.*

Proof. First consider the case that $n = 1$, so X is just a regular language. Write $X = L(G,I,T)$ where G has $\|X\|$ vertices. Then, for a given language Y, $z \in Y^{-1}X$ iff $x = yz$ for some $x \in X$, $y \in Y$. This is equivalent to the requirement that there are paths p and q in G so that p starts in I and ends at some vertex v, q starts at v and ends in T, $\lambda(p) \in Y$ and $\lambda(q) = z$. In other words, $Y^{-1}X = L(G,J,T)$ where J is the set of vertices v in G such that $Y \cap L(G,I,\{v\}) \neq \emptyset$.

In the general case, in order to analyze $Y_m^{-1}X$ we look at the quotients $Y_{m,ij}^{-1}X_{ik}$. For each choice of m,i,j,k the possible quotients correspond to sets $J_{m,ijk}$ of vertices in a minimal size automaton for X_{ik}. In order for $Y_{m+1}^{-1}X$ to be strictly larger than $Y_m^{-1}X$ at least one of the sets $J_{m,ijk}$ must strictly increase as m increases to $m+1$. But each $J_{m,ijk}$ can strictly increase at most $\|X_{ik}\|$ times. Summing over i,j,k, we see that the sequence $Y_m^{-1}X$ can change values at most $n\|X\|$ times.

The other two cases are similar, starting with representations $XZ^{-1} = L(G,I,K)$ and $Y^{-1}XZ^{-1} = L(G,J,K)$. □

The following converts Theorem 6.3 into a procedure for calculating splicing matrices.

Theorem 7.3. *Let $\mu = (A,B,D,P)$ be a regular MH scheme with P a 0-1 matrix. Define the functions ϕ_0 and ϕ on matrices by*

$$\phi_0(Z) = ((I + ZP)^{-1}(I + B))^{-*}L((I + D)(I + PZ)^{-1})^{-*},$$
$$\phi(Z) = (\phi_0(Z)P)^*\phi_0(Z).$$

Then, for any matrix L :

1. $\phi^{m+1}(L) = \phi^m(L)$ *for some* $m \leqslant n(\|I+B\| + \|I+D\|)$*. If L is regular then also* $m \leqslant 2n\|L\|$.
2. *For any m, if* $\phi^{m+1}(L) = \phi^m(L)$ *then* $\mu^*(L) = \phi^m(L)$.

Remark. (1). This gives, in some sense, an algorithm for determining the splicing matrix $\mu^*(L)$. However, each step involves determining many quotients of languages. These may be determined algorithmically in the regular case. However, more generally, determining these quotients may not be algorithmically possible.

(2). This result may also be interpreted as follows: There is a fixed formula, in terms of a finite number of concatenations, quotients, unions and Kleene closures, for the splicing matrix. Specifically, for any L, $\mu^*(L) = \phi^M(L)$ where $M = n(\|I+B\|+\|I+D\|)$. This is not particularly useful, but it does provide an "explicit" formula for the splicing matrix.

(3). If neither μ nor L is regular then there is no guarantee that $\mu^*(L) = \phi^*(L)$, since the function ϕ does not generally satisfy the continuity hypothesis of Lemma 2.1.

Proof. Define $T_m = \phi^m(L)$, so $T_0 = L$ and T_m is an increasing sequence. Then part 1 follows from Lemma 7.2 applied to the two sequences $(I + T_mP)^{-1}(I + B)$ and $(I+D)(I+PT_m)^{-1}$. In the case L is regular we get the alternate estimate $m \leqslant 2n\|L\|$ from the two-sided version of Lemma 7.2.

Now suppose $T_m = T_{m+1}$ for some m. We first note that $T_m \leqslant S = \mu^*(L)$. In fact, this is an obvious induction since ϕ is order-preserving, $T_0 = L \leqslant S$, and, according to Theorem 6.3, $\phi(S) = S$. In order to show $S \leqslant T_m$ we need $L \leqslant T_m$, which is clear, and $\mu(T_m) \leqslant T_m$. Clearly $T_mPT_m \leqslant T_m$ so we only need $B^{-1}T_m \leqslant T_m$ and $T_mD^{-1} \leqslant T_m$. As usual we give the argument only for B. As in the proof of Theorem 6.3 there is no harm in assuming $I \leqslant B$ and $I \leqslant D$.

We can organize the calculation of T_m as follows (remembering $I \leqslant B$ and $I \leqslant D$):

$$B_k = B + (T_kP)^{-1}B, \qquad D_k = D + D(PT_k)^{-1}, \qquad L_k = \phi_0(T_k) = B_k^{-*}LD_k^{-*}$$
$$T_{k+1} = (L_kP)^*L_k. \tag{7.1}$$

Then $T_m = T_{m+1} = \tilde{\mu}^*(L_m)$ with $\tilde{\mu} = (A, O, O, P)$ so Lemma 6.2 applies: $B_m^{-1}T_m \leqslant \tilde{\mu}^*(B_m^{-1}L_m)$. Using $B \leqslant B_m$ and $B_m^{-1}L_m = B_m^{-1}(B_m^{-*}LD_m^{-*}) \leqslant L_m$, this implies $B^{-1}T_m \leqslant \tilde{\mu}^*(L_m) = T_m$. $\square$

We illustrate this algorithm with a simple example, which we first present in cutting and pasting terms. The alphabet is $A = \{a, b\}$ and the set of end markers is $E = \{\alpha, \beta, \gamma, \delta\}$. We consider the strings in $\alpha A^*\alpha$ to form the "finished" language, and we consider strings starting or ending with the other markers as fragments. There are three cutting actions:

$$xay \Longrightarrow \beta y; \quad xay \Longrightarrow x\gamma; \quad xbby \Longrightarrow x\beta$$

(the “left” fragment resulting from cutting at bb is not needed). There are two pasting operations:

$$(x\gamma, \beta y) \Longrightarrow xby; \quad (x\delta, \beta y) \Longrightarrow xaby.$$

Finally, the initial language is just $\{ab\}$.

We follow Proposition 3.4 to translate this into an MH system. With some ad hoc simplifications to keep the sizes of the matrices reasonably small we obtain a 4×4 system with the pasting strings in the initial matrix. After adding I to both B and D we have

$$B = \begin{bmatrix} 1 & A^*a & 0 & 0 \\ 0 & 1+A^*a & 0 & 0 \\ 0 & A^*a & 1 & 0 \\ 0 & A^*a & 0 & 1 \end{bmatrix}, \qquad D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ aA^* & aA^* & 1+aA^* & aA^* \\ bbA^* & bbA^* & bbA^* & 1+bbA^* \end{bmatrix},$$

$$P = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad L = \begin{bmatrix} ab & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & ab & 0 & 0 \end{bmatrix}.$$

We start with $T_0 = L$ and use the definitions in (7.1). Since $\|B\| = 11$, $\|D\| = 22$ and $\|L\| = 8$ we can expect to find the solution in at most $\min(4\cdot(11+22), 2\cdot 4\cdot 8) = 64$ steps. Fortunately, this is a very generous estimate in this case.

First iteration:

$$B_0 = B, \qquad D_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ aA^* & aA^* & 1+aA^* & 1+aA^* \\ bbA^* & bbA^* & b+bbA^* & 1+bbA^* \end{bmatrix}$$

$$L_0 = \begin{bmatrix} ab & 0 & 1 & 0 \\ b & b & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & ab & 1 & 0 \end{bmatrix}, \qquad T_1 = \begin{bmatrix} ab+bb^+ & b^+ & 1 & 0 \\ b^+ & b^+ & 0 & 0 \\ bb^+ & b^+ & 0 & 0 \\ abb^+ + bb^+ & ab^+ + b^+ & 1 & 0 \end{bmatrix}.$$

Second iteration:

$$B_1 = \begin{bmatrix} 1 & A^*a & 0 & 0 \\ 0 & 1+A^*a & 0 & 0 \\ 1 & A^*a & 1 & 1 \\ 0 & A^*a & 0 & 1 \end{bmatrix},$$

$$D_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ aA^* & aA^* & 1+aA^* & 1+aA^* \\ bbA^* & 1+b+bbA^* & 1+b+bbA^* & 1+b+bbA^* \end{bmatrix}$$

$$L_1 = \begin{bmatrix} ab & b & 1+b & 1+b \\ b & b & 1+b & 1+b \\ 0 & b & 1+b & 1+b \\ 0 & b+ab & 1+a+b+ab & 1+a+b+ab \end{bmatrix},$$

$$T_2 = \begin{bmatrix} ab+A^*bb & A^*b & A^* & A^* \\ b+A^*bb & A^*b & A^* & A^* \\ A^*bb & A^*b & A^* & A^* \\ A^*bb & A^*b & A^* & A^* \end{bmatrix}.$$

Third iteration:

$$B_2 = \begin{bmatrix} 1 & A^*a & 0 & 0 \\ 0 & 1+A^*a & 0 & 0 \\ 1 & 1+A^*a & 1 & 1 \\ 1 & 1+A^*a & 1 & 1 \end{bmatrix},$$

$$D_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ aA^* & 1+aA^* & 1+aA^* & 1+aA^* \\ bbA^* & 1+b+bbA^* & 1+b+bbA^* & 1+b+bbA^* \end{bmatrix}$$

$$L_2 = \begin{bmatrix} ab & b+ab & 1+a+b+ab & 1+a+b+ab \\ b & b+ab & 1+a+b+ab & 1+a+b+ab \\ 0 & b+ab & 1+a+b+ab & 1+a+b+ab \\ 0 & b+ab & 1+a+b+ab & 1+a+b+ab \end{bmatrix}, \qquad T_3 = T_2.$$

Thus, the splicing language corresponding to the original problem is $ab + A^*bb$, the 1,1 entry of T_2. Note that at this stage we can describe the splicing system very simply: The "enhanced" cutting matrices are

$$B_2^* = \begin{bmatrix} 1 & A^*a & 0 & 0 \\ 0 & 1+A^*a & 0 & 0 \\ 1 & 1+A^*a & 1 & 1 \\ 1 & 1+A^*a & 1 & 1 \end{bmatrix}, \qquad D_2^* = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ aA^*+bA^+ & A^* & A^* & A^* \\ aA^*+bA^+ & A^* & A^* & A^* \end{bmatrix}.$$

Then L_2 is the result of cutting L by these matrices, and the splicing matrix T_2 is obtained by pasting together a sequence of copies of L_2, with pasting matrix P.

Remark. Technically, it is necessary to verify that the pasting strings $\gamma b\beta$ and $\delta ab\beta$ have "extensions" in the sense of Proposition 3.4 before adding them to the initial matrix in L_{32} and L_{42}. This is in fact true and easy to verify before starting the iteration. In a more formal algorithmic process it would be necessary to start with all entries empty except for L_{11}. Then at the end of each iteration we can easily test whether various entries L_{jk} may be populated from the set of pasting strings.

8. Circular splicing

Informally, a "circular string" is a string $x_1x_2\dots x_n$ of symbols from the alphabet A, with the understanding that a cyclic permutation of the symbols, say $x_{k+1}\dots x_nx_1\dots x_k$, should represent the same circular string. To formalize this notion we define an equivalence relation $\sim$ on A^* by $xy \sim yx$, so two strings are equivalent iff one is a cyclic permutation of the other. We denote the set of equivalence classes by $A^\wedge$, and we write $^\wedge w$ for the equivalence class of the linear string $w \in A^*$. We say a subset C of $A^\wedge$ is a *circular language*, and a subset M of the disjoint union $A^* \cup A^\wedge$ is a *mixed language*.

The application of splicing rules to mixed strings was introduced in [7], and is described in more detail in [16]. We can summarize the use of a rule $u\#u'\$v'\#v$ as follows:

1. $(xuu'y', x'v'vy) \Longrightarrow xuvy$ (ordinary linear splicing).
2. $(^\wedge xuu', {}^\wedge yv'v) \Longrightarrow {}^\wedge xuvyv'u'$ (circular splicing).
3. $(xuu'y, {}^\wedge zv'v) \Longrightarrow xuvzv'u'y$ (mixed splicing).
4. $^\wedge xuu'yv'u' \Longrightarrow (^\wedge xuv, {}^\wedge yv'u')$ (circular self-splicing).
5. $xuu'yv'vz \Longrightarrow (xuvz, {}^\wedge yv'u')$ (mixed self-splicing).

More obviously, we have the following cutting and pasting operations, defined in terms of a cutting site $\delta u\varepsilon$ and a pasting string $\delta v\varepsilon$:

1. $xuy \Longrightarrow (x\delta, \varepsilon y)$ (linear cutting).
2. $(x\delta, \varepsilon y) \Longrightarrow xvy$ (linear pasting).
3. $^\wedge xu \Longrightarrow \varepsilon x\delta$ (circular cutting).
4. $\varepsilon x\delta \Longrightarrow {}^\wedge xv$ (self-pasting).

Suppose M is a mixed language, σ is an H scheme, and κ is a CPH scheme. We define $\sigma_m(M)$ as usual, using all five mixed splicing operations, and we reserve $\sigma(M)$ for the result of applying the linear operations to the linear strings in M, ignoring the circular strings. Similarly, we define $\kappa_m(M)$ and $\kappa(M)$.

We say that an H scheme $\sigma = (A,R)$ is *symmetric* iff whenever $u\#u'\$v'\#v$ is in R then $v'\#v\$u\#u'$ is in R, and we say that σ is *reflexive* iff whenever $u\#u'\$v'\#v$ is in R then both $u\#u'\$u\#u'$ and $v'\#v\$v'\#v$ are in R. Symmetry is a natural assumption to make in the circular context: All the splicing operations except ordinary linear splicing leave both uv and $v'u'$ in the result so, in effect, they use both $u\#u'\$v'\#v$ and $v'\#v\$u\#u'$. Reflexivity is an attempt to translate in splicing terms the idea that a ligase should always be able to rejoin the two fragments of a single DNA molecule cut by a restriction enzyme. Moreover, the reflexivity assumption is actually necessary for the closure theorem: See the example of Siromoney et al. [18], as discussed in [16], which shows that, without the reflexivity assumption, a finite initial circular language and a finite H scheme can generate a non-regular splicing language.

Note that the assumptions of symmetry and finitely many visible sites imply that the H scheme is finite.

We first translate from an H scheme to a CPH scheme, giving a version of Proposition 3.3 in the mixed context. We start with a finite reflexive and symmetric H scheme

$\sigma = (A,R)$. For each rule $r = u\#u'\$v'\#v$ in R we create two end markers δ_r and ε_r. We declare $\delta_r uu'\delta_r$ and $\varepsilon_r v'v\varepsilon_r$ to be both cutting sites and pasting strings, and we also declare $\delta_r uv\varepsilon_r$ and $\varepsilon_r v'u'\delta_r$ to be pasting strings. These determine the finite sets C and P, and, with the addition of one more marker ∇, the set E.

Proposition 8.1. *Suppose $\sigma = (A,R)$ is a finite reflexive and symmetric H scheme and define the finite CPH scheme $\kappa = (A,E,C,P)$ as above. For any mixed language $M \subset A^* \cup A^{\wedge}$, if we define $L = M \cap A^*$ and $Z = M \cap A^{\wedge}$ then we have*
1. $\sigma_m^*(M) \cap A^* = \nabla^{-1}(\kappa_m^*(\nabla L\nabla \cup Z) \cap A^*)\nabla^{-1}$;
2. $\sigma_m^*(M) \cap A^{\wedge} = \kappa_m^*(\nabla L\nabla \cup Z) \cap A^{\wedge}$.

Proof. This follows the general outline of the proof of Proposition 3.3.

The first half of the proof is the verification that each splicing operation on strings in $A^* \cup A^{\wedge}$ can be mirrored by a sequence of cutting and pasting operations in $(\nabla A^* \nabla) \cup A^{\wedge}$. Suppose that we have a splicing operation involving the rule $r = u\#u'\$v'\#v$. In the following we use, as needed, the cutting sites and pasting strings derived from r. Since only one rule is involved, we shall suppress the subscripts on the end markers.
1. Ordinary splicing $(xuu'y', x'v'vy) \Longrightarrow xuvy$: Cut $\nabla xuu'y'\nabla \Longrightarrow (\nabla x\delta, \delta y'\nabla)$ and cut $\nabla x'v'vy\nabla \Longrightarrow (\nabla x'\varepsilon, \varepsilon y\nabla)$; paste $(\nabla x\delta, \varepsilon y\nabla) \Longrightarrow \nabla xuvy\nabla$.
2. Circular splicing $({}^{\wedge}xuu', {}^{\wedge}yv'v) \Longrightarrow {}^{\wedge}xuvyv'u'$: Cut ${}^{\wedge}xuu' \Longrightarrow \delta x\delta$ and cut ${}^{\wedge}yv'v \Longrightarrow \varepsilon y\varepsilon$; paste $(\delta x\delta, \varepsilon y\varepsilon) \Longrightarrow \delta xuvy\varepsilon$; self-paste $\delta xuvy\varepsilon \Longrightarrow {}^{\wedge}xuvyv'u'$.
3. Mixed splicing $(xuu'y, {}^{\wedge}zv'v) \Longrightarrow xuvzv'u'y$: Cut $\nabla xuu'y\nabla \Longrightarrow (\nabla x\delta, \delta y\nabla)$ and cut ${}^{\wedge}zv'v \Longrightarrow \varepsilon z\varepsilon$; paste $(\nabla x\delta, \varepsilon z\varepsilon) \Longrightarrow \nabla xuvz\varepsilon$; paste $(\nabla xuvz\varepsilon, \delta y\nabla) \Longrightarrow \nabla xuvzv'u'y\nabla$.
4. Circular self-splicing ${}^{\wedge}xuu'yv'v \Longrightarrow ({}^{\wedge}xuv, {}^{\wedge}yv'u')$: Cut ${}^{\wedge}xuu'yv'v \Longrightarrow \varepsilon xuu'y\varepsilon$ and cut $\varepsilon xuu'y\varepsilon \Longrightarrow (\varepsilon x\delta, \delta y\varepsilon)$; self-paste $\varepsilon x\delta \Longrightarrow {}^{\wedge}xuv$ and self-paste $\delta y\varepsilon \Longrightarrow {}^{\wedge}yv'v$.
5. Mixed self-splicing $xuu'yv'vz \Longrightarrow (xuvz, {}^{\wedge}yv'u')$: Cut $\nabla xuu'yv'vz\nabla \Longrightarrow (\nabla x\delta, \delta yv'vz\nabla)$ and cut $\delta yv'vz\nabla \Longrightarrow (\delta y\varepsilon, \varepsilon z\nabla)$; paste $(\nabla x\delta, \varepsilon z\nabla) \Longrightarrow \nabla xuvz\nabla$ and self-paste $\delta y\varepsilon \Longrightarrow {}^{\wedge}yv'u'$.

As in the proof of Proposition 3.3 this mirroring shows that $(\nabla\bar{L}\nabla) \cup \bar{Z} \subset \kappa^*(\nabla L\nabla \cup Z)$, where we have written $\bar{L} = \sigma_m^*(M) \cap A^*$ and $\bar{Z} = \sigma_m^*(M) \cap A^{\wedge}$.

Now let κ_c be the CPH scheme $(A,E,C,\emptyset)$. For the second half of the proof we shall show that $\kappa_m^*(\nabla L\nabla \cup Z) \subset \kappa_{cm}^*(\nabla\bar{L}\nabla \cup \bar{Z})$.

To describe $\kappa_{cm}^*(\nabla\bar{L}\nabla)$ we use Lemma 3.1. For fragments of circular strings we have the following.

Lemma 8.2. *$\xi z\eta$ is in $\kappa_{cm}^*(\bar{Z})$ if and only if there are strings $\xi c\xi$ and $\eta c'\eta$ in C and a string ${}^{\wedge}cxc'y$ in $\bar{Z}$.*

Proof. The argument is identical to that of Lemma 3.1, except for the possible presence of strings which are the result of cutting a circular string in $\bar{Z}$ exactly once. For this to happen there must be a circular string ${}^{\wedge}zc$ in $\bar{Z}$ and a cutting site $\zeta c\zeta$ in C, so the result of the cut is $\zeta z\zeta$. If $\zeta c\zeta = \delta_r uu'\delta_r$ with $r = u\#u'\$v'\#v$ then, by reflexivity, $r' = u\#u'\$u\#u'$ is also in R. Using r' we can perform circular splicing on two copies of $\hat{}zc = \hat{}zuu'$ to produce $\hat{}zuu'zuu' = \hat{}zczc = \hat{}czcz$ in $\bar{Z}$, and this string satisfies the requirements of the Lemma. If $\zeta c\zeta = \varepsilon_r v'v\varepsilon_r$ then we proceed just as above, but using the rule $r'' = v'\#v\$v'\#v$, which is also provided by the reflexivity assumption. □

Let $W = \kappa^*_{cm}(\nabla\bar{L}\nabla \cup \bar{Z})$. In order to show $\kappa^*_m(\nabla L\nabla \cup Z) \subset W$ we only need to show that $\kappa_m(W) \subset W$, since $W \supset \nabla L\nabla \cup Z$ is obvious. Also $\kappa_{cm}(W) \subset W$ is trivial, so we need to show that W is closed under pasting operations.

Suppose that a pasting operation uses the pasting string $\xi v\eta \in R$. We need the following bookkeeping exercise:

There is a rule $r = u\#u'\$v'\#v$ in R so that ξ and η are in $\{\delta_r, \varepsilon_r\}$. Determine $r' = s\#s'\$t'\#t$ as follows:
1. If $\xi v\eta = \delta_r uu'\delta_r$ then $r' = u\#u'\$u\#u'$.
2. If $\xi v\eta = \varepsilon_r v'v\varepsilon_r$ then $r' = v'\#v\$v'\#v$.
3. If $\xi v\eta = \delta_r uv\varepsilon_r$ then $r' = u\#u'\$v'\#v$.
4. If $\xi v\eta = \varepsilon_r v'u'\delta_r$ then $r' = v'\#v\$u\#u'$.

In each case either $r' = r$ or r' is guaranteed by symmetry or reflexivity, so r' is in R. Moreover, the reader may check that $v = st$ and that $\xi ss'\xi$ and $\eta t' t\eta$ are exactly the cutting sites in C bounded by either ξ or η.

Suppose first that the pasting string $\xi st\eta$ is used to self-paste a string $\eta z\xi$ in W to produce $\hat{}zst$. Since $\eta z\xi$ is in W it is the result of cutting a string in $\nabla\bar{L}\nabla \cup \bar{Z}$. Hence, according to Lemma 3.1 or Lemma 8.2, there is a string $\nabla xt'tzss'y\nabla$ or $\hat{}xt'tzss'$ in $\nabla\bar{L}\nabla \cup \bar{Z}$. In either case this string can be self-spliced using r' to demonstrate that $\hat{}zst$ is in $\bar{Z} \subset \nabla\bar{L}\nabla \cup \bar{Z}$.

For the remaining case suppose that $\zeta z\xi$ and $\eta z'\zeta'$ in W are pasted using $\xi st\eta$ to produce $\zeta zstz'\zeta'$. Again by Lemma 3.1 or Lemma 8.2 we can find in $\nabla\bar{L}\nabla \cup \bar{Z}$ two strings; the first is either $\nabla xczss'y\nabla$ or $\hat{}czss'y$ and the second is either $\nabla x't'tz'c'y'\nabla$ or $\hat{}t'tz'c'y'$. We have either $\zeta = \nabla$ and $xc = 1$ or $\zeta c\zeta \in C$, and similarly either $\zeta' = \nabla$ and $c'y' = 1$ or $\zeta'c'\zeta' \in C$. In either case we can recover $\zeta zstz'\zeta'$ by first splicing these strings using r', and then cutting at c (if $\zeta \neq \nabla$) and at c' (if $\zeta' \neq \nabla$). For example, splicing $\hat{}czss'y$ and $\hat{}t'tz'c'y'$ using r' yields $\hat{}czstz'c'y't's'y$, and the fragments remaining after cutting at c and c' are $\zeta zstz'\zeta'$ and $\zeta'y't's'y\zeta$. There are three other possibilities: Two involve mixed splicing, and the last involves linear splicing. The details are essentially the same.

This completes the proof that $\kappa^*_m(\nabla L\nabla \cup Z) \subset W = \kappa^*_{cm}(\nabla\bar{L}\nabla \cup \bar{Z})$. As in the proof of Proposition 3.3 this implies the inclusion $\kappa^*(\nabla L\nabla \cup Z) \cap (\nabla A^*\nabla \cup \hat{}A) \subset \nabla(\sigma^*_m(M) \cap A^*)\nabla \cup (\sigma^*_m(M) \cap \hat{}A)$ since results of cutting operations are never in $\nabla A^*\nabla \cup \hat{}A$. This, with the reverse inclusion proved earlier, finishes the proof of the Proposition. □

Proposition 8.1 reduces the analysis of mixed H systems to the analysis of mixed CPH systems. Our next result shows that the evolution of a mixed CPH system can be explained as follows: First cut all possible initial circular strings; then allow the resulting mixed language to evolve purely as a linear CPH system (ignoring the circular strings); finally self-paste all possible linear strings to form circular strings.

Proposition 8.3. *Suppose that $\kappa = (A,E,C,P)$ is a CPH scheme and let $\kappa_c = (A,E,C,\emptyset)$ and $\kappa_p = (A,E,\emptyset,P)$. If $M \subset EA^*E \cup A^\wedge$ then*

$$\kappa_m^*(M) = \kappa_{pm}(\kappa^*(\kappa_{cm}(M))).$$

Proof. Clearly $\kappa_m^*(M) \supset S' = \kappa_{pm}(\kappa^*(\kappa_{cm}(M)))$. To prove the opposite inequality we only need $\kappa_m(S') \subset S'$. Now $\kappa^*(\kappa_{cm}(M))$ is closed under linear operations, so the only difference between S' and $\kappa^*(\kappa_{cm}(M))$ is the result of self-paste operations. Since linear cut and paste operations do not see this difference they cannot produce strings outside of S'. Similarly, self-pasting elements of S' cannot produce anything not already in S'. So we only have to consider the result of cutting one of the circular strings that resulted from self-pasting.

Suppose $\varepsilon z\delta \in \kappa^*(\kappa_{cm}(M))$ is self-pasted using $\delta v\varepsilon \in P$ to produce $^\wedge zv$ in S'. Suppose $^\wedge zv = {}^\wedge z'v'$ and $\delta'v'\varepsilon'$ is in C, so $\varepsilon'z'\delta'$ is in $\kappa_m(S')$. To see that $\varepsilon'z'\delta'$ is actually in S' we proceed as follows. First, paste together three copies of $\varepsilon z\delta$ using $\delta v\varepsilon$ to produce $\varepsilon zvzvzvz\delta$ in $\kappa^*(\kappa_{cm}(M))$. Now $^\wedge zv = {}^\wedge z'v'$ means that $zv = xy$ and $z'v' = yx$ for some strings x and y. Hence $\varepsilon zvzvzvz\delta = \varepsilon xyxyxyz\delta = \varepsilon xz'v'z'v'yz\delta$. This string can now be cut twice, using $\delta'v'\varepsilon' \in C$, to produce $\varepsilon'z'\delta'$ in $\kappa^*(\kappa_{cm}(M)) \subset S'$, as desired. □

The following is an immediate consequence of this and Proposition 8.1. It states that if the initial language is purely linear then the linear part of the splicing language can be derived entirely by linear splicing (or cut and paste) operations.

Corollary 8.4. 1. *If $M \subset EA^*E$ and κ is any CPH scheme then $\kappa_m^*(M) \cap (EA^*E) = \kappa^*(M)$.*
2. *If $M \subset A^*$ and σ is a finite symmetric and reflexive H scheme then $\sigma_m^*(M) \cap A^* = \sigma^*(M)$.*

Before our last result we need to explain the application of AFL terminology to circular and mixed languages.

If $L \subset A^*$ then we define the *circularization of* L as $\mathrm{Cir}(L) = \{^\wedge w : w \in L\}$. Conversely, if $C \subset A^\wedge$ we call any language $L \subset A^*$ for which $\mathrm{Cir}(L) = C$ a *linearization of* C, and we define the *full linearization of* C as $\mathrm{Lin}(C) = \{w : {}^\wedge w \in C\}$. If $\mathscr{F}$ is a family of languages we say a circular language C is in $\mathscr{F}$ iff some linearization of C is in $\mathscr{F}$, and we say a mixed language M is in $\mathscr{F}$ iff both $M \cap A^*$ and $M \cap A^\wedge$ are in $\mathscr{F}$. If $L \subset A^*$ then the set of all cyclic permutations of words in L is denoted Cycle(L) and is called the *cyclic closure* of L; it is immediate that Cycle(L)=Lin(Cir(L)). We say that a family of languages $\mathscr{F}$ is closed under cyclic closure iff Cycle(L) is in $\mathscr{F}$ whenever L is in $\mathscr{F}$. It is shown in [9] that the regular family and the context-free family are closed under cyclic closure.

Lemma 8.5. *If $\mathscr{F}$ is closed under cyclic closure then a circular language C is in $\mathscr{F}$ iff Lin(C) is in $\mathscr{F}$.*

The proof is immediate, since if L is a linearization of C which is in $\mathscr{F}$ then Lin(C) $=$ Cycle(L) is in $\mathscr{F}$.

From this lemma the basic closure properties for AFL's of circular languages follow:

Corollary 8.6. *If $\mathscr{F}$ is a full AFL which is closed under cyclic closure then the circular languages in $\mathscr{F}$ are closed under finite union, intersection with regular circular languages, and forward and inverse homomorphic images.*

Now finally we have the precise form of the Circular Closure Theorem of Section 1.

Theorem 8.7 (Circular Closure Theorem). *Suppose $\mathscr{F}$ is a full AFL which is closed under cyclic closure.*
1. *If κ is a regular CPH scheme and $M \subset (EA^*E) \cup A\hat{}$ is in $\mathscr{F}$ then $\kappa_m^*(M)$ is in $\mathscr{F}$.*
2. *If σ is a finite symmetric and reflexive H scheme and $M \subset A^* \cup A\hat{}$ is in $\mathscr{F}$ then $\sigma_m^*(M)$ is in $\mathscr{F}$.*

Proof. The H version reduces to the CPH version via Proposition 8.1.

For the CPH version we use Proposition 8.3. We first need to consider $\kappa_{cm}(M)$. Let $L = M \cap (EA^*E)$ and $Z = M \cap A\hat{}$, so $\kappa_{cm}(M) = \kappa_c(L) \cup \kappa_{cm}(Z)$. Clearly (from the matrix formulation) $\kappa_c(L)$ is in $\mathscr{F}$, so we need to consider $\kappa_{cm}(Z)$. For each pair δ, ε of end markers we form $C_{\delta\varepsilon} = \delta^{-1}C\varepsilon^{-1}$. Then $C_{\delta\varepsilon}$ is just the set of cutting sites bounded by δ and ε, but without the end markers. We let $Z_{\delta\varepsilon} = Z \cap$ Cir $(A^*C_{\delta\varepsilon})$, the set of strings in Z which contain a site bounded by δ and ε. Since $C_{\delta\varepsilon}$ is regular, this is in $\mathscr{F}$. Now we can describe a cutting operation on a string of Z as follows: Linearize the string, cyclically permute it so it ends in a site, cut off the site and add appropriate end markers. In other words, if $\tilde{L}$ is the set of results of circular cutting, then

$$\tilde{L} = \kappa_{cm}(Z) \cap EA^*E = \bigcup_{\delta\varepsilon} \varepsilon(\text{Lin }(Z_{\delta\varepsilon})C_{\delta\varepsilon}^{-1})\delta.$$

From this it is clear that $\kappa_{cm}(Z) = Z \cup \tilde{L}$ is in $\mathscr{F}$.

Next, $\kappa^*(\kappa_{cm}(M)) = \kappa_{cm}(M) \cup \kappa^*(L \cup \tilde{L})$, which is in $\mathscr{F}$ by the Closure Theorem 6.4.

Finally, consider pasting operations on strings in $\kappa^*(\kappa_{cm}(M))$. Linear pasting will not produce anything new, so consider self-pasting. Specifically, let $\bar{L} = \kappa^*(\kappa_{cm}(M)) \cap EA^*E$ (so $\bar{L}$ is in $\mathscr{F}$) and let $\bar{Z}$ be the set of circular strings resulting from self-pasting operations on strings in $\bar{L}$. Define $P_{\delta\varepsilon} = \delta^{-1}P\varepsilon^{-1}$. Then we can describe a self-pasting operation as follows: Find a string in $\bar{L}$ which begins with ε and ends with δ; remove the end markers; append an element of $P_{\delta\varepsilon}$; and circularize. That is,

$$\bar{Z} = \bigcup_{\delta\varepsilon} \text{Cir }((\varepsilon^{-1}\bar{L}\delta^{-1})P_{\delta\varepsilon}).$$

From this it is clear that $\bar{Z}$ is in $\mathscr{F}$, so $\kappa_m^*(M) = \kappa_{pm}(\kappa^*(\kappa_{cm}(M))) = \bar{Z} \cup \kappa^*(\kappa_{cm}(M))$ is in $\mathscr{F}$. $\square$

References

[1] E. Csuhaj-Varjú, R. Freund, L. Kari, G. Păun, DNA computation based on splicing: universality results, L. Hunter and T. Klein, (Eds.), Biocomputing: Proc. 1996 Pacific Symp., Maui, HI, World Scientific Publishing Co., Singapore, 1996.

[2] K. Culik II, T. Harju, Splicing semigroups of dominoes and DNA, Discrete Appl. Math. 31 (1991) 261–277.

[3] R. Freund, E. Csuhaj-Varjú, F. Wachtler, Test tube systems with cutting/recombination operations, Pacific Symp. on Biocomputing, 1997.

[4] R.W. Gatterdam, Splicing systems and regularity, Internet. J. Comput. Math. 31 (1989) 63–67.

[5] R.W. Gatterdam, DNA and twist-free splicing systems, in: M. Ito and H. Jürgensen, (Eds.), Words, Languages and Combinatorics, vol. 2, World Scientific, Singapore, 1994, pp. 170–178.

[6] T. Head, Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors, Bull. Math. Biol. 49 (6) (1987) 737–759.

[7] T. Head, Splicing systems and DNA, in: Grzegorz Rozenberg and Arto Salomaa, (Eds.), Lindenmayer Systems: Impacts on Theoretical Computer Science, Computer Graphics and Developmental Biology, Springer, Berlin, Heidelberg, New York, 1992, pp. 371–383.

[8] T. Head, G. Păun, D. Pixton, Generative mechanisms suggested by DNA Recombination, in: G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages. vol. 2, Springer, Berlin, 1996.

[9] J. Hopcroft, J. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison–Wesley, Reading MA, 1979.

[10] L. Kari, DNA computing: arrival of biological mathematics, Math. Intelligencer 19 (2) (1997) 9–22.

[11] S.M. Kim, Computational modeling for genetic splicing systems, SIAM J. Comput., to appear.

[12] G. Păun, On the power of the splicing operation, Int. J. Comput. Math. 59 (1995), 27–35.

[13] G. Păun, On the splicing operation, Discrete Appl. Math. 70 (1) (1996) 57–79.

[14] G. Păun, Regular extended H systems are computationally universal, J. Automata, Languages, Combin. 1 (1) (1996), 27–36.

[15] D. Pixton, Linear and circular splicing systems, Proc. 1st Int. Symp. on Intelligence in Neural and Biological Systems (Herndon, VA), IEEE Computer Society Technical Committee on Pattern Recognition and Machine Intelligence (PAMI), IEEE Computer Society Press, Silver Spring, 1995, pp. 181–188.

[16] D. Pixton, Regularity of splicing languages, Discrete Appl. Math. 69 (1–2) (1996) 99–122.

[17] A. Salomaa, M. Soittola, Automata, Theoretic Aspects of Formal Power Series, Springer, Berlin, 1978.

[18] R. Siromoney, K.G. Subramanian, V.R. Dare, Circular DNA and splicing systems, A. Nakamura, M. Nivat, A. Saoudi, P.S.P. Wang, K. Inoue (Eds.), Proceedings of Parallel Image Analysis, 2nd Int. Conf. ICPIA '92, Ube, Japan, 21-23 Dec 1992. (Ube, Japan) Lecture Notes in Computer Science, no. 654, Springer, Berlin, Heidelberg, New York, 1992, pp. 260–273.