Information and Computation 242 (2015) 340–353

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Existence of constants in regular splicing languages

Paola Bonizzoni [a,*], Nataša Jonoska [b]

[a] Dipartimento di Informatica Sistemistica e Comunicazione, Univ. degli Studi di Milano-Bicocca, Viale Sarca 336, 20126 Milano, Italy
[b] Department of Mathematics and Statistics, University of South Florida, Tampa, FL, USA

ARTICLE INFO

Article history:
Received 7 June 2012
Received in revised form 14 October 2014
Available online 9 April 2015

ABSTRACT

In spite of wide investigations of finite splicing systems in formal language theory, basic questions, such as their characterization, remain unsolved. It has been conjectured that a necessary condition for a regular language L to be a splicing language is that L must have a constant in the Schützenberger sense. We prove this longstanding conjecture to be true. The result is based on properties of strongly connected components of the minimal deterministic finite state automaton for a regular splicing language. Using constants of the corresponding languages, we also provide properties of transitive automata and path-automata.

1. Introduction

A splicing system, originally introduced in [12], is a formal model that uses contextual cross-over operation over words to generate languages called *splicing languages*. This cross-over splicing formalizes the behavior of basic biomolecular processes involving cut and paste of DNA performed by restriction enzymes and a ligase. Restriction enzymes act on double stranded DNA molecules by cleaving certain recognized segments leaving short single stranded overhangs. Molecules with same overhangs can join (in a cross-over fashion) in presence of a ligase enzyme. In the introductory paper, T. Head proved that if the splicing is performed by a finite set of certain simple rules, then splicing of finite set of words can generate the class of strictly locally testable languages [9]. The splicing notion was reformulated by G. Paun at a less restrictive level of generality, giving rise to the splicing operation that is commonly adopted and appears nowadays as a standard [17].

Theoretical results in splicing systems have contributed to new research in formal language theory focused on modeling of biochemical processes [18]. On the other side, the field suggested new ideas in the framework of biomolecular science, for example, the design of automated enzymatic processes.

In this paper, we focus on finite splicing systems, called here simply as splicing systems. A splicing system is meant to have a finite set of rules (modeling enzymes) applied on a finite set of initial strings (modeling DNA sequences). 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, (see Section 4 for the definitions). The formal language generated by the splicing system is the smallest language containing I and closed under the splicing operation.

There have been successes in characterizing certain subclasses of splicing languages, for example those generated by reflexive rules and those generated by symmetric rules [2]. Reflexivity and symmetry are natural properties for splicing systems because they assure splicing of molecules cut with the same enzyme, as well as recombining molecules resulting of

* Corresponding author.
E-mail addresses: bonizzoni@disco.unimib.it (P. Bonizzoni), jonoska@mail.usf.edu (N. Jonoska).

http://dx.doi.org/10.1016/j.ic.2015.04.001
0890-5401/

the same type of cut [12]. The formal language of a general splicing system may have a set of rules R that is not necessarily symmetric, nor reflexive. Under the formal model, a splicing system is a generative mechanism for a language which belongs to a class that is a proper subclass of the regular languages. This basic result has been firstly proved in [8], and later proved in several other papers by using different approaches (see for example [19,21]).

In spite of the vast literature on the topic, a structural characterization of the finite splicing systems is still an open problem, although decidability of regular splicing languages has been recently proved in [15].

On the other hand, progress has been made towards the characterization of certain sub-classes of splicing systems. Authors in [11] prove that it is decidable whether a regular language is a reflexive splicing language and provide an example of a regular splicing language that is neither reflexive nor symmetric, 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 [4,5]. This characterization has been given by using the concept of a constant of a language introduced by Schützenberger [20].

In order to solve the open problem of characterizing he whole class of splicing languages, it seems necessary to understand the role of constants. Indeed, since the introduction of splicing languages it has been conjectured, and more formally in [10], and in [11], that existence of a constant is a necessary condition for a regular language to be splicing. In this paper we solve this longstanding open question by proving this conjecture true. This result is proved by investigating structural properties of connected components of the transition graph given by the minimal finite state automaton for a regular splicing language. More precisely, properties of the factor language of transitive components are related to the notion of synchronizing words [7]. Synchronizing words have been studied in automata theory for a long time and are of interest in both coding theory [1] and symbolic dynamics [16,14]. Our proof uses an old observation that a synchronizing word for an automaton is a constant for the language recognized by the automaton [20].

The paper is organized as follows.

In Section 2 we introduce preliminary concepts, including the notion of a synchronizing word and a constant. In Section 3 we introduce the notion of a transitive automaton and a path-automaton, as well as show several results connecting terminal components automata and synchronizing words. Moreover, we show a relationship between transitive languages, transitive automata, transitive components, and constants of the language. Then in Section 4 we recall the basic notion of a splicing system and revisit the notion of splicing rules of a splicing system by providing properties that are necessary in proving the main result of the paper. Finally in Section 5 we give examples of non reflexive splicing languages, show a relationship between transitive languages and splicing languages and we prove the main result of the paper. A preliminary extended abstract of this paper appeared in [6]

2. Preliminaries

We refer the reader to [13] for the background of automata theory, and assume some familiarity of the subject. Let A^* be the free monoid over a finite alphabet A and let $A^+ = A^* \setminus 1$, where 1 is the empty word. A *deterministic finite state automaton* (DFA) is a 5-tuple $\mathcal{A} = (Q, A, I, T, \mathcal{E})$, where Q is a finite set of *states*, $I \subseteq Q$ is the set of *initial* states, $T \subseteq Q$ is the set of *terminal* (final) states and $\mathcal{E} \subseteq Q \times A \times Q$, is the set of *transitions* such that for every $q \in Q$ and every $a \in A$ the set $\{q' \mid (q, a, q') \in \mathcal{E}, q \in Q, a \in A\}$ consists of at most one element. Given a deterministic finite state automaton $\mathcal{A}$, the set of transitions defines a partial action of A^* on Q. It is generated with $a : Q \to Q$ for $a \in A$ defined with $q(a) = q'$ iff $q' \in Q$ is the unique state with $(q, a, q') \in \mathcal{E}$. We use the standard notation qa to denote q'. If such q' does not exist, we write $qa = \emptyset$. Inductively, we extend the notation on words with $qwa = (qw)a$. Similarly, we write Qw for the image of the set Q under the map $w : Q \to Q$ defined with $w(q) = qw$. If qa is defined for all $q \in Q$ and $a \in A$ we say that $\mathcal{A}$ is *complete*. A deterministic finite state automaton is usually depicted as a directed graph with vertices Q and a set of directed edges $\mathcal{E}$. For an edge $e = (q, a, q')$ we say that q is its "start" state, q' is its "end" state (also refer to as an end-point) and a is its label. A word w is *accepted by an automaton* $\mathcal{A}$ if there is a path with label w that starts at an initial state and ends at a terminal state. We denote with $L(\mathcal{A})$ the language *recognized by* $\mathcal{A}$, that is, the set of all words accepted by $\mathcal{A}$ [13]. Given a regular language $L \subseteq A^*$ it is well-known that there is a unique *minimal* complete deterministic finite state automaton (mDFA) $\hat{\mathcal{A}} = (Q, A, \{q_0\}, T, \mathcal{E})$ that recognizes L such that all other complete DFA with one initial state that recognize L map homomorphically onto $\hat{\mathcal{A}}$ [13]. This automaton is unique up to possible renaming of the states, i.e., up to an isomorphism. We reserve the notation $\hat{\mathcal{A}}(L)$ to denote this automaton.

Given a language L, the language $F(L)$ is the set of all *factors* of words in L, where x is *factor* of a word w if $w = zxy$ for $z, y \in A^*$. We say L is *factor-closed* if $F(L) = L$.

The *right context of a word* $w \in A^*$ *with respect to a language* L is defined with $\mathcal{R}_L(w) = \{x \in A^* \mid wx \in L\}$. Symmetrically, the *left context* of w with respect of L is the set $\mathcal{L}_L(w) = \{x \in A^* \mid xw \in L\}$.

The *right context of a state in* $\mathcal{A}$ is $\mathcal{R}_{\mathcal{A}}(q) = \{x \in A^* \mid qx \in T\}$. An automaton $\mathcal{A}$ is said to be *reduced* if there are no two states in $\mathcal{A}$ with the same right context. Observe that the right context depends only on the terminal states in the automaton. In other words, if the initial state(s) are changed in $\mathcal{A}$ but the transitions and the set of terminal states remain, the right contexts of the states don't change. It is well-known (see for ex. [13]) that given a regular language L, there is a one-to-one correspondence between the right contexts of words with respect to L and the right contexts of the states in the minimal deterministic finite state automaton $\hat{\mathcal{A}}$ for L, i.e.,

$$q_0 w = q \text{ iff } \mathcal{R}_L(w) = \mathcal{R}_{\hat{\mathcal{A}}}(q).$$