Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 334 (2005) 71–98
Theoretical Computer Science
www.elsevier.com/locate/tcs

The structure of reflexive regular splicing languages via Schützenberger constants☆

Paola Bonizzoni[a], Clelia De Felice[b,*], Rosalba Zizza[b]

[a] *Dipartimento di Informatica Sistemistica e Comunicazione,Università degli Studi di Milano-Bicocca, Via Bicocca degli Arcimboldi 8, 20126 Milano, Italy*
[b] *Dipartimento di Informatica ed Applicazioni,Università di Salerno, Via S. Allende, I-84081 Baronissi (SA), Italy*

Received 10 March 2004; received in revised form 10 December 2004; accepted 13 December 2004

Communicated by G. Rozenberg

Abstract

The splicing operation was introduced in 1987 by Head as a mathematical model of the recombination of DNA molecules under the influence of restriction and ligases enzymes. This operation allows us to define a computing (language generating) device, called a *splicing system*. Other variants of this original definition were also proposed by Paun and Pixton respectively. The computational power of splicing systems has been thoroughly investigated. Nevertheless, an interesting problem is still open, namely the characterization of the class of regular languages generated by finite splicing systems. In this paper, we will solve the problem for a special class of finite splicing systems, termed *reflexive splicing systems*, according to each of the definitions of splicing given by Paun and Pixton. This special class of systems contains, in perticular, *finite Head splicing systems*. The notion of a *constant*, given by Schützenberger, once again intervenes.

Keywords: Automata; Regular languages; Molecular computing

☆ Partially supported by MIUR Project "*Linguaggi Formali e Automi*: *Metodi, Modelli e Applicazioni*" (2003), by the contribution of EU Commission under The Fifth Framework Programme (*project MolCoNet* IST-2001-32008) and by 60% Project "*Linguaggi Formali e Codici*: *Problemi classici e Modelli innovativi*" (University of Salerno, 2003).

* Corresponding author.
E-mail addresses: bonizzoni@disco.unimib.it (P. Bonizzoni), defelice@dia.unisa.it (C. De Felice), zizza@dia.unisa.it (R. Zizza).

0304-3975/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2004.12.033

1. Introduction

The splicing operation was introduced in 1987 by Head [12], as a mathematical model of the recombination of DNA molecules under the influence of restriction and ligases enzymes. In the meantime, the literature of the theoretical investigations of computing by splicing has impressively increased. In particular, at least three versions of the basic operation have been proposed, along with several variants of it [20].

The splicing of two DNA molecules corresponds to two phenomena. First, restriction enzymes cut the molecules, and then ligase enzymes paste together the fragments obtained, provided that they have matching sticky ends. It is quite natural to pass from the biochemical process to a word operation: molecules are considered as words and the behaviour of the enzymes (pattern recognition, the cut and paste operation) is specified by a (splicing) rule. This word operation allows us to define a computing (language generating) device, called *splicing system*. A splicing system is a triplet $S = (A, I, R)$, where A is a finite alphabet, I is a set of words over A (called initial language) and R is a set of splicing rules. The language $L(S)$ associated with S (*splicing language*) is defined as follows. We start with the words in I and we apply to each pair in $I \times I$ the splicing operation defined by the rules in R. The resulting set is joined to I and the process is iterated on this new set. The splicing language $L(S)$ is the set of all the words which can be obtained in this way (iterated splicing). The reader is referred to [20] for several variants and extensions of this definition. For instance, a standard extension, which will not be considered here, leads to a splicing language which is the set of words obtained by splicing as above and, in addition, which are in T^*, where $T \subseteq A$ is a terminal alphabet given with S (extended H system) [20].

As we have already said, there are at least three definitions of the splicing operation, given by Head, Paun and Pixton [12,14]. It is natural to compare the computational power of the corresponding devices. When we restrict ourselves to finite splicing systems (i.e., splicing systems $S = (A, I, R)$ with I and R being finite sets), we know that Pixton systems are more powerful than Paun systems, which in turn are more powerful than Head systems. The inclusions between the corresponding classes of languages are strict: there are languages which can be generated by finite Pixton splicing systems but not by finite Paun splicing systems, and there are languages which can be generated by finite Paun splicing systems but not by finite Head splicing systems [5]. Another parameter in the investigation of the computational power of splicing systems is the level in the Chomsky hierarchy I, R belong to. Here we suppose that this hierarchy contains the class of finite sets. Let us restrict ourselves to Paun's definition and let F_1, F_2 be families in the Chomsky hierarchy. We denote $H(F_1, F_2) = \{L(S) \mid S = (A, I, R)$ with $I \in F_1, R \in F_2\}$ the class of languages generated by Paun splicing systems, where the initial language I belongs to F_1 and the set of rules R belongs to F_2. Results obtained in several papers prove that either $H(F_1, F_2)$ is a specific class of languages in the Chomsky hierarchy or it is strictly intermediate between two of them [14,20]. In the latter case, a characterization of the structure of $H(F_1, F_2)$ is still lacking [14,20].

In particular, finite splicing systems generate regular languages. For Head systems this result was proved in [7], whereas the same result for Paun systems and for Pixton systems was proved by Pixton in [22] and [23] respectively.