Regular splicing languages and subclasses

Paola Bonizzoni*, Giancarlo Mauri

Dipartimento di Informatica Sistemistica e Comunicazione, Università degli Studi di Milano – Bicocca, Via Bicocca degli Arcimboldi 8, 20126 Milano, Italy

Abstract

Recent developments in the theory of finite splicing systems have revealed surprising connections between long-standing notions in the formal language theory and splicing operation. More precisely, the syntactic monoid and Schützenberger constant have strong interaction with the investigation of regular splicing languages. This paper surveys results of structural characterization of classes of regular splicing languages based on the above two notions and discusses basic questions that motivate further investigations in this field.

In particular, we improve the result in [6] that provides a structural characterization of reflexive symmetric splicing languages by showing that it can be extended to the class of all reflexive splicing languages: this is the larger class for which a characterization is known.

© 2005 Elsevier B.V. All rights reserved.

Keywords: Automata; Regular languages; Molecular computing; Splicing systems

1. Introduction

The formal model of splicing system has been originally introduced in [13] to investigate the potentiality of a fundamental biological mechanism occurring in nature: restriction enzymes act on double-stranded DNA molecules by cutting them into segments that can be joined in the presence of ligase enzyme. The original definition of splicing system was formulated to describe the biochemical processes involved in molecular cut and paste phenomenon. Later the notion was reformulated by G. Paun at a less restrictive level of
generality, giving rise to the model of splicing operation that was then commonly adopted in splicing systems theory, and is nowadays a standard [19]. Since, a splicing system is a formal device to generate languages, called splicing languages, the splicing operation has been deeply investigated in the framework of formal language theory, by establishing a link between biomolecular sciences and language theory [20]. Moreover, this strict connection has contributed to a novel interest for the development of language theory. On the other side, theoretical results in splicing systems theory suggested new ideas in the framework of biomolecular science, for example the design of automated enzymatic processes.

In this paper, we focus on the original concept of finite splicing system that is closest to the real biological process: the splicing operation is meant to act by a finite set of rules (modelling enzymes) on a finite set of initial strings (modelling DNA sequences). Under this formal model, a splicing system is a generative mechanism of languages, which turn out to be regular splicing languages. This basic result has been first proved in [9], and later proved in several other papers by using different approaches (see for example [23,17,26]). More precisely, not all regular languages can be generated by splicing and a characterization of the class of regular splicing languages is still unknown. This open question is related to several challenging issues concerning splicing theory and formal language theory that motivate a continuous development of the research in this direction [12,6]. Several progress have been made in the investigation of the generative power of finite splicing systems.

For a better understanding of the basic issues in this field, it is necessary to classify splicing systems w.r.t. the splicing operation. In the literature three main splicing operations have been introduced, known as Head, Paun and Pixton operations. Each splicing operation leads to distinct classes of splicing languages (known also as Head, Paun, Pixton splicing languages) generated by splicing systems. Actually, it turns out that the relationship between the different classes of splicing languages can be explained by using the classical notion of Paun splicing operation and viewing the set of splicing rules as inducing a binary relation. A set $R$ of rules consists of ordered pairs of factored words, denoted as $((u_1, u_2)S(u_3, u_4))$, called rules, where $u_1u_2, u_3u_4$ are splicing sites. The set $R$ specifies a binary relation between factored sites that can be reflexive, symmetric or a transitive relation as shown in [4]. It turns out that distinct classes of splicing languages are generated by splicing systems where $R$ is a binary relation that obeys different restrictions. For instance, when $R$ is restricted to be reflexive, symmetric and transitive it allows one to characterize splicing languages generated by the original Head splicing operation. On the other hand, Paun splicing languages are generated by splicing systems where the set $R$ of rules are not necessarily symmetric or reflexive.

In particular, reflexivity and symmetry are natural properties for splicing systems as originally defined in [15]. More precisely, reflexive and symmetric splicing systems are the most important from a modelling perspective.

The first characterization of reflexive symmetric splicing languages has been given in [6] by using the concept of constant, introduced by Schützenberger [24]. Every language $L$ in this class is constructed from a finite set of constants for $L$, as $L$ is expressed by a finite union of constant languages and split languages, where a split language is a language obtained by a single iteration of a splicing operation over constant languages.

In this paper, we discuss this result which is a first significative progress in this research field, as it sheds light on the fundamental concepts in formal language theory that can explain
the generative power of splicing operation and how they can be used in this framework: these are the concepts of constant and of syntactic congruence.

Moreover, we improve the result of structural characterization given in [6], by showing that it generalizes to all reflexive (i.e., not necessarily symmetric) splicing languages: this result is stated in Proposition 4.2. Furthermore, by observing that a reflexive regular splicing language $L$ is characterized by one iteration step of splicing rules applied to constant languages, we prove that a recent characterization of reflexive languages given in [12] can be obtained as a Corollary of Proposition 4.2.

Two fundamental questions arise when dealing with splicing systems.

- **Question A: recognition**
  Give an effective procedure to decide whether a regular language is an $X$-splicing language ($X$ reflexive, symmetric).

- **Question B: synthesis**
  Give an effective procedure to construct, given $L$ an $X$-splicing language, a splicing system $S$ with $X$-rules such that $L = L(S)$.

In the paper, we address these two questions by presenting and analyzing main results related to them appearing in the literature. In particular, question A has been solved for the class of reflexive splicing languages (in [12] a decision procedure for this class has been proposed), and for special subclasses of regular languages. Clearly, the problem is strictly related to the question of providing a structural characterization of splicing languages. A graph-based algorithm that solves this question for null context splicing languages (NCS) is proposed in [7]. Other decision results have been given for larger classes of languages including the class NCS, such as the classes of FCS languages and of marker languages characterized by the notions of constant and of syntactic monoid. Question B has been solved for the class of reflexive symmetric languages in [6].

The paper is organized as follows. In Section 2 basic notions on finite splicing systems are given, Sections 3 and 4 are devoted to Question B, while in Section 5 we discuss results concerning Question A. Finally, in Section 6 we list open problems in this research field.

### 2. Finite splicing systems

In this section, we give the basic notions of finite splicing systems theory and of formal language theory that have been used to investigate subclasses of splicing languages.

Let $A$ be a finite alphabet. We denote the empty word over $A$ by $1$. Assume that $w \in A^+$, a 2-factor of $w$ is an ordered pair $(w_1, w_2)$ such that $w_1, w_2 \in A^*$ and $w = w_1w_2$. A rule $r$ consists of a pair of 2-factors $(u_1, u_2)$ and $(u_3, u_4)$ and is denoted as $((u_1, u_2)(u_3, u_4))$: each single word $u_1u_2$ and $u_3u_4$ is called splicing site of rule $r$. A set $R$ of rules specifies a binary relation between 2-factors of sites that can be reflexive, symmetric or even a transitive relation [4].

Precisely, $R$ is reflexive if given $((u_1, u_2)(u_3, u_4)) \in R$, then $((u_1, u_2)(u_3, u_2)) \in R$ and $((u_3, u_4)(u_1, u_4)) \in R$, while $R$ is symmetric if given $((u_1, u_2)(u_3, u_4)) \in R$, then $((u_3, u_4)(u_1, u_2)) \in R$. 
In this paper, we deal with finite splicing systems—that is, splicing systems where
the splice site \( u_1 u_2 \) and \( y \) has factor the splice site \( u_3 u_4 \), that is \( x = x_1 u_1 u_2 x_2 \) and \( y = y_1 u_3 u_4 y_2 \).
Then the result of a splicing operation of \( x, y \) by rule \( r \) is the word \( w = x_1 u_1 u_4 y_2 \) which is said to be generated by splicing of \( x, y \) by \( r \). If \( R \) is symmetric, since given rule \( r = ((u_1, u_2)S(u_3, u_4)) \in R \), also \( r = ((u_3, u_4)S(u_1, u_2)) \in R \), then word \( w = y_1 u_3 u_2 x_2 \) is generated by splicing of \( x, y \) by \( r \).

Let \( L \subseteq \mathbb{A}^* \). We define the closure of \( L \) by \( R \) as the set \( cl(L, R) \) of all words that are obtained as the result of a splicing operation of a pair of words in \( L \) by a rule \( r \in R \).

A splicing system \( S \) consists of a triple \( S = (A, I, R) \), where \( A \) is the alphabet of the system, \( R \) is a set of splicing rules and \( I \subseteq \mathbb{A}^* \) an initial language.

Given a splicing system \( S = (A, I, R) \), the iterated splicing is defined as follows:

\[
\sigma^0(I) = I, \quad \text{and} \quad \sigma^{i+1}(I) = \sigma^i(I) \cup cl(\sigma^i(I), R), \quad \text{for } i \geq 0,
\]

\[
\sigma^*(I) = \bigcup_{i \geq 0} \sigma^i(I).
\]

In this paper, we deal with finite splicing systems that is splicing systems where \( I \) and \( R \) are meant to be finite sets: in this case \( S \) is called \( H \)-system and \( L(S) = \sigma^*(I) \) is the splicing language generated by \( S \). Thus in the rest of the paper, by a splicing system we mean a finite splicing system and a splicing language is a language generated by a finite splicing system.

For convenience, we assume that all rules in \( R \) are useful for the language \( L(S) \), that is, for each rule \( r \in R \), there exist \( w, x, y \in L(S) \) such that \( w \) is generated by splicing of \( x, y \) by \( r \).

A splicing language \( L \) is a reflexive or symmetric splicing language if \( L = L(S) \), where \( S = (A, I, R) \) and \( R \) is reflexive or symmetric, respectively.

It must be pointed out that in the literature at least two other notions of splicing rules and splicing operations have been proposed. These are known as Head and Pixton splicing operations, respectively. In [8] it has been shown that splicing systems based on Pixton splicing operation are more powerful than the ones based on the standard (Paun) splicing, and these systems are more powerful than Head splicing systems.

A classification of these different notions of splicing may be given by using the standard (Paun) splicing operation adopted also in this paper, simply by requiring that the set \( R \) of rules is a specific (symmetric, reflexive or transitive) binary relation over 2-factors, as pointed out partially in [4].

The relationship between symmetric and nonsymmetric splicing languages has been investigated in [25]. The class of splicing languages (called 1-splicing languages and introduced in [20]) includes properly the class of symmetric splicing languages as proved in [25] (these are equivalent to the 2-splicing languages introduced in [20]), indeed, the languages of Lemma 4.3 show the strict inclusion.

**Remark 2.1.** Observe that we use a definition of \( cl(I, R) \) based on the 1-splicing operation [20]. This notion is generalized to the case of 2-splicing operation by defining the set \( cl_2(I, R) = \{x_1 u_1 u_4 y_2, y_1 u_3 u_2 x_2 : x_1 u_1 u_2 x_2, y_1 u_3 u_4 y_2 \in I, ((u_1, u_2)S(u_3, u_4)) \in R\} \), while \( cl(I, R) = \{x_1 u_1 u_4 y_2 : x_1 u_1 u_2 x_2, y_1 u_3 u_4 y_2 \in I, ((u_1, u_2)S(u_3, u_4)) \in R\} \).
Given $R$ a set of rules, let us denote by $\text{sym}(R)$ the symmetric closure of set $R$. Formally, $\text{sym}(R) = \{(s_1s_2), (s_2s_1) : (s_1s_2) \in R\}$. Then it is immediate to verify that $\text{cl}_2(I, R) = \text{cl}(I, \text{sym}(R))$. Vice versa, given $\text{cl}(I, R)$, where $R$ is a set of symmetric rules, then $\text{cl}(I, R) = \text{cl}_2(I, R)$.

In [26], a proof that splicing languages are regular languages is given, thus providing an alternative proof of the known inclusion of splicing languages in the class of regular languages. Actually, this main result in splicing theory has been firstly proved in [9], but there are several other proofs using different approaches (see for example [23,17]). For example, in [17], an algorithm has been given to construct a finite state automaton that recognizes the language generated by a splicing system $(A, L, R)$ that is not necessarily finite, as $L$ is a regular language and $R$ is a finite set. Clearly, this result proves the existence of a finite state automaton that recognizes a splicing language generated by a finite splicing systems, i.e., in the special case $L$ is finite.

A fundamental property introduced in several papers [6,5,11,12] relating rules to a language is the closure of $L$ under a set $R$ of rules, stated below.

**Definition 2.1.** A language $L$ is closed under a set $R$ of rules iff $\text{cl}(L, R) \subseteq L$.

We conclude the section by giving the basic notions of formal language theory used in the paper: these are the notions of a constant and syntactic monoid.

In this paper, when dealing with a finite state automaton $A = (A, Q, \delta, q_0, F)$ recognizing a regular language $L$, we assume that $A$ is trim, that is each state is accessible and coaccessible, and is the minimal automaton of $L$ (see [21] for basic notions). Then $\delta$ is the transition function of the deterministic automaton $A$, $q_0$ is the initial state, $F$ the set of final states [2,16]. Given $L$ a regular language, the reduced graph $\mathcal{G}_A(L)$, denotes the transition diagram for the minimal automaton $A$ recognizing $L$. A path $\pi$ in the reduced graph $\mathcal{G}_A(L)$ is a finite sequence $\pi = (q, a_1, q_1)(a_2, q_2)\ldots(q_n−1, a_n, q_n)$ where for each $i = 1,\ldots,n−1$, then $\delta(q_i, a_{i+1}) = q_{i+1}$ and $\delta(q_1, a_1) = q_1$. An abbreviated notation for a path is $\pi = (q, a_1a_2\ldots a_n, q_n)$ and $a_1a_2\ldots a_n$ is called the label of $\pi$. A path $\pi = (q, x, q_n)$ with $x \in A^+$, is a closed path iff $q = q_n$. Moreover, we say that $q, q_1,\ldots,q_n$ are the states crossed by the path $(q, a_1\ldots a_n, q_n)$ and, for each $i \in \{1,\ldots,n−1\}$, $q_i$ is an internal state crossed by the same path. Given $w \in A^+$ and $q \in Q$, then $Q_w$ denotes the set $\{q \in Q : \delta(q, w) = q', q' \in Q\}$. Given $m \in A^+$, we define the left context and right context of $m$ w.r.t. $L$, the language $\text{Cont}_L(L, m) = \{x \in A^* : xmy \in L\}$ and $\text{Cont}_R(L, m) = \{y \in A^* : xmy \in L\}$, respectively. Moreover, given $q \in Q_m$, then $\text{Cont}_{R,q}(L, m) = \{y \in A^* : \delta(q, my) \in F\}$.

A word $w \in A^+$ is a constant for a regular language $L$ iff given $xwy \in L$ and $zwy \in L$, then $xwy, zwy \in L$, i.e., $\text{Cont}_L(L, w)w\text{Cont}_R(L, w) \subseteq L$. The notion of constant has been introduced by Schützenberger [24]. A word $w \in A^+$ is singular iff $|Q_w| = 1$.

A characterization of constants of a regular language $L$ in terms of the reduced graph $\mathcal{G}_A(L)$ is given in Proposition 2.1. This result is more or less a folklore and its proof can be found in [6].
Proposition 2.1. Let $L \subseteq A^*$ be a regular language and let $G_A(L)$ be the reduced graph
for $L$. A word $m \in A^*$ is a constant for $L$ if and only if there exists $q_m \in Q$ such that
for each path $\pi$ in $G_A(L)$ which has $m$ as a label, there exists $q \in Q_m$ such that $\pi = (q, m, q_m)$.

The syntactic congruence plays a central role in the development of regular language
theory [21,22]. The syntactic congruence $\equiv_L$ w.r.t. a language $L$ is a binary relation over
words: $u \equiv_L z$ iff for all $x, y \in A^*$, $xwy \in L$ if and only if $xzy \in L$. The quotient $A^*$
w.r.t. the congruence $\equiv_L$ is the syntactic monoid of $L$, $M(L)$. In the paper, the equivalence
class of word $x$ is denoted as $[x]$.

3. Classes of splicing languages

The notion of constant appears to be crucial in characterizing the computational power
of splicing systems. Indeed, the structure of reflexive languages, as well as of other special
classes of splicing languages below the regular ones, is defined in terms of constants, as
proved in recent papers [6,12].

3.1. Constant and splicing languages

The first characterization of classes of splicing languages in terms of the concept of
constant is given in the seminal work on splicing operation [13]: the class of null context
splicing languages (NCS, in short) is equal to the one of strictly locally testable languages.
This result is based on a characterization of strictly locally testable languages (SLT) by
means of the concept of constant given by De Luca and Restivo in [10]. In [10], SLT are
characterized as those languages for which there exists a positive integer $k$ such that every
string in $A^*$ of length $k$ is a constant. Let us recall that null context splicing languages are
those languages generated by a system $S = (A, I, R)$ where each rule $r \in R$ is of the form
$((x, 1) \# (x, 1))$ or $((1, x) \# (1, x))$, for $x \in A^+$.

A crucial notion in finite splicing theory that has been firstly introduced in [14] is that of
constant language.

Definition 3.1 (constant language for $m$). Let $L$ be a regular language and $m$ be a word in
$A^*$ that is a constant for $L$. A constant language in $L$ for $m$ is the language $L(m, L) \subseteq L$
such that $L(m, L) = Cont_L(L, m)mCont_R(L, m)$. A language $L$ is simply a constant language
if $L(m, L) = L$.

In the paper, for simplicity we use the notation $L(m)$ for denoting a constant language
$L(m, L)$ in $L$.

Null context splicing languages are properly included in a larger class of languages
investigated in [14] that we call finitely constant generated splicing languages, or simply
FCS languages. These languages are the splicing languages generated by systems with one-
sided rules that are reflexive, which generalize the rules of NCS languages: one-sided rules
are rules of the form $(1, v)\#(1, u)$ or $(v, 1)\#(u, 1)$, for $u, v \in A^*$. 
The language $L = b^+ab^+$ is an example of FCS language that is not a NCS language, as indeed $L$ is not strictly locally testable. Moreover, note that a NCS language is not necessarily a constant language, as it holds in the case of language $L = a^* \cup b^*$, as $L$ is an NCS language that is union of two constant languages over two distinct symbols of the alphabet.

As for NCS languages, a nice characterization of FCS languages is given in terms of constants in [14]: a language $L$ is a FCS language if it is a finite union of a finite set with a finite set of constant languages in $L$ for a set $\mathcal{M}$ of constants of $L$ (these languages are called finitely constant based languages in [12]). This result is stated in the following theorem.

**Theorem 3.1 (FCS languages (Head [14])).** Let $L \subset A^*$ be a regular splicing language. Then the following are equivalent.

1. $L$ is generated by a splicing system $S = (A, I, R)$, where each rule $r \in R$ is one-sided and $R$ is reflexive.
2. $L = \bigcup_{m \in \mathcal{M}} L(m) \cup X$, where $X$ is a finite subset of $A^*$, $L(m)$ is a constant language in $L$ for $m \in \mathcal{M} \subseteq A^*$ and $\mathcal{M}$ is a finite set of constants for $L$.

### 3.2. Syntactic monoid and splicing languages

The notion of syntactic monoid has been used in [3] and in [5] in order to characterize new classes of regular languages generated by splicing. An example of how the use of the syntactic monoid may provide new insight in the investigation of splicing languages is obtained by naturally extending the notion of a constant language introducing congruence classes in place of constants. Precisely, in [3] it has been proved that regular languages of the form $L = L_1[x]_1 L_2$, where $L_1$ and $L_2$ are regular languages and $[x]_1$ is a marker that is defined by means of a syntactic congruence class $[x]$ of $M(L)$ are splicing languages, called marker languages. More precisely, a marker $[x]_1$ for the congruence class $[x]$ is defines as $[x]_1 = [x]$ if $[x]$ is finite, otherwise $[x]_1 = [x] \cup \{1\}$ where $x$ is a label of a closed path that is singular. Marker languages form a class of regular splicing languages which is not comparable to the class of FCS languages [5].

More precisely, there are regular languages that are marker languages of the form $L_1[x]_1 L_2$ and are not in the class FCS, even though they are generated by splicing, as shown in the following example.

**Example 3.1.** The regular language $L = L_1(ab^+a)^* L_2 = b^+a^2da(ab^+a)^*ada^2b^+$, with $L_1 = b^+a^2da$ and $L_2 = ada^2b^+$ is a marker language which is not in the class FCS. First observe that $(ab^+a)^+$ is a syntactic congruence class of language $L$, and thus $\{(aba)\} \cup \{1\}$ is a marker as $aba$ is the singular label of a closed path. The language $L$ is not in the class FCS, as it is an infinite union of constant languages as proved below. Let us first show that every factor of language $L_1ab^+ = b^+aL_2$ is not a constant. Indeed, assume that $z$ is a factor of $w \in L_1ab^+$, that is $w = w_1zw_2$. As $w \in b^+aL_2$, it follows that there exists a word $y \in L$ such that $y = ww$, as $L \supseteq L_1ab^+b^+aL_2$. Given $y = w_1zw_2w_1zw_2$, if $z$ is a constant, by definition of a constant it holds that $w_1zw_2 \in L$, that is there exists $b^ia^2da^2b^j \in L$, for $i, j > 0$. This fact leads to a contradiction as each word in $L$ must
contain two \(d\) symbols of the alphabet. Consequently a constant of \(L\) must be a factor of \(L\), but not of \(L_1\) and \(b^+aL_2\). Thus, each constant \(z\) of \(L\) must be of the form \(z_1ab^iaz_2\), with \(i > 0\), for \(z_1, z_2 \in A^*\). Indeed, each factor \(ab^ia\) is a constant by Proposition 2.1 as it is a singular word and thus every word having \(ab^ia\) as a factor is also a constant, by a known property proved in [6] and in [10]. But, for each \(i > 0\), there exists an infinite set of words in \(L\) that do not have \(z\) as a factor, thus implying that there exists no finite set \(M\) of constants of \(L\) such that \(L = \bigcup_{m \in M} L(m) \cup X\), where \(X\) is a finite subset of \(A^*\).

4. Reflexive symmetric splicing languages

In this section, we illustrate the characterization of reflexive symmetric splicing languages given in [6] and show that this result extends to all reflexive splicing languages. Moreover, we relate this result to a decision algorithm proposed in [12] for reflexive splicing languages. Again, the notion of constant is fundamental in giving a structural description of regular splicing languages. Indeed, given \(L\), a reflexive symmetric splicing language, then \(L\) is characterized in terms of a finite set \(M\) of constants for language \(L\). More precisely, \(L\) is defined in finite terms as a finite union of languages obtained by one single iteration of a splicing operation.

The first intermediate significant result relating splicing languages to constants has been proved for symmetric and reflexive languages in [6]: it states that splicing sites of rules of a symmetric and reflexive splicing language \(L\) are constants for the language. Actually, we can improve this result by showing in Proposition 4.1 that reflexivity is a necessary and sufficient condition for a splicing language to satisfy the above stated property (an independent proof of this Lemma is given in [12]).

**Lemma 4.1.** Let \(L\) be a language and \(r = ((u_1, u_2)\$\$)(u_1, u_2))\) a rule. Then \(L\) is closed under rule \(r\) iff \(u_1u_2\) is a constant for \(L\).

**Proof.** Let \(L\) be closed under rule \(r\). Let \(w_1 = xu_1u_2y \in L\) and \(w_2 = zu_1u_2v \in L\). Since \(r\) is applied to \(w_1, w_2\) in different order, then it is immediate that \(xu_1u_2v \in L\) and \(zu_1u_2y \in L\). Consequently, \(u_1u_2\) is a constant for \(L\).

Vice versa, assume that \(u_1u_2\) is a constant for \(L\). By definition of constant, given \(xu_1u_2y \in L\) and \(zu_1u_2v \in L\), then \(xu_1u_2v \in L\) and \(zu_1u_2y \in L\), thus implying that \(L\) is closed under \(r\). \(\blacksquare\)

Thus we state the first characterization theorem for reflexive splicing languages.

**Proposition 4.1.** Let \(L\) be a regular language. Then \(L\) is a reflexive splicing language iff there exists a splicing system \(S = (A, I, R)\) such that \(L(S) = L\) and the sites of rules in \(R\) are constants for \(L\).

**Proof.** If \(L\) is reflexive, then there exists a system \(S = (A, I, R)\), where \(R\) is a reflexive set of rules and \(L = L(S)\). Then for each pair \(s_{ij} = (u_i, u_j)\) such that \((s_{ij}\$\$)\) or \((s\$s_{ij})\) is a rule in \(R\), there exists the rule \((s_{ij}\$\$)\) in \(R\). By Lemma 4.1, \(u_iu_j\) is a constant for \(L\). Vice
versa, if each site $u_iu_j$ of a rule $r$ is a constant, rule $(s_{ij}s_{ij})$ may be added to $R$ as by Lemma 4.1, $L$ is closed under rule $r$. This fact implies that there exists a splicing system $S = (A, I, R')$, with a reflexive set of rules $R'$ such that $L = L(S)$ and thus $L$ is reflexive. □

Given $\mathcal{M}$ a finite set of constants for language $L$, we define the set $F(\mathcal{M})$ of 2-factors of words in $\mathcal{M}$ (a 2-factor in $F(\mathcal{M})$ is named split of a constant in [6]):

$$F(\mathcal{M}) = \{((m_{i1}, m_{i2}) : m_{i1}m_{i2} \in \mathcal{M})\}.$$ 

A binary relation over $F(\mathcal{M})$ induces a set $R_\mathcal{M}$ of rules, precisely, $R_\mathcal{M} \subseteq \{(s_1s_2) : s_1, s_2 \in F(\mathcal{M})\}$; let us call $R_\mathcal{M}$ a set of constant-based rules over $\mathcal{M}$.

A splicing operation is defined for a pair of constant languages $L(m_i)$, $L(m_j)$ by a rule $r \in R_\mathcal{M}$ if each of the constants $m_i$ and $m_j$ is a distinct site of rule $r$. Formally, given $r = ((u_1, u_2)s(u_3, u_4))$, such that $u_1u_2 = m_i$, $u_3u_4 = m_j$, and $L(m_i) = L_{i1}u_1u_2L_{i2}$, $L(m_j) = L_{j1}u_3u_4L_{j2}$, then the result of a splicing operation of $L(m_i)$, $L(m_j)$ by $r$ is the language $L_{i1}u_1u_4L_{j2}$ denoted as $\text{SPLIT}(L(m_i), L(m_j), r)$ and called split language. Clearly, a split language is obtained as $\text{cl}(L_i \cup L_j, r)$ (see Section 2).

Remark 4.1. In [6], the notion of a split language is introduced by using the 2-splicing operation. More precisely, the split language of $L(m_i)$ and $L(m_j)$ by a rule $r_{ij}$ consists of $\text{cl}_2(L(m_i) \cup L(m_j), r_{ij})$. But, by Remark 2.1, it is immediate that $\text{cl}_2(L(m_i) \cup L(m_j), r_{ij}) = \text{cl}(L(m_i) \cup L(m_j), \text{sym}([r_{ij}]))$.

By the above remark, the characterization theorem for reflexive symmetric splicing languages in [6] can be also stated as follows:

**Theorem 4.1.** Let $L$ be a regular language. Then $L$ is a reflexive symmetric splicing language if there exists a finite set $X \subset A^*$, a finite set of constants $\mathcal{M}$ for $L$, a set $R_\mathcal{M}$ of constant based rules over $\mathcal{M}$ such that is symmetric and

$$L = \bigcup_{m_i \in \mathcal{M}} L(m_i) \cup \bigcup_{r_{ij} \in R_\mathcal{M}} \text{SPLIT}(L(m_i), L(m_j)), r_{ij}) \cup X.$$ 

In [6], Theorem 4.1 is proved under the additional hypothesis that $X$ is a finite set of words such that no factor of a word in $X$ is a constant $m \in \mathcal{M}$.

Given a rule $r_{ij} \in R_\mathcal{M}$, it holds that the language $L'$ of all words in $L$ that have the splice site $m$ of $r_{ij}$ as a factor is uniquely specified by the expression $\text{Cont}_L(L, m)m\text{Cont}_R(L, m)$, i.e., $L' = L(m)$. Based on this observation, the finite union of split languages can be denoted by the closure of union of constant languages under rules in $R_\mathcal{M}$.

**Lemma 4.2.** Let $R_\mathcal{M}$ be a set of constant-based rules over $\mathcal{M}$. Then, it holds that

$$\bigcup_{r_{ij} \in R_\mathcal{M}} \text{SPLIT}(L(m_i), L(m_j)), r_{ij}) = \text{cl}\left(\bigcup_{m_i \in \mathcal{M}} L(m_i), R_\mathcal{M}\right).$$
Proof. Clearly, $\bigcup_{r_{ij} \in R_M} \text{SPLIT}(L(m_i), L(m_j)), r_{ij}) \subseteq cl(\bigcup_{m_i \in M} L(m_i), R_M)$. Now, given $r_{ij} \in R_M$ and $x, y \in \bigcup_{m_i \in M} L(m_i)$ such that $r_{ij}$ applies to $x, y$, it holds that $x \in L(m_i)$ and $y \in L(m_j)$, where $m_i, m_j$ are the two splicing sites of $r_{ij}$. Consequently, it holds that $cl(\bigcup_{m_i \in M} L(m_i), r_{ij}) \subseteq \text{SPLIT}(L(m_i), L(m_j), r_{ij})$, which concludes the proof of the Lemma. □

By using Proposition 4.1, in the following we show that Theorem 4.1 can be generalized to all reflexive (symmetric or nonsymmetric) splicing languages.

**Proposition 4.2.** Let $L$ be a regular language. The following are equivalent:

1. $L$ is a reflexive (symmetric) splicing language.
2. There exists a finite set $X \subseteq A^*$, a finite set of constants $M$ for $L$, a set $R_M$ of (symmetric) constant-based rules over $M$ and

$$L = \bigcup_{m_i \in M} L(m_i) \cup cl \left( \bigcup_{m_i \in M} L(m_i), R_M \right) \cup X.$$  \hspace{1cm} (1)

**Proof.** The proof of the implication (2) $\implies$ (1) is obtained by showing that the proof of the same implication given for Theorem 4.1 in [6] holds in general, without assuming that $R_M$ is necessarily symmetric. Thus let us now show that (1) $\implies$ (2) holds. If (1) holds then there exists a splicing system $S = (A, I, R)$ such that $L = L(S)$ and $R$ is reflexive. By Proposition 4.1 the set $M$ of sites of rules in $R$ is a finite set of constants. Thus $R_M = R$ is a set of constant base rules over $M$, and $L$ is closed under $R_M$. By this fact it holds that $L \supseteq L' = \bigcup_{m_i \in M} L(m_i) \cup cl \left( \bigcup_{m_i \in M} L(m_i), R_M \right)$. Thus $L' \cup I \subseteq L$, where $I$ is the initial language of $S$. Let us now show by induction on the length of a word $w \in L$ that $L \subseteq L' \cup I$. Clearly, if $w \in I$, then $w \in L$. Thus assume that $w \in L, w \notin I$. Then, $w \in cl(x \cup y, r)$, for $r \in R$. By induction $x, y \in L' \cup I$ and consequently $w \in cl(\bigcup_{m_i \in M} L(m_i), R_M)$, thus proving that $w \in L'$. □

**Example 4.1.** The regular language $L = a^+ba^+ba^+ \cup a^+ca^+ba^+$ is a reflexive symmetric splicing language. Indeed, given the set $M = \{c, bab\}$ of constants for $L$ and the constant languages $L_1 = a^+m_1a^+$ and $L_2 = a^+m_2a^+ba^+$, where $m_1 = bab, m_2 = c$, then $L = L_1 \cup L_2 \cup \text{SPLIT}(L_1 \cup L_2, r)$, where $r = ((b, ab)$S(ac, 1)) $\in R_M$.

The following remark has been stated in [6].

**Remark 4.2.** Given $L$ a regular language, a constant language $L(m)$ is a special case of split language, as indeed $L(m) = \text{SPLIT}(L(m), L(m), r)$, where $r = ((m, 1)$S($m, 1))$ is a constant-based rule.

Then, we obtain as a Corollary of Theorem 4.1 the following characterization of reflexive splicing languages, proved in [12].

**Corollary 1.** Let $L$ be a regular language. Then the following are equivalent:

1. $L$ is a reflexive (symmetric) splicing language.
There exists a set $R$ of reflexive (symmetric) rules such that $L$ is closed under $R$ and $L = \text{cl}(L, R) \cup X$, for $X \subset A^*$ a finite set.

**Proof.** By using Remark 4.2, Lemmas 4.2 and 4.1, we can show that statement (2) is equivalent to statement (2) of Proposition 4.2. Consequently, by a direct application of Proposition 4.2, the equality of the two statements holds. □

There exist splicing languages that are reflexive and not symmetric as stated in Lemma 4.3. Indeed, by applying a Theorem stated in [25], we can show that $L_1 = a^* + a^*ba^*$ and $L_2 = a^* \cup da^* \cup a^*c$ are not symmetric languages, while Lemma 4.3 shows that these languages are reflexive.

**Lemma 4.3.** Languages $L_1 = a^* + a^*ba^*$ and $L_2 = a^* \cup da^* \cup a^*c$ are splicing languages that are reflexive and not symmetric.

**Proof.** The language $L_1$ can be expressed as $L(b) \cup \text{cl}(L(b), R)$, where $R = \{r\}$, $r = ((1, b)s(b, 1), L(b) = a^*ba^*$ is a constant language. Similarly, the language $L_2 = L(d) \cup L(c) \cup \text{cl}(L(c) \cup L(d), r)$, where $L(d) = da^*$ and $L(c) = a^*c$ and $r = ((1, c), (d, 1))$. Then, by Proposition 4.2, $L_1$ is a reflexive splicing language. □

The existence of nonreflexive splicing languages has been proved in [12], indeed, as shown in [12], $a^*b^*a^*b^*a^* \cup a^*b^*a^*$ is an example of a symmetric, nonreflexive splicing language, while $b^*a^*b^*a^* \cup a^*b^*a^* \cup a^*$ provides an example of a splicing language that is neither symmetric nor reflexive. Language $a^*b^*a^*$ is an example of reflexive splicing language that is not in FCS, as shown in [12].

### 5. Decision algorithms for subclasses of regular splicing languages

A characterization theorem that extends the result for reflexive languages to all regular splicing languages is still unknown. Indeed, a procedure to decide whether a regular language is a splicing language is still unknown. On the other end, we still do not know how to use the characterization of Theorem 4.1 to obtain a procedure to decide whether a regular language is a reflexive splicing languages. Indeed, this question is a generalization of the problem posed in [14]: find a decision procedure for the class of FCS languages. However partial results have been achieved in [12], where it is proved that we can decide whether a regular language is reflexive.

The design of algorithms to solve decision problems for regular splicing languages and subclasses of splicing languages is a topic that is still unexplored.

In the following we list basic results that have been achieved in different papers and are strongly related to the solution of the above-mentioned questions. These results are stated below and then detailed by the Lemmas and Remarks that follow.

- A decision procedure to establish when a language $L$ is closed w.r.t. to a given set $R$ of rules (see Lemmas 5.1, 5.2 and Remark 5.1).
A standard procedure for the construction of an initial language and basic rules to generate constant generated splicing languages or marker splicing languages, given the reduced graph for the language [6,5,11].

A characterization of splice sites in terms of the syntactic congruence (see Lemma 5.3). The following Lemma has been proved in [6] for symmetric splicing systems, but it can be easily extended to the general case.

**Lemma 5.1.** Let $S = (A, I, R)$ be a splicing system, let $L \subseteq A^*$ be a regular language and let $A$ be the automaton recognizing $L$. Then $L = L(A)$ is closed with respect to a rule $r = ((u_1, u_2)S(u_3, u_4)) \in R$ if and only if for each pair $(p, q) \in Q_{u_1u_2} \times Q_{u_3u_4}$, we have

$$\text{Cont}_{r,p}(L, u_1u_2) \subseteq \text{Cont}_{r,q}(L, u_3u_4).$$

More precisely, the following result is used to prove containment relation between languages.

**Lemma 5.2 (Bonizzoni et al. [3]).** Let $S = (A, I, R)$ be a splicing system, let $L \subseteq A^*$ be a regular language such that $I \subseteq L$. If $L$ is closed under $R$, then $L \subseteq L(S)$.

**Remark 5.1.** There is an effective procedure to decide whether a language $L$ is closed under a set $R$ of rules, given $A$ the automaton for $L$. Indeed, given $w \in A^+$, and $p \in Q_w$, then $\text{Cont}_{r,p}(L, w)$ is a regular language (see definition in Section 2).

**Remark 5.2.** There is an effective procedure to build the splicing system that generates a reflexive splicing language (see [5] and [6] for a simpler construction).

The following property relates splice sites w.r.t. the syntactic congruence and has been proved in several papers [5,12].

**Lemma 5.3.** Let $L \subseteq A^*$ be a regular language that is closed under rule $r = ((u_1, u_2), (u_3, u_4))$. Then $L$ is closed under each rule $\tilde{r} = ((u_1', u_2'), (u_3', u_4'))$, where $u_i' \in [u_i]$, for $i \in \{1, 2, 3, 4\}$.

5.1. **Decision algorithms for reflexive and symmetric splicing languages**

The characterization Theorem 4.1 (and Proposition 4.2) leads to an effective algorithm to decide whether a regular language $L$ is a reflexive symmetric splicing language, whenever a bound on the size of each rule in $R$ can be given. Assume that given $L$, such a bound is specified by the value $\text{Bound}(L)$. Thus the set of rules generating $L$ consists of the larger set of constant-based rules $R_M$ over set $M$ such that $L$ is closed under $R_M$, where $M$ is the finite set of all constants of $L$ of length $n \leq \text{Bound}(L)$: by Remark 5.1 such a set has an effective construction algorithm.

Since, given two regular languages $X, Y$, it is decidable whether $X = Y$, then equation 1 of Theorem 4.1 can be tested by classical algorithms, thus it is immediate to obtain the required decision procedure. Actually, this algorithmic approach has been proposed in [12] to find a procedural application of Corollary 1. Such a procedure is based on an upper bound for $\text{Bound}(L)$ in terms of the size of the syntactic monoid for $L$. 
5.2. Decision algorithms for NCS and marker languages

A decision algorithm for marker language, based on properties of markers, is given in [5]. An almost unexplored approach to the development of decision algorithms for the classes of regular splicing languages discussed in the previous sections is based on properties of the reduced graphs recognizing such languages. An example in this direction is given in [7], where a characterization of NCS languages in terms of a property of the reduced graph automaton recognizing such languages is proposed. More precisely, in [7], using the algorithmic approach proposed in [18] to recognize locally testable languages, the graph properties that relate SLT to their reduced graphs are investigated and a graph-based algorithm to recognize SLT languages and other subclasses of regular languages is given. Recently, we discovered that similar results have been achieved independently in [1] in a different framework.

6. Conclusions and open problems

Finite splicing systems theory has revealed that there are extensive interactions between the notion of splicing operation and two classical tools in formal language theory: the constant and the syntactic congruence. Even though many theorists have moved their attention towards new models for molecular computation, we believe that the finite splicing systems theory still hides promising developments, mainly from the point of view of formal language theory as well as concerning the original motivation of finding procedures for building simple models to describe enzymatic processes.

In this paper, we have discussed the most significative progress in this theory made to understand the structure of regular splicing languages. We improve the result given in [6], by showing that the larger class of regular languages that has a structural characterization is the one of reflexive splicing language. It remains a challenging open question to drop the reflexivity assumption.

In this paper, we also discuss the most recent progress made towards the solution of two fundamental questions in this theory: the development of decision algorithms for classes of regular splicing languages, and the synthesis of splicing systems for such languages.

In this direction, some basic questions are still open and we believe that it will be fruitful for the formal language theory of splicing systems to look for their solution. Below, we just list some intriguing open questions.

• **Question 1:** Is there a nice characterization of reflexive splicing languages in terms of classes of the syntactic monoids, as for marker languages [3] or in terms of reduced graphs properties as for NCS languages?
• **Question 2:** Find a characterization of the finite set of constants that are used in Theorem 4.1.
• **Question 3:** Investigate boolean closure properties of reflexive and nonreflexive splicing languages.

We conclude this list by pointing out an intriguing conjecture proposed in [12] and mentioned in [11].

**Conjecture 1.** A splicing language must have constants.
Acknowledgements

The authors would like to thank C. De Felice and R. Zizza for long discussions on the topics covered in the paper. This work is partially supported by MIUR Project “Linguaggi Formali e Automi: teoria ed applicazioni”, by the contribution of EU Commission under the Fifth Framework Programme project MolCoNet IST-2001-32008.

References