Regular Splicing Languages, constants and synchronizing words

Paola Bonizzoni\textsuperscript{1} and Natasha Jonoska\textsuperscript{2}

\textsuperscript{1} Dipartimento di Informatica Sistemistica e Comunicazione  
Univ. degli Studi di Milano - Bicocca  
Viale Sarca 336, 20126 Milano - Italy  
bonizzoni@disco.unimib.it

\textsuperscript{2} Department of Mathematics and Statistics, University of South Florida, Tampa  
FL, USA  
jonoska@math.usf.edu

1 Extended abstract

Since its introduction [8], splicing, as an operation on words has been shown to be intimately connected to a wide range of concepts and notions in formal language theory. The motivation for the splicing operation comes from molecular biology. Certain restriction enzymes recognize a given sequence of nucleotides in the DNA and cleave the molecule leaving short overhang. Then molecules with complementary overhangs can be ligated and produce cross-over in the molecules. In the original paper [8], T. Head asked the question about the generative power of the splicing operation. There, he pointed out a connection between splicing and a well known notion of a constant in a language (introduced by Schützenberger [14]) and proved that when the overhangs left by the enzymes are constants, then the set of molecules generated by a finite set of enzymes represents a strictly locally testable language [1].

Formally, a splicing system (or H-system) is a triple $H = (A, I, R)$, where $A$ is a finite alphabet, $I \subseteq A^*$ is the initial language and $R$ is the set of rules, $R \subseteq (A^*)^4$. Splicing by a rule $r \in R$, denoted $r = (u_1|u_2, u_3|u_4)$ is obtained when $r$ is applied to two words $x_1u_1u_2x_2$ and $y_1u_3u_4y_2$ producing words $x_1u_1u_4y_2$ and $y_1u_3u_2x_2$. The formal language generated by the splicing system is the smallest language containing $I$ and closed under all possible splicing defined by $R$. This definition, reformulated by G. Paun, is nowadays a standard [11].

Theoretical results in splicing systems theory have contributed to the growth of a new research field in formal language theory focused on the modeling of biochemical processes and systems [12].

It was proven early on [5], and later in several other papers by different approaches (see for example [13], [15]) that given a finite initial set of words and a finite set of rules, the language generated by the splicing system is regular. However, there are simple examples of regular languages (for example $(aa)^*$) that cannot be generated by any splicing system. Unfortunately, characterization of the class of regular splicing languages remains elusive.
There have been successes in characterizing certain subclasses of splicing languages, for example those generated by reflexive rules (if \((u_1|u_2, u_3|u_4) \in R\) the both \((u_1|u_2, u_1|u_2)\) and \((u_3|u_4, u_3|u_4)\) are in \(R\)) and those generated by symmetric rules (if \((u_1|u_2, u_3|u_4) \in R\) then \((u_3|u_4, u_1|u_2)\) is in \(R\)). Reflexivity and symmetry are natural properties for splicing systems as originally defined in [9]. In [10] an example of regular splicing language that is neither reflexive nor symmetric is provided, and it has been proven that it is decidable whether a given regular language is a reflexive splicing language. A quite different characterization of reflexive symmetric splicing languages is given in [3] and it has been extended to the general class of reflexive regular languages in [2]. Not surprisingly, in this characterization the notion of constant plays an essential role.

It seems that the notion of a constant may play vital role in solving the open problem of characterizing of the whole class of regular splicing languages. Indeed, it has been conjectured as folklore officially in [6] and in [10], that a necessary condition for a regular language to be splicing is that it must have a constant. Recently, we solved this question by proving that the conjecture is true [4]. In this abstract we briefly sketch the main steps of the proof and point out the strict connection with synchronizing words and properties of the minimal finite state automaton for a regular splicing language.

1.1 The main result

We prove the following Proposition.

**Proposition 1 (Main result).** If \(L\) is a regular Paun splicing language, then \(L\) has a constant.

The main steps that lead to the proof of Proposition 1 are listed below.

Given a regular language \(L \subseteq A^*\), in we consider the minimal deterministic finite state automaton (mDFA) \(A\) that recognizes \(L\). Moreover we assume that the mDFA is trimmed, that is, every state in \(A\) is accessible and co-accessible. Some properties of the graph representing the mDFA, (unique up to a possible renaming of the states, i.e., up to an isomorphism) are investigated with respect to the notion of synchronizing word and a constant. A word \(w\) is synchronizing for a state \(q\) in \(A\) if every path in \(A\) with label \(w\) ends at \(q\). A word \(w\) is constant for the language \(L\) if for all \(x_1, x_2, x_3, x_4 \in A^*\), whenever \(x_1wx_2\) and \(x_3wx_4\) are in \(L\) we have that \(x_1wx_4\) and \(x_3wx_2\) are also in \(L\).

A relationship between of constants and synchronizing words, which is more or less folklore is stated below.

**Proposition 2.** Let \(L \subseteq A^*\) be a regular language and let \(A\) be the (trimmed) mDFA recognizing \(L\). A word \(w \in A^*\) is a constant for \(L\) if and only if \(Q_w\) is a singleton, i.e., there is a unique state \(q_w\) such that \(\delta(q, w) = q_w\) for all \(q \in Q\).

By the Proposition 2 in a trimmed mDFA \(A\) or a regular language \(L\), the set of synchronizing words for \(A\) coincides with the set of constants of \(L\).
The proof of the main result uses the notion of a transitive component \( C \) of the mDFA for a language \( L \): this is defined as a strongly connected component of the directed graph for a deterministic automaton that recognizes \( L \). If in a transitive component, every edge that starts at a state in this component also ends at the same component, then the transitive component is called terminal. For every state in the mDFA for a language \( L \) there is a path that leads from that state to a terminal component.

Thus the following observation is also used to prove the main result.

**Proposition 3.** Let \( L \) be a regular language, \( x \) a factor of language \( L \) and \( A \) be an mDFA for \( L \). At least one of the two cases holds:

(i) \( x \) is a factor of a constant for \( L \).
(ii) there is a path automaton with a path labeled \( x \) that ends in a non-trivial terminal transitive component with two non-zero states.

We sketch the main steps used in the proof. Let \( L \) be a regular splicing language and \( A \) the mDFA for \( L \).

- **Step 1:**
  Assuming that \( L \) has no constant, then Proposition 3 applies. The mDFA for \( L \), denoted \( A \), has non-trivial terminal components. We consider among all terminal components in a mDFA the component \( C \) and a state \( q \) in \( C \) such that the language recognized by the automaton \( C_q \) induced by the component \( C \) having the initial state \( q \) is minimal among all terminal components that are factor-equivalent to \( C \). In other words \( L(C_q) \) is minimal among all languages of terminal components that have the same set of labels of paths as \( C \).

- **Step 2:**
  Given such chosen \( C \), then we consider the generation by splicing rules of words in \( L \) of the form \( wc^\ast x \), where \( w \) is label of a path in the mDFA from the initial state to the component \( C \). The word \( c \) is a specially chosen word, it is a label of a path that starts and ends at the same state and visits all other states in \( C \).

  We prove the existence of a rule \( r = (u_1|u_2, u_3|u_4) \) such that \( c^i \) for arbitrarily large \( i \)'s appears in the right-context \( R(u_3u_4) \) of word \( u_3u_4 \), being \( R(u_3u_4) = \{ z : xu_3u_4z \in L, x, z \in A^\ast \} \).

- **Step 3:**
  We show that every state \( \bar{q} \) in a terminal component \( \bar{C} \) reached by reading \( u_3u_4 \) in automaton \( A \), must be the same state \( p \) that is reached by reading \( vu_1u_4 \) along the path labeled by \( wc^\ast x \), for some word \( v \in A^\ast \). This step shows that \( u_3u_4 \) is synchronizing for state \( p \), that is \( u_3u_4 \) must be constant for \( L \).

  Observe that the proof that state \( \bar{q} \) reached by reading \( u_3u_4 \) in automaton \( A \) is unique depends on the following fact: the component \( \bar{C} \) having state \( \bar{q} \) is terminal and due to the minimality of \( C_q \), \( \bar{C} \) must be the same component as \( C \). This fact also uses the property of splicing rules ( \( R(u_3u_4) \subseteq R(u_1u_4) \)) and the minimality of mDFA.
Acknowledgment PB is partially supported by MIUR Project “Mathematical aspects and emerging applications of automata and formal languages”, NJ is supported in part by NSF grants CCF-0726396 and DMS-0900671. Parts of this research were done during NJ’s visit to University of Milano -Bicocca in the Spring of 2008.

References

4. P. Bonizzoni, N. Ionoska, Regular languages must have a constant. Manuscript.