entropy

MDPI

Article

Concatenation, Splicing and Insertion–Deletion Operations on Cyclic DNA Molecules

Bilal Abdallah [1,2,*] and Benedek Nagy [1,3,*]

[1] Department of Mathematics, Faculty of Arts and Sciences, Eastern Mediterranean University, via Mersin-10, 99450 Famagusta, North Cyprus, Türkiye
[2] College of Business Administration, American University of the Middle East, Egaila 54200, Kuwait
[3] Department of Computer Science, Institute of Mathematics and Informatics, Eszterházy Károly Catholic University, 3300 Eger, Hungary
* Correspondence: bilal.abdallah@aum.edu.kw (B.A.); benedek.nagy@emu.edu.tr (B.N.)

Abstract

Splicing systems (H-systems) and insertion-deletion systems are well-studied DNA computing paradigms inspired by biological operations observed in nature. They are modelled in a formal, information-theoretic manner. In these systems, splicing, insertion and deletion operations are used. In this paper, we study these operations on circular words, which naturally model cyclic DNA molecules, focusing on the relationship between algebraic operations and biologically inspired string transformations. In particular, we formally compare concatenation with circular splicing and insertion. While the splicing operation has already been defined and used for circular words in the literature, for insertion–deletion systems, e.g., circular insertion was defined to insert a linearized variant of a circular string (cyclic molecule) into a linear (ordinary) string (or linear molecule that can be represented by a string). In this paper, we provide a formal definition of insertion operators acting on circular words. We prove that if we use circular splicing and insertion in a somewhat context-independent way (using the empty word in the formal rules), these computational operations coincide with the one-wheel concatenation operation. Our results support the theoretical understanding of these operations and their biological interpretations, with implications for DNA computing and synthetic biology.

Keywords: circular words; splicing systems; H-systems; insertion–deletion systems; DNA operations; concatenation of necklaces; insertion–deletion to circular words; circular splicing; biological information processing

Check for updates

Academic Editor: Young-Seok Choi

Received: 26 August 2026
Revised: 8 October 2026
Accepted: 9 October 2026
Published: 12 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

In the field of molecular chemistry and biology, chemists have worked for centuries to obtain ever newer molecules which play essential roles in complex biological systems and are useful in medicine or in industrial applications [1]. Thinking about DNA molecules as important biological molecules, there are various enzymes that are able to modify them, and these enzymes can be used to build them or to make some changes to them [2,3]. A branch of computing and information theory, also related to formal language theory, is used to provide formal models of these operations. In this paper, we concentrate on modeling circular (DNA) molecules and computational operations on them.

Circular splicing is a well-explored concept in both molecular biology and theoretical computer science, primarily due to the significance of circular DNA in various biological processes and the insight splicing systems provide into the properties of circular words [4,5].

Despite extensive investigation, many classical problems related to circular splicing—such as characterizing all circular languages generated by finite circular splicing systems and understanding their computational power—remain unsolved in formal language theory [6,7].

The splicing operation was first proposed in 1987 by Tom Head [8]. Formal systems based on splicing operations are called splicing systems or H-systems. Further, he also introduced circular splicing systems in 1992 [9], along with a number of open problems concerning their computational capabilities. Then the idea was generalized by Păun [10,11] and Pixton [12] in 1996 to simulate the recombinant behavior of DNA sequences. The splicing concept is inspired by the action of two enzymes that cut DNA molecules (represented by strings here) at designated substrings, referred to as sites, and then reattach the first segment of one string to the second segment of the other, appropriately modifying the sites. From a mathematical and information-theoretical modeling perspective, we express this in the following way:

Let Σ be an arbitrary finite alphabet. A splicing rule over Σ is defined as a string of the form $u_1\#u_2\$u_3\#u_4$ where u_1, u_2, u_3, u_4 are strings over Σ while # and \$ are special symbols not contained in Σ. Intuitively, the pairs (u_1,u_2) and (u_3,u_4) model the recognition sites of two restriction enzymes; the first enzyme cuts the first string (molecule) between u_1 and u_2, while the second enzyme cuts the second between u_3 and u_4. To shorten the notation, we will represent the splicing rules by quadruples of the form $(u_1, u_2; u_3, u_4)$. For a given splicing rule $r = (u_1, u_2; u_3, u_4)$ and three strings x, y, z over Σ, we denote the relation $(x,y)\vdash_r z$ if the following conditions hold: $x = x_1u_1u_2x_2$, $y = y_1u_3u_4y_2$, $z = x_1u_1u_4y_2$ for some $x_1, x_2, y_1, y_2 \in \Sigma^*$. We may also examine the symmetric rule $r' = (u_3, u_4; u_1, u_2)$, which yields the result $w = y_1u_3u_2x_2$ where $(y,x)\vdash_{r'} w$. We will recall the circular splicing operation at the beginning of Section 3.

Similar to splicing, the insertion and deletion operations have gained attention as abstract models of bio-molecular processes. Moreover, they also have a central role in formal language theory and in formal models of DNA computing [13]. These operations reflect fundamental mechanisms such as the insertion and removal of DNA segments, and thus provide a natural basis for bio-inspired computational models [11].

The insertion and deletion operations were introduced for linear words [14]. Their circular variants were studied in [15], where a circular word is inserted into a linear word: the cyclic molecule is cut, and its linearization is placed between two contexts of a linear host molecule. Thus, in the known formulations, only the inserted molecule is cyclic, while the host, and hence also the result, is linear. What has not been defined so far is the operation in which the host molecule is cyclic as well, that is, an insertion that takes two circular words and returns a circular word, and, symmetrically, a deletion that removes a cyclic factor from a circular word. This is precisely the situation modeled in this paper. The gap is not merely notational: an operation whose result is linear cannot be iterated within a family of circular languages, nor can it be compared with circular splicing operations or the concatenation of circular words, as all of these operations act on circular words. Moreover, as we show in Section 4, applying the known operation to a linearization of a circular word and circularizing the outcome does not give the same result, because the occurrences of the context that span the position at which the circular word was cut are not exhibited by any linear representative of the molecule.

To define the insertion–deletion operations, we consider Σ as a finite alphabet, and R is a finite set of triples of the form $(u,\alpha/\beta,v)$ where $u, v \in \Sigma^*$, $(\alpha,\beta) \in (\Sigma^+ \times \{\lambda\}) \cup (\{\lambda\} \times \Sigma^+)$. The triples in R are called insertion–deletion rules, and they are used as follows: $r_{ins} = (u,\lambda/\beta,v)$ means that β can be inserted between u and v in accordance with the rewriting rule $uv \to u\beta v$, while $r_{del} = (u,\alpha/\lambda,v)$ means that α can be deleted from the (u,v) context corresponding to the rewriting rule $u\alpha v \to uv$. Thus, for $x,y \in \Sigma^*$, we denote the relations $x \vdash_{r_{ins}} y$ and

$x \vdash_{r_{del}} y$ if y can be obtained from x by using an insertion and deletion rule, respectively, as defined above. In other words, one of the following two cases holds:

- $x = x_1uvx_2$, $y = x_1u\beta vx_2$ for $x_1, x_2 \in \Sigma^*$, $\beta \in \Sigma^+$ and $r_{ins} = (u,\lambda/\beta,v) \in R$;
- $x = x_1u\alpha vx_2$, $y = x_1uvx_2$ for $x_1, x_2 \in \Sigma^*$, $\alpha \in \Sigma^+$ and $r_{del} = (u,\alpha/\lambda,v) \in R$.

Note that $(\alpha,\beta) \neq (\lambda,\lambda)$ by definition, as no insertion or deletion occurs in this case. The circular extension of the insertion and deletion operations will be defined formally at the beginning of Section 4.

The concatenation operation is the usual way to unite information represented by words: it makes a new word from the two input words by continuing the first word, after its last letter, with the first letter of the second word. However, circular words have neither first nor last letters (see the formal descriptions in the next section); thus, it is not straightforward to generalize concatenation for them. Therefore, there are various concatenation operators defined for circular words [16]. In [17], two concatenation operations were considered, namely, the one-wheel and two-wheel concatenations. Their naming is motivated by the graphical representation of a circular word as a wheel, in which its letters are arranged cyclically around a circle. Under this interpretation, one-wheel concatenation combines two circular words into a single wheel, and thus the result consists of circular words, whereas two-wheel concatenation retains the two circular words as two distinct wheels while establishing the concatenation between them, and therefore may produce more complex structures.

This paper explores the theoretical foundations of circular splicing and insertion-deletion operations used in the corresponding modeling systems, with a focus on their computational power and biological relevance. We will examine various models, their relations to the one-wheel concatenation [17] and the open questions that continue to challenge researchers in this field.

The main contributions of this paper are the following:

- We prove that Head's and Păun's circular splicing operations produce the same circular words under an explicit translation of the rules, i.e., as single operations they are equivalent in the sense that one operation can be simulated by an operation of the other type (Proposition 1);
- We prove that the one-wheel concatenation of two circular words is exactly the set obtained by all applicable Păun circular splicing rules, and also by the single rule (λ, λ; λ, λ), and the analogous result for Pixton's operation with the rule pair (λ, λ; λ) (Theorem 2, Corollary 1 and Theorem 4);
- We give if-and-only-if characterizations of the cases when the Păun and the Pixton circular splicing operations coincide with the two-wheel concatenation (Theorems 3 and 5);
- We provide a formal definition of the insertion operation acting on circular words, establish the hierarchy of the obtained sets and characterize when they coincide, and prove that this insertion can be simulated by Păun circular splicing and that, with empty contexts, it coincides with the one-wheel concatenation (Definition 7, Lemmas 2 and 3, Theorems 6–8);
- We define the deletion operation on circular words and show that it is strictly length-decreasing, hence never equivalent to a concatenation, that with empty contexts it realizes the inverse relation of the one-wheel concatenation, and that it can be simulated by Pixton circular splicing but not by the Head or the Păun one (Definition 8, Theorems 9 and 10).

These results concern single applications of the operations, thus the open problems on finite circular splicing *systems* mentioned above are not solved here and remain open,

namely, the characterization of the circular languages, in particular the regular ones, generated by such systems, which is known only for restricted classes [4,6,7,18], and the comparison of the generative power of the systems based on the three definitions [6,19]; our Proposition 1 does not settle the latter, since the systems differ also in the structure and use of their sets of rules, however we hope that our results can be applied in further studies also in the direction to solve open problems about splicing systems.

The paper is organized as follows. Section 2 fixes the notations and recalls the necessary notions on words and circular words. Section 3 recalls the circular splicing operation in the definitions of Head, Păun and Pixton, compares these definitions with each other, and relates them to the one-wheel and to the two-wheel concatenations of circular words. Section 4 is devoted to the insertion and deletion operations: the circular insertion into circular words is defined and compared in Section 4.1 with the known circular insertion into linear words, with the Păun circular splicing and with the one-wheel concatenation, while Section 4.2 deals with the circular deletion. Section 5 discusses the implications of our results for systems generating circular languages. Finally, Section 6 summarizes the results and indicates directions for further research.

2. Preliminaries

The topic of the paper belongs to DNA computing, bioinformatics and models of DNA-based computing and information systems. In this section, we briefly recall some necessary preliminaries, and we also fix our notations. For any unexplained concepts, the reader is referred to [11,20,21].

Let Σ denote the alphabet, a finite non-empty set of symbols (or letters). A string (also called a word) is a finite sequence of letters. In this paper, the term 'linear word' is also used to contrast these words with the main subject of this paper, namely the circular or cyclic words, which are defined below. Further, as usual, Σ^* denotes the set of all words over the alphabet Σ. Let $\mathbb{N}$ and $\mathbb{N}_0$ denote the set of positive and non-negative integers, respectively. The length of a string $x \in \Sigma^*$ is denoted by $|x|$ and it is the length of the sequence, i.e., the number of symbols in x. Let λ denote the empty word; it is the only word having zero length. The j^{th} letter of $x \in \Sigma^*$ is denoted by x_j, for $j \in \{1, \ldots, |x|\}$. A word $w \in \Sigma^*$ is a factor of $x \in \Sigma^*$, written as $w \in Fact(x)$, if there are two words $u, v \in \Sigma^*$ such that $x = uwv$. Moreover, if $u = \lambda$ (resp. $v = \lambda$), then w is also a prefix (resp. suffix) of x. For any non-negative integer and word x, the power of x is defined as follows: $x^0 = \lambda$, further $x^{k+1} = xx^k$ for $k \geq 0$. Two words y and x are called conjugates if there are words $v,u \in \Sigma^*$ such that $y = vu$ and $x = uv$.

A circular word $x_\circ$ can be viewed graphically as the result of joining the first symbol of a linear word x to its last, thereby forming a closed loop. Circular words are also referred to as cyclic words and necklaces in the scientific literature. Unlike a linear word, a circular word has neither a distinguished beginning nor an end. Formally, a circular word associated with x is defined as the set of all conjugates (or, equivalently, all cyclic shifts) of x. This set-based representation is commonly referred to as the linearization of the circular word $x_\circ$.

$$x_\circ = \{ y \mid y \text{ is a conjugate of } x\} \tag{1}$$

For a circular word $x_\circ = (x_1 \ldots x_n)_\circ$, we define $Fact(x_\circ)$ to be the set that contains all substrings occurring in any conjugate of x (including the empty word). A circular word of length zero is called the empty circular word, and we may also write it as $\lambda_\circ$. Further, let $|x_\circ|_a$ be the number of occurrences of a in $x_\circ$, where $a \in \Sigma$, and it is defined to be the same as the number of occurrences of a in x (or in any other element of the set $x_\circ$), i.e., $|x_\circ|_a = |x|_a$. Although a circular word is formally represented as a set, all of its constituent words have

the same length; hence, the notation $|x_\circ|$ is well-defined for their length. Conversely, the set $x_\circ$ itself has a cardinality denoted by $\|x_\circ\|$, which does not exceed $|x_\circ|$.

In the next sections, we also formally recall the splicing and insertion–deletion operations known in the literature for circular words.

3. Splicing Operation and Circular Splicing

Since circular splicing systems depend on the splicing rules used, various definitions are possible for them; however, some of them look very unnatural. As previously stated in [4,6,18], Head, Păun, and Pixton have provided three different definitions of the circular splicing operation. Generally, the main difference between the definitions of circular splicing lies in the way the splicing rules work.

Definition 1 (Head's Definition). *Let* Σ *be a finite alphabet, and* (u_1, x, u_2), (u_3, x, u_4) *be two triples of words in* Σ^* *sharing the same middle word* x. *Given two circular words* $(z_1)_\circ = (hu_1xu_2)_\circ$ *and* $(z_2)_\circ = (ku_3xu_4)_\circ$, *Head's circular splicing operation produces* $(hu_1xu_4ku_3xu_2)_\circ$.

Although a splicing operation is applied prior to concatenation, the resulting word is identical to that obtained by concatenating alternative linearizations of the original circular words, namely u_2hu_1x and u_4ku_3x, respectively (see Figure 1).

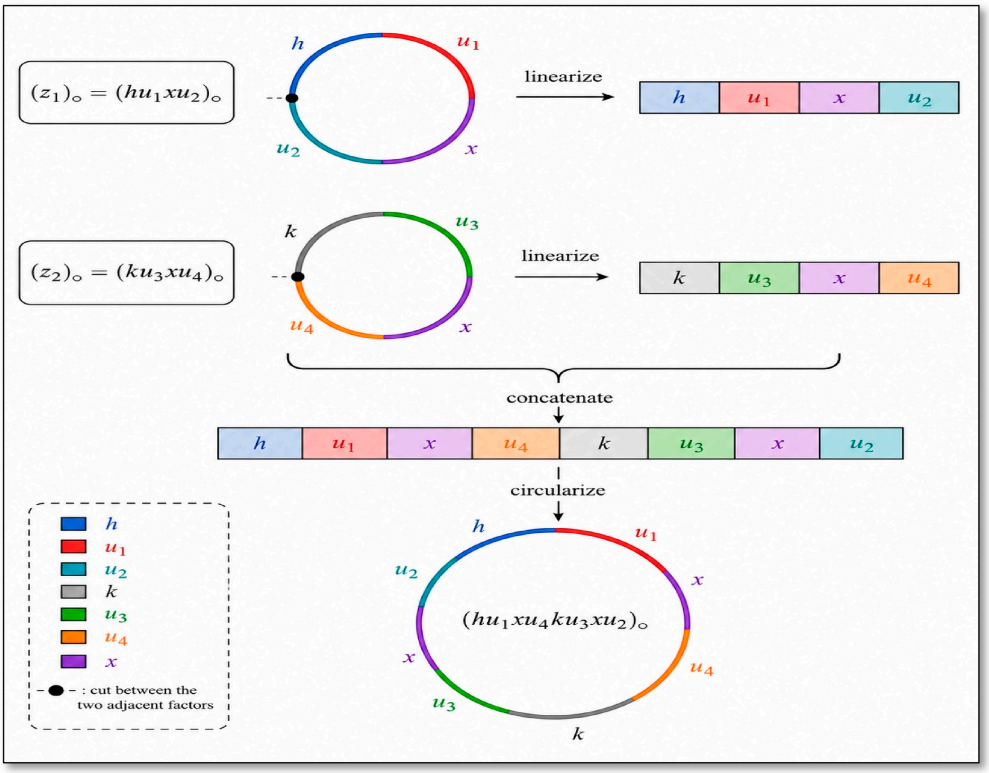


Figure 1. The Head Circular Splicing of Two Circular Words.

As we shall see, applying a splicing operation to two circular words produces a circular word. However, the operation need not be applicable to every pair of input circular words; in such cases, no output circular word can be generated (obtained). Moreover, it may also happen that the input circular words have many occurrences of the subwords specified by the rule, and therefore the rule can be applied in more than one way, and thus it may be able to produce more than one possible result.

In our examples below, generally, the computations follow the same three steps prescribed by the corresponding definition (and also shown in the corresponding figure).

(i) *Cut*, i.e., determine the linearizations of the two input circular words that have the form required by the rule;
(ii) *Concatenate* the obtained linear words;
(iii) *Circularize* the result.

Every possible choice in step (*i*) gives a result. Different choices may give the same circular word in step (*iii*). We display these steps in detail for Examples 1, 3 and 5; the other examples are computed analogously.

Example 1. *Given the two triples of words (a, λ, b) and (b, λ, a), Head's splicing operation applied to* $\left(a^2b^2\right)_\circ$ *and* $(ab)_\circ$ *generates the circular word* $\left(a^3b^3\right)_\circ$*, and two copies of* $(ab)_\circ$ *produce* $\left(a^2b^2\right)_\circ$*. However, using the same triple of words, Head's splicing operation is not applicable to* $\left(a^2b^2\right)_\circ$ *and* $(aa)_\circ$ *as the factor b is missing; hence, it generates the empty set. In addition, Head's splicing operation applied to* $(abaab)_\circ$ *and* $(aba)_\circ$ *produces two different circular words* $(aaabbaab)_\circ$ *and* $(abaaaabb)_\circ$*.*

For instance, in the first case of Example 1, the triples (a, λ, b) and (b, λ, a) require linearizations of the form $hu_1xu_2 = hab$ *and* $ku_3xu_4 = kba$*. The only conjugate of aabb ending with ab is baab (so h = ba), and the only conjugate of ab ending with ba is ba (so k = λ). By Definition 1, the result is* $(hu_1xu_4ku_3xu_2)_\circ = (ba \cdot a \cdot a \cdot b \cdot b)_\circ = (baaabb)_\circ = \left(a^3b^3\right)_\circ$.

Definition 2 (Păun's Definition). *Let* Σ *be a finite alphabet, and let* $r = (u_1, u_2; u_3, u_4) \in \Sigma^* \times \Sigma^* \times \Sigma^* \times \Sigma^*$ *be a splicing rule. Given two circular words* $(z_1)_\circ = (u_2hu_1)_\circ$ *and* $(z_2)_\circ = (u_4ku_3)_\circ$*, the Păun circular splicing operation defined by r cuts the two circular words at the specified contexts, linearizes them to the words* u_2hu_1 *and* u_4ku_3*, concatenates the obtained linear words, and circularizes the result, producing the circular word* $(u_2hu_1u_4ku_3)_\circ$*. We denote this operation by* $((z_1)_\circ, (z_2)_\circ) \vdash_r (u_2hu_1u_4ku_3)_\circ$.

Example 2. *The Păun circular splicing operation is defined by the splicing rule r = (a,b;b,a), and applied to* $\left(a^2b^2\right)_\circ$ *and* $(ab)_\circ$ *generates the circular word* $\left(a^3b^3\right)_\circ$*. It produces* $\left(a^2b^2\right)_\circ$ *when applied to two copies of* $(ab)_\circ$*.*

Example 3. *Given the two triples of words* $(u_1,x,u_2) = (aa,ab,aa)$ *and* $(u_3,x,u_4) = (bb,ab,bb)$*, Head's splicing operation applied to* $(aaaaab)_\circ$ *and* $(bbbbba)_\circ$ *generates the circular word* $(aaabbbbbabaa)_\circ$*. However, using the same components for Păun circular splicing rule* $r = (u_1, u_2; u_3, u_4) = (aa,aa;bb,bb)$*, the Păun splicing operation applied to the same circular words produces four different circular words* $(aabaaabbabbb)_\circ$*,* $(aabaaabbbabb)_\circ$*,* $(aaabaabbabbb)_\circ$ *and* $(aaabaabbbabb)_\circ$*.*

The computation is as follows. For Head's operation, the triples (aa,ab,aa) and (bb,ab,bb) require linearizations of the form h· aa· ab· aa and k· bb· ab· bb. Since these words already have length 6, we get h = k = λ, and the unique suitable conjugates are aaabaa$\in (aaaaab)_\circ$ *and bbabbb*$\in (bbbbba)_\circ$*. By Definition 1, the result is*

$$(hu_1xu_4ku_3xu_2)_\circ = (aa \cdot ab \cdot bb \cdot bb \cdot ab \cdot aa)_\circ = (aaabbbbbabaa)_\circ.$$

For the Păun rule r = (aa, aa; bb, bb), by Definition 2 *the linearizations must have the form* $u_2hu_1 = aa \cdot h \cdot aa$ *and* $u_4ku_3 = bb \cdot k \cdot bb$ *(refer to* Figure 2*). Among the six conjugates of aaaaab, exactly two begin and end with aa, namely aaabaa (h = ab) and aabaaa (h = ba). Similarly, there are two options for linearizing the 6-letter long circular word* $(bbbbba)_\circ$ *with prefix bb and suffix bb,* i.e., *bbabbb (k = ab) and bbbabb (k = ba). Combining them in the four possible ways gives the four distinct results:*

Linearization of $x_\circ$	Linearization of $y_\circ$	Concatenation	Resulting Circular Word
aaabaa	*bbabbb*	*aaabaabbbabbb*	$(aaabaabbbabbb)_\circ$
aaabaa	*bbbabb*	*aaabaabbbbabb*	$(aaabaabbbbabb)_\circ$
aabaaa	*bbabbb*	*aabaaabbbabbb*	$(aabaaabbbabbb)_\circ$
aabaaa	*bbbabb*	*aabaaabbbbabb*	$(aabaaabbbbabb)_\circ$

These four circular words are pairwise different, since none of the four linear words in the third column is a conjugate of another one. Notice that the difference in the results comes from the fact that in Head's operation an additional parameter x = ab was used that has not appeared in the applied Păun splicing operation.

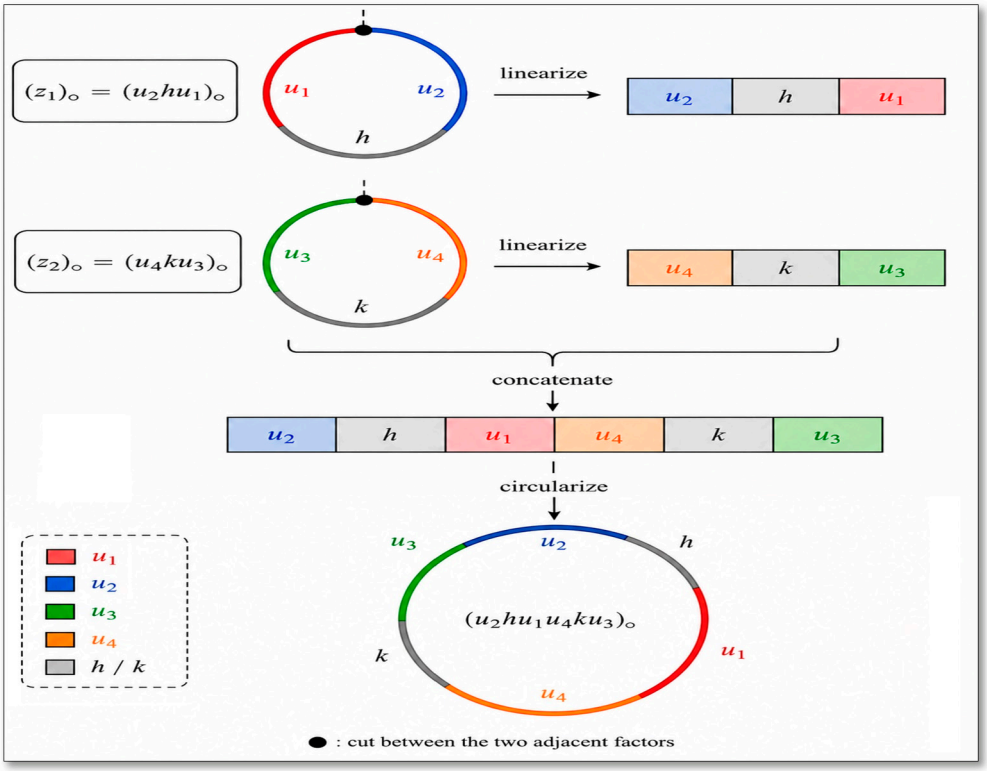


Figure 2. The Păun Circular Splicing of Two Circular Words.

From Examples 1 and 2, we observe that the definitions of Head and Păun yield the same resulting circular word when their respective splicing operations are applied to the circular words of these particular examples, but not for Example 3. However, the Head triples and the Păun rule in Example 3 do not correspond to each other as rules; Proposition 1 below shows that, under the appropriate translation of rules, the two operations yield the same results. Observe that although the splicing on linear words is applied in Head's definition, it is applied in both ways, i.e., the splicing with its symmetric pair, and the concatenation of these two resulting strings yields a circular word that has as a subword a conjugate of both initial circular words, namely, xu_2hu_1 and xu_4ku_3, where the subword occurrence of x with the other subsequences written in the rules specify the positions where the cycles are cut before concatenating the result. By the Păun-type circular splicing, the rules also specify only how to cut the original circular words into linear words to obtain the new circular word after concatenating these two linear words, and thus, in the resulting circular word, a conjugate of each of the original words must appear. In Păun's definition,

this concatenation is more direct. In [6,19], these operations and computing systems based on them are compared. Now we complement the results shown there as follows:

Proposition 1. *Let* Σ *be a finite alphabet, and let* (u_1,x,u_2) *and* (u_3,x,u_4) *be two triples of words in* Σ^* *sharing the same middle word* x*. Then, for any two circular words* $(z_1)_\circ$ *and* $(z_2)_\circ$*, Head's circular splicing operation with these triples and the Păun circular splicing operation with the rule* $r = (u_1x, u_2; u_3x, u_4)$ *produce the same set of circular words. In particular, if* $x = \lambda$*, then every Păun rule* $(u_1, u_2; u_3, u_4)$ *is reproduced by the Head triples* (u_1, λ, u_2) *and* (u_3, λ, u_4) *and vice versa.*

Proof. Applying the Head splicing operation with triples (u_1,x,u_2) and (u_3,x,u_4) to the circular words $(z_1)_\circ = (hu_1xu_2)_\circ$ and $(z_2)_\circ = (ku_3xu_4)_\circ$, by Definition 1, the resulting circular word is $(hu_1xu_4ku_3xu_2)_\circ$.

On the other hand, the Păun splicing rule $r = (u_1x,u_2;u_3x,u_4)$ applying it to $(z_1)_\circ$ and $(z_2)_\circ$, by Definition 2, produces $(u_2hu_1xu_4ku_3x)_\circ$. Since $hu_1xu_4ku_3xu_2$ is a conjugate of $u_2hu_1xu_4ku_3x$, then $(hu_1xu_4ku_3xu_2)_\circ = (u_2hu_1xu_4ku_3x)_\circ$.

Setting $x = \lambda$ gives the second statement. □

We just proved in Proposition 1 that any circular word producible by a Head splicing can also be produced by a corresponding Păun splicing and vice versa.

For instance, based on the previous result, we complement Example 3.

Example 4. *The Păun rule corresponding to the Head triples of Example 3 is* (*aaab*,*aa*;*bbab*,*bb*)*, which produces from* $(aaaaab)_\circ$ *and* $(bbbbba)_\circ$ *exactly the circular word* $(aaabbbbbabaa)_\circ$*; conversely, the Păun rule* (*aa*,*aa*;*bb*,*bb*) *of Example 3 corresponds to the Head triples* (*aa*,λ,*aa*) *and* (*bb*,λ,*bb*)*, which yield the same four circular words we have described there.*

Differences between the two approaches arise only at the level of circular splicing systems (e.g., the structure and symmetry of the sets of rules, or the self-splicing included in Păun's definition); see [6,19].

Example 5. *Consider a Păun rule with all contexts empty,* $r = (\lambda,\lambda;\lambda,\lambda)$*. Splicing the two circular single words* $(a)_\circ$ *and* $(b)_\circ$ *results in* $(ab)_\circ$*. Also, the Head triple pair* (a,x,λ) *and* (b,x,λ) *produces this from* $(a)_\circ$ *and* $(b)_\circ$*, with* $x = \lambda$*.*

It is important to remember that the splicing rules can be applied in two different ways according to Păun's initial definition of the circular splicing operation as in [22] (one method has already been explained here, and the other, known as self-splicing, will not be discussed in this paper). Note that circular splicing is defined as length-increasing in both Head's and Păun's definitions; this means that when two circular words are spliced from one another, the spliced word's length is equal to the sum of the lengths of the two words.

Pixton's definition does not necessarily satisfy the same fact, as he defined the circular splicing system differently, allowing some rewriting, as follows:

Definition 3 (Pixton's Definition). *Let* Σ *be a finite alphabet, and let* $r = (\alpha,\alpha';\beta) \in \Sigma^* \times \Sigma^* \times \Sigma^*$ *be a splicing rule. Assume there exists a partner rule* $\bar{r} = (\alpha',\alpha;\beta\prime) \in \Sigma^* \times \Sigma^* \times \Sigma^*$ *associated with r. Given two circular words* $(z_1)_\circ = (\alpha\varepsilon)_\circ$ *and* $(z_2)_\circ = (\alpha\prime\varepsilon\prime)_\circ$*, the Pixton circular splicing operation defined by the rule pair* $(r, \bar{r})$ *cuts the two circular words at the specified contexts, linearizes them to the words* $\varepsilon\alpha$ *and* $\varepsilon'\alpha'$*, substitutes the suffixes* α,α' *by* β, β'*, concatenates the resulting linear words, and circularizes the result, producing the circular word* $(\varepsilon\beta\varepsilon'\beta')_\circ$*. We denote this operation by* $((z_1)_\circ,(z_2)_\circ) \vdash_{(r,\bar{r})} (\varepsilon\beta\varepsilon\prime\beta\prime)_\circ$.

As Pixton's definition allows some rewriting, step (*ii*) in the computation in our examples is modified accordingly: instead of a simple concatenation of the linearized words, one needs to replace the marked suffixes by the ones given in the rule (see Figure 3). We show an example below:

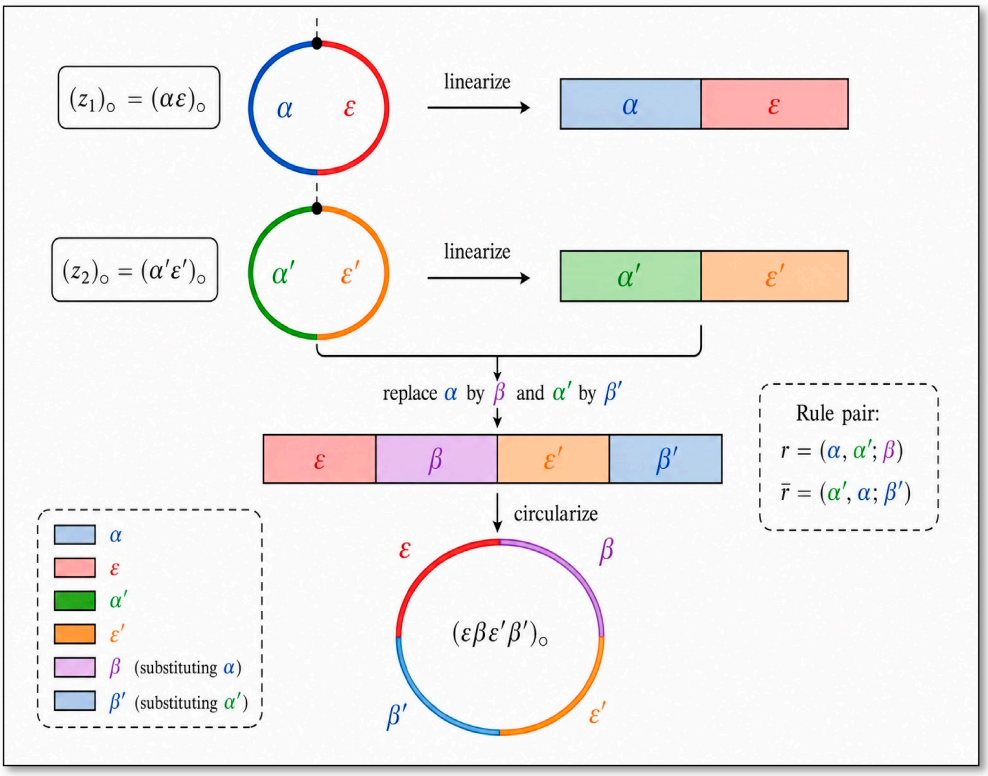


Figure 3. The Pixton Circular Splicing of Two Circular Words.

Example 6. *The Pixton circular splicing operation defined by the splicing rules* $r = (a^2, a^2b; a^2)$ *and* $\bar{r} = (a^2b, a^2; \lambda)$, *and applied to two copies of* $(a^4b)_\circ$, *generates the circular word* $(a^6b)_\circ$. *In detail, the rule r requires a linearization of the first copy of* $(a^4b)_\circ$ *of the form* $\alpha\varepsilon$ *with* $\alpha = a^2$, *giving* $\varepsilon = a^2b$, *and its partner* $\bar{r}$ *requires a linearization of the second copy of the form* $\alpha'\varepsilon'$ *with* $\alpha' = a^2$. *By Definition* 3, *the two circular words are linearized to* $\varepsilon\alpha = a^2ba^2$ *and* $\varepsilon'\alpha' = a^2a^2b$, *the suffixes* α *and* α' *are replaced by* $\beta = a^2$ *and* $\beta' = \lambda$, *respectively, and the resulting words* $\varepsilon\beta = a^2ba^2$ *and* $\varepsilon'\beta' = a^2$ *are concatenated and circularized, producing* $(a^2ba^4)_\circ = (a^6b)_\circ$.

Pixton's rules are symmetric by construction (every r has a partner $\bar{r}$), so in statements one should always mention the existence of the partner and that the pair is applied together. Moreover, for a rule $r = (\alpha,\alpha';\beta)$, there may exist distinct rules $\bar{r}_i = (\alpha', \alpha; \beta_i')$ such that the pair $(r, \bar{r}_i)$ can be applied. Nevertheless, for the sake of simplicity and since no ambiguity arises from the context, we will adopt the notation $(r, \bar{r})$ when a pair is applied.

Example 7. *Let* $r = (ab,\lambda;a^2)$ *be a Pixton splicing rule over the alphabet* $\Sigma = \{a,b\}$, *having three partners* $\bar{r}_1 = (\lambda, ab; a), \bar{r}_2 = (\lambda, ab; b)$ *and* $\bar{r}_3 = (\lambda, ab; ba)$. *Then, the Pixton circular splicing operation defined by the pair rules* $(r, \bar{r}_1), (r, \bar{r}_2), (r, \bar{r}_3)$ *and applied to* $(a^2b)_\circ$ *and* $(ab)_\circ$ *produce respectively* $(a^5b)_\circ$, $(a^4b^2)_\circ$ *and* $(a^5b^2)_\circ$.

In the next part of the section, we analyze the operations described above.

Theorem 1. *Let* Σ *be an arbitrary finite alphabet, and let* $x_\circ = (x_1 \dots x_n)_\circ$ *and* $y_\circ = (y_1 \dots y_m)_\circ$ *be two circular words, with lengths at least 2 and with indices taken modulo n and m respec-*

tively (so $x_{n+1} = x_1$ *and* $y_{m+1} = y_1$*). For fixed indices* $i \in \{1,\ldots,n\}$ *and* $j \in \{1,\ldots,m\}$*, the following nine Păun splicing rules applied to* $x_\circ$ *and* $y_\circ$ *generate the same circular word if* $|x_\circ|_{x_i} = |x_\circ|_{x_{i+1}} = |y_\circ|_{y_j} = |y_\circ|_{y_{j+1}} = 1$:

$r_1 = (x_i,x_{i+1};y_j,y_{j+1})$, $r_2 = (x_i,\lambda;y_j,y_{j+1})$, $r_3 = (\lambda,x_{i+1};y_j,y_{j+1})$,
$r_4 = (x_i,x_{i+1};y_j,\lambda)$, $r_5 = (x_i,x_{i+1};\lambda,y_{j+1})$, $r_6 = (x_i,\lambda;y_j,\lambda)$,
$r_7 = (x_i,\lambda;\lambda,y_{j+1})$. $r_8 = (\lambda,x_{i+1};y_j,\lambda)$, $r_9 = (\lambda,x_{i+1};\lambda,y_{j+1})$.

Proof. Since $n,m > 1$, the indices i and $i+1$ and j and $j+1$, respectively, denote neighbor positions in the two words. Assume $|x_\circ|_{x_i} = |x_\circ|_{x_{i+1}} = |y_\circ|_{y_j} = |y_\circ|_{y_{j+1}} = 1$, meaning that the letters x_i and x_{i+1} (resp. y_j and y_{j+1}) occur only once in $x_\circ$ (resp. $y_\circ$). Thus, the conjugate of x that starts at x_{i+1} (or ends at x_i) is the unique word $x_{i+1} \ldots x_n x_1 \ldots x_i$, and likewise, the conjugate of y that starts at y_{j+1} (or ends at y_j) is also uniquely defined $y_{j+1} \ldots y_m y_1 \ldots y_j$. Thus, each of the nine rules applied to $x_\circ$ and $y_\circ$ cuts the two circular words at the same positions, and therefore, generates the same circular word $z_\circ = \left(x_{i+1} \ldots x_n x_1 \ldots x_i y_{j+1} \ldots y_m y_1 \ldots y_j\right)_\circ$. □

Note here that the hypothesis $|x_\circ|, |y_\circ| \geq 2$ cannot be dropped. For example, if $|x_\circ| = 1$, then indices taken modulo n give $x_i = x_{i+1}$, hence any rule whose first pair is (x_i,x_{i+1}) requires the factor $x_i x_i$ in $x_\circ$ and is therefore inapplicable, while rules with λ in that position remain applicable. For instance, with $x_\circ = (a)_\circ$, $y_\circ = (bcd)_\circ$ and $i = j = 1$, the rule $r_1 = (x_i,x_{i+1};y_j,y_{j+1}) = (a,a;b,c)$ generates nothing, whereas $r_2 = (x_i,\lambda;y_j,y_{j+1}) = (a,\lambda;b,c)$ produces $(acdb)_\circ$. Therefore, the results obtained by $r_1, \ldots, r_9$ do not coincide, and the same argument applies symmetrically when $|y_\circ| = 1$. Further, the scenario is similar in the case when one or both of the circular words are the empty word. In general, the Păun splicing rules $r = (x_i,x_{i+1};y_j,y_{j+1})$ and $r' = (\lambda,\lambda;y_j,y_{j+1})$ applied to $x_\circ$ and $y_\circ$ do not necessarily give the same results. On the one hand, while the rule r' can cut and linearize $x_\circ$ to obtain any conjugate of x, the rule r cuts and linearizes $x_\circ$ obtaining, e.g., $x_{i+1} \ldots x_n x_1 \ldots x_i$. However, if $x_\circ$ is built up only by one of the letters, i.e., it is of the form $(a^n)_\circ$ for a letter $a \in \Sigma$, then there is only one linear word representing it, and any cut will lead to this sole word.

Example 8. *Consider the circular words* $x_\circ = (x_1x_2x_3x_4)_\circ = (abcd)_\circ$ *and* $y_\circ = (y_1y_2)_\circ = (12)_\circ$ *in the alphabet* $\Sigma = \{a,b,c,d,1,2\} = \{a,b,c,d,1,2\}$*. For fixed indices* $i = 3$ *and* $j = 2$ *(*$x_i = x_3 = c$ *and* $y_j = y_2 = 2$*), each of the nine Păun splicing rules in Theorem* 1 *applied to* $x_\circ$ *and* $y_\circ$ *generates exactly the same circular word* $(x_4x_1x_2x_3y_1y_2)_\circ = (dabc12)_\circ$.

On the other hand, consider the circular word $z_\circ = (z_1z_2z_3z_4z_5)_\circ = (abcad)_\circ$ *where the letter a occurs twice. Fixing the indices* $i = 1$ *and* $j = 2$ *(*$z_i = z_1 = a$ *and* $y_j = y_2 = 2$*), the Păun circular splicing rule* $r_1 = (a,b;2,1)$ *applied to* $z_\circ$ *and* $y_\circ$ *generates the circular word* $(bcada12)_\circ$*, while the splicing rule* $r_2 = (a,\lambda;2,1)$ *applied to* $z_\circ$ *and* $y_\circ$ *generates two different circular words* $(bcada12)_\circ$ *and* $(dabca12)_\circ$.

In addition, the factor ab in the circular word $w_\circ = (w_1w_2w_3w_4w_5w_6)_\circ = (abcabd)_\circ$ *occurs twice. However, the Păun circular splicing rule* $r_1 = (a,b;2,1)$ *applied to* $w_\circ$ *and* $y_\circ$ *generates two different circular words* $(bcabda12)_\circ$ *and* $(bdabca12)_\circ$.

We recall the definitions of one-wheel and two-wheel concatenations of circular words from [17] to allow comparing them to the splicing operations.

Definition 4. *Let* $x_\circ$ *and* $y_\circ$ *be two circular words; their two-wheel concatenation is the sequence of the circular words obtained by:*

$$x_\circ y_\circ = x_\circ \cdot y_\circ = \{\, z \mid z = uv \text{ where } u \in x_\circ \text{ and } v \in y_\circ\} \tag{2}$$

Definition 5. *Let $x_\circ$ and $y_\circ$ be two circular words; their one-wheel concatenation is:*

$$x_\circ \odot y_\circ = \{w \in z_\circ \mid z = uv \text{ where } u \in x_\circ \text{ and } v \in y_\circ\} \quad (3)$$

It produces all cyclic permutations of the concatenation of any conjugate of x and any conjugate of y.

In [17], it was proved that the one-wheel concatenation is an operation that is commutative but not associative, while the two-wheel concatenation is associative but not commutative.

Theorem 2. *Let Σ be an arbitrary finite alphabet, and let $x_\circ = (x_1 \dots x_n)_\circ$ and $y_\circ = (y_1 \dots y_m)_\circ$ be any arbitrary circular words over Σ with indices taken modulo n and m respectively (so $x_{n+1} = x_1$ and $y_{m+1} = y_1$). Then, considering any Păun circular splicing operation $r = (u_1, u_2; u_3, u_4)$ the set of circular words obtained is a subset of the set obtained by the one-wheel concatenation applied to $x_\circ$ and $y_\circ$. Furthermore, applying all possible Păun circular splicing operations that produce a non-empty set, the union of the obtained circular words coincides with the set obtained by the one-wheel concatenation.*

Proof. We prove the two inclusions separately. Since $x_\circ$ is the set of all cyclic permutations of $x = x_1 \dots x_n$, we have:

$$x_\circ = \{x_{i+1} \dots x_n x_1 \dots x_i \mid i = 1,\dots,n\}.$$

Similarly,

$$y_\circ = \{y_{j+1} \dots y_m y_1 \dots y_j \mid j = 1,\dots,m\}.$$

And by Definition 5, $x_\circ \odot y_\circ$ is the set of circular words $\left(x_{i+1} \dots x_n x_1 \dots x_i y_{j+1} \dots y_m y_1 \dots y_j\right)_\circ$ for all $i = 1,\dots,n$ and $j = 1,\dots,m$. The proof continues with the verification of the following two claims:

Claim 1. *For every Păun rule $r = (u_1, u_2; u_3, u_4)$, the set of circular words produced by r from $x_\circ$ and $y_\circ$ is contained in $x_\circ \odot y_\circ$.*

Proof of the Claim. Indeed, by Definition 2, every application of r to $x_\circ$ and $y_\circ$ is determined by a pair $(h,k) \in \Sigma^* \times \Sigma^*$ such that $x_\circ = (u_2 h u_1)_\circ$ and $y_\circ = (u_4 k u_3)_\circ$, and it produces $(u_2 h u_1 u_4 k u_3)_\circ$. Several such pairs may exist, one for each occurrence of the specified subwords, and the rule is applied once for each of them; however, for every one of them the word $u_2 h u_1$ is a conjugate of x and $u_4 k u_3$ is a conjugate of y, so that the produced circular word is the circularization of the concatenation of a conjugate of x and a conjugate of y, i.e., an element of $x_\circ \odot y_\circ$. If no such pair exists, the rule produces the empty set, which is a subset of $x_\circ \odot y_\circ$ as well. This proves the claim, the first part of the statement of the theorem. □

Claim 2. *Every element of $x_\circ \odot y_\circ$ is produced by some Păun circular splicing rules.*

Proof of the Claim. Let $z_\circ = \left(x_{i+1} \dots x_n x_1 \dots x_i y_{j+1} \dots y_m y_1 \dots y_j\right)_\circ$ be an arbitrary element of $x_\circ \odot y_\circ$, with i and j fixed. Assume first $n,m \geq 2$, and consider the Păun circular splicing rule $r = (x_i,x_{i+1};y_j,y_{j+1})$. It is applicable to $x_\circ$ and $y_\circ$, since $x_\circ = (x_{i+1} h x_i)_\circ$ with $h = x_{i+2} \dots x_n x_1 \dots x_{i-1}$ and, analogously, $y_\circ = \left(y_{j+1} k y_j\right)_\circ$ with $k = y_{j+2} \dots y_m y_1 \dots y_{j-1}$. The corresponding application produces exactly $z_\circ$. If the factor $x_i x_{i+1}$ occurs at other positions of $x_\circ$, and/or $y_j y_{j+1}$ at other positions of $y_\circ$, then r produces further circular words, which, by Claim 1, also belong to $x_\circ \odot y_\circ$; this does not affect the argument, as we only need to show that $z_\circ$ to be produced.

If $n \leq 1$, then $x_\circ$ has a unique linearization, and we replace the first pair of the rule by (λ,λ); analogously, if $m \leq 1$, we replace the second pair by (λ,λ). In each case, the obtained rule is applicable and produces $z_\circ$. □

Now, continuing the proof of the theorem, by Claim 1, the union of the sets produced by all the Păun rules is contained in $x_\circ \odot y_\circ$, and by Claim 2 it contains it in the other way around; hence the two sets are equal, which proves the second part of the statement of the theorem. □

Theorem 2 is a characterization rather than a technically involved result: once the splicing rules are read as prescriptions of the positions at which the input circular words may be cut, both inclusions follow from Definitions 2 and 5 as it is shown above. Its role in this paper is to fix the exact ceiling of one application of the operations: namely, that whatever the contexts are, a single step cannot produce more than the one-wheel concatenation of its inputs, and that the whole of it is obtained when no site is prescribed. All the subsequent results rest on this bound: Corollary 1 and Proposition 2 describe how the produced sets vary with the contexts, Theorems 7 and 9 characterize the cases in which the two-wheel concatenation is reached instead, Theorems 7 and 8 place the circular insertion in the same picture, and Corollary 2 transfers the bound to the generated circular languages.

Example 9. *Consider the circular words* $x_\circ = (x_1x_2x_3)_\circ = (abc)_\circ$ *and* $y_\circ = (y_1y_2)_\circ = (12)_\circ$, *with each of the following Păun circular splicing rules* $(b,c;1,2)$, $(b,\lambda;1,2)$, $(\lambda,c;1,\lambda)$, *over the alphabet* $\Sigma = \{a,b,c,1,2\}$. *Each rule applied to* $x_\circ$ *and* $y_\circ$ *corresponds to the same cut-and-paste site, and generates the same circular word* $(cab21)_\circ$.

Corollary 1. *The Păun circular splicing rule* $(\lambda,\lambda;\lambda,\lambda)$ *applied to any arbitrary circular words* $x_\circ$ *and* $y_\circ$ *generates their one-wheel concatenation.*

Proof. By Claim 1 in the proof of Theorem 2, the set of circular words produced by this rule is contained in $x_\circ \odot y_\circ$. Conversely, let $z_\circ \in x_\circ \odot y_\circ$, so that z is a conjugate of uv for some $u \in x_\circ$ and $v \in y_\circ$. Since λ is both a prefix and a suffix of every word, we have $x_\circ = (\lambda u \lambda)_\circ$ and $y_\circ = (\lambda v \lambda)_\circ$, i.e., the rule $(\lambda,\lambda;\lambda,\lambda)$ is applicable with $h = u$ and $k = v$, and by Definition 2 it produces $(uv)_\circ = z_\circ$. Hence every element of $x_\circ \odot y_\circ$ is obtained. □

Corollary 1 concerns the rule with empty contexts, which is the least restrictive one, and one may ask whether this case is not too degenerate to be informative. Two observations show that it is not.

First, the inclusion proved in the first part of Theorem 2 holds for arbitrary rules: whatever the contexts are, the circular words produced by a Păun splicing are always among those given by the one-wheel concatenation, and the same holds for the circular insertion (Theorem 7), where the contexts are arbitrary as well. The empty contexts are needed only for the reverse inclusion, i.e., to reach every element of the concatenation. The equivalence results are therefore not about one particular rule, but the two ends of a general property: the one-wheel concatenation is an upper bound for all these operations, which is attained exactly when no site is prescribed.

Second, between these two ends the behaviour is monotone with respect to the contexts, in the following sense.

Proposition 2. *Let* $r = (u_1, u_2; u_3, u_4)$ *and* $r' = (u'_1, u'_2; u'_3, u'_4)$ *be two Păun splicing rules such that* u'_1 *is a suffix of* u_1, u'_2 *is a prefix of* u_2, u'_3 *is a suffix of* u_3 *and* u'_4 *is a prefix of* u_4. *Then, for*

any circular words $x_\circ$ and $y_\circ$, the set of circular words produced by r is a subset of the set produced by r'.

Proof. Assume that r is applicable to $x_\circ$ and $y_\circ$, i.e., $x_\circ = (u_2hu_1)_\circ$ and $y_\circ = (u_4ku_3)_\circ$ for some $h,k \in \Sigma^*$, then a produced circular word being $(u_2hu_1u_4ku_3)_\circ$. By the hypothesis, there are words $p,p',q,q' \in \Sigma^*$ with $u_1 = pu_1'$, $u_2 = u_2'p'$, $u_3 = qu_3'$ and $u_4 = u_4'q'$. Putting $h' = p'hp$ and $k' = q'kq$, we obtain $u_2hu_1 = u_2'h'u_1'$ and $u_4ku_3 = u_4'k'u_3'$. Hence $x_\circ = (u_2'h'u_1')_\circ$ and $y_\circ = (u_4'k'u_3')_\circ$, so r' is also applicable to them, and by Definition 2 it can produce $(u_2'h'u_1'u_4'k'u_3')_\circ = (u_2hu_1u_4ku_3)_\circ$, i.e., the same circular word. □

In particular, taking $u_1' = u_2' = u_3' = u_4' = \lambda$ gives the inclusion of Corollary 1, so the rule $(\lambda, \lambda; \lambda, \lambda)$ is the maximum of this ordering, while prescribing longer sites yields smaller sets of products; the results of Theorem 1 are the special case in which the restriction imposed by the contexts does not change the set at all. Together with Claim 2 in the proof of Theorem 2, which recovers the whole one-wheel concatenation as the union over all rules, this describes how the produced sets vary with the contexts, and not only at the degenerate end. Moreover, the characterizations of the forthcoming Theorems 7 and 9 are not restricted to empty contexts either: there the input circular words are forced to be of a special form, whereas the contexts may be arbitrary factors of them.

Beyond the one-wheel concatenation, which combines conjugates of two circular words into a single circular structure, the two-wheel concatenation preserves the independent circularity of each component before concatenation. However, unlike circular splicing, the result of a two-wheel concatenation is not necessarily a circular word, since the operation may yield a non-circular configuration. As the cases where one-wheel and two-wheel concatenations coincide were investigated in [17], it is now natural to examine the conditions under which the outcomes of Păun splicing coincide with those of two-wheel concatenation. Such a correspondence provides a precise structural characterization of when the generative power of Păun's operation matches the two-wheel concatenation of two circular words. The next technical lemma and its proof are helpful to establish our further results.

Lemma 1. *Let $x_\circ$ and $y_\circ$ be two circular words over Σ. Then, the two-wheel concatenation of $x_\circ$ and $y_\circ$ yields a circular word language if and only if $x,y \in \{a\}^*$ for some $a \in \Sigma$, or $x_\circ = \lambda_\circ$, or $y_\circ = \lambda_\circ$.*

Proof. (Necessary condition) Assume that $x_\circ \cdot y_\circ$ is a circular word language (i.e., closed under conjugation), and suppose that $x \neq \lambda$ and $y \neq \lambda$. Let A and B denote the sets of letters occurring in x and in y, respectively. We distinguish three cases, which cover all the possibilities.

- Case 1: there is a letter a that is in A but not in B. Every element of $x_\circ \cdot y_\circ$ is of the form uv with $u \in x_\circ$ and $v \in y_\circ$, with $v \neq \lambda$. So, its last letter is a letter of B; hence, no element of the set ends with a. On the other hand, a occurs in x, so it occurs in every element of the set, and therefore some conjugate of such an element ends with a. By the assumed closure under conjugation, this conjugate belongs to the set, a contradiction.
- Case 2: there is a letter b that is in B but not in A. The argument is symmetric: every element uv of the set begins with a letter of A because $u \neq \lambda$, so no element begins with b, while some conjugate of an element does; again, a contradiction.
- Case 3: $A = B$ and $\|A\| \geq 2$. Fix $a \in A$, and let $n = max\{k \in \mathbb{N}_0 \mid a^k \in Fact(x_\circ)\}$ and $m = max\{k \in \mathbb{N}_0 \mid a^k \in Fact(y_\circ)\}$; both are well defined and $n,m \geq 1$, since a occurs in x and in y, while $n < |x|$ and $m < |y|$, since x and y contain a letter different from a. As the occurrences of a^n in the circular word $x_\circ$ are preceded and followed by

letters different from a, there is a linearization $u \in x_\circ$ ending with a^n, and, analogously, a linearization $v \in y_\circ$ beginning with a^m. Then $uv \in x_\circ \cdot y_\circ$ has the factor a^{n+m}, and therefore it has a conjugate z beginning with a^{n+m}. By the assumed closure under conjugation, $z \in x_\circ \cdot y_\circ$, i.e., $z = u'v'$ for some $u' \in x_\circ$ and $v' \in y_\circ$. Since u' is a conjugate of x, it is not a power of a and it is not empty, so the maximal prefix of z of the form a^* is the maximal prefix of u' of this form, whose length is at most n. This contradicts the fact that z begins with a^{n+m}, because $m \geq 1$.

Consequently, if $x \neq \lambda$ and $y \neq \lambda$, then $A = B$ and $\|A\| = 1$, i.e., $x,y \in \{a\}^*$ for some $a \in \Sigma$; otherwise, $x_\circ = \lambda_\circ$ or $y_\circ = \lambda_\circ$.

(Sufficient condition) If $x,y \in \{a\}^*$, then $x_\circ \cdot y_\circ = \{a^{|x|+|y|}\}$, and if one of the circular words is $\lambda_\circ$, then $x_\circ \cdot y_\circ$ is $x_\circ$ or $y_\circ$; all these sets are circular word languages, hence the two-wheel concatenation is a circular word language. □

Theorem 3. *Let Σ be an arbitrary finite alphabet, and let $x_\circ = (x_1 \dots x_n)_\circ$ and $y_\circ = (y_1 \dots y_m)_\circ$ be any two arbitrary circular words with indices taken modulo n and m respectively (so $x_{n+1} = x_1$ and $y_{m+1} = y_1$). Then, the Păun circular splicing applied to them has the same result as their two-wheel concatenation if and only if (at least) one of the following three cases holds:*

1. *$x,y \in \{a\}^*$ for any $a \in \Sigma$, and the rule $r = (u_1, u_2; u_3, u_4)$ has the properties*
   - *$u_1u_2 \in \{ a^p \mid p \in \mathbb{N}_0$ with $0 \leq p \leq |x| \}$ and*
   - *$u_3u_4 \in \{ a^q \mid q \in \mathbb{N}_0$ with $0 \leq q \leq |y| \}$;*
2. *$x_\circ = (x_1 \dots x_n)_\circ \neq \lambda_\circ$, $y_\circ = \lambda_\circ$ and $r = (u_1,u_2;\lambda,\lambda)$ where $u_1u_2 \in Fact(x_\circ)$;*
3. *$x_\circ = \lambda_\circ$, $y_\circ = (y_1 \dots y_m)_\circ \neq \lambda_\circ$ and $r = (\lambda, \lambda; u_3, u_4)$ where $u_3u_4 \in Fact(y_\circ)$.*

Proof. (Necessary condition) Assume that the Păun circular splicing operation applied to $x_\circ$ and $y_\circ$ results in $x_\circ \cdot y_\circ$. Since this operation produces a circular word language, $x_\circ \cdot y_\circ$ is one, and by Lemma 1, either $x,y \in \{a\}^*$ for some $a \in \Sigma$, or one of the two circular words is $\lambda_\circ$.

Let $x,y \in \{a\}^*$. Then, $x_\circ \cdot y_\circ = \{a^{|x|+|y|}\}$, and this must be produced by the Păun splicing rule $r = (u_1, u_2; u_3, u_4)$ so that $x_\circ = (u_2hu_1)_\circ$ and $y_\circ = (u_4ku_3)_\circ$ for some $h,k \in \Sigma^*$. Hence:

$$u_1u_2 \in Fact(x_\circ) = Fact(a^{|x|}) \text{ and } u_3u_4 \in Fact(y_\circ) = Fact(a^{|y|}).$$

Thus, we obtain: $u_1u_2 \in \{ a^p \mid p \in \mathbb{N}_0$ with $0 \leq p \leq |x| \}$ and $u_3u_4 \in \{ a^q \mid q \in \mathbb{N}_0$ with $0 \leq q \leq |y| \}$. This yields case (1), including the case when both of $x_\circ$ and $y_\circ$ are the empty circular word.

On the other hand, if $x_\circ = (x_1 \dots x_n)_\circ \neq \lambda_\circ$ and $y_\circ = \lambda_\circ$, then $x_\circ \cdot y_\circ$ is just the set of linear words equal to conjugates of x. For equality with the result obtained by the Păun splicing rule, we must be able to produce the circular word $x_\circ$. This can be done by any applicable Păun splicing rule, i.e., by any rule of the form $(u_1,u_2;\lambda,\lambda)$, where $u_1u_2 \in Fact(x_\circ)$ (as a special case, as we have shown before, it can be done by the rule $(\lambda,\lambda;\lambda,\lambda)$). Thus, case (2) holds.

Analogously, having $x_\circ = \lambda_\circ$ and $y_\circ = (y_1 \dots y_m)_\circ \neq \lambda_\circ$ yields that the rules of the form $(\lambda,\lambda;u_3,u_4)$ with $u_3u_4 \in Fact(y_\circ)$ satisfy case (3). This completes the necessity direction.

(Sufficient condition) We now show that each of the listed cases is sufficient to guarantee equality.

Case 1: Let $x,y \in \{a\}^*$ for any $a \in \Sigma$, and assume that the splicing rule $r = (u_1, u_2; u_3, u_4)$ has the property

$$u_1u_2 \in \{ a^p \mid p \in \mathbb{N}_0 \text{ with } 0 \leq p \leq |x| \} \text{ and } u_3u_4 \in \{ a^q \mid q \in \mathbb{N}_0 \text{ with } 0 \leq q \leq |y| \}.$$

Take $h = a^{|x|-p}$ and $k = a^{|y|-q}$ so that $x = u_2hu_1$ and $y = u_4ku_3$. Then, applying r yields $(u_2hu_1u_4ku_3)_\circ = \left\{a^{|x|+|y|}\right\} = x_\circ \cdot y_\circ$. Hence, the result.

Case 2: Since $u_1u_2 \in Fact(x_\circ)$, there exists an element of $x_\circ$ of the form $x_\circ = (u_2hu_1)_\circ$ for some $h \in \Sigma^*$. Moreover, $y_\circ = \lambda_\circ = (\lambda\lambda\lambda)_\circ$. Applying the Păun splicing rule $r = (u_1,u_2;\lambda,\lambda)$, the two circular words are linearized as u_2hu_1 and λ, respectively. By Definition 2, the resulting circular word is $(u_2hu_1)_\circ = x_\circ$. On the other hand, by the definition of the two-wheel concatenation, $x_\circ \cdot y_\circ = x_\circ \cdot \lambda_\circ = x_\circ$. Hence, the Păun circular splicing applied to $x_\circ$ and $y_\circ$ has the same result as their two-wheel concatenation.

The proof of Case 3 is analogous to Case 2: in fact, $\lambda_\circ$ is the unit element of the two-wheel concatenation; therefore, the result will be exactly $x_\circ \cdot y_\circ$, thus the equality. □

Example 10. *The following cases illustrate Theorem* 3 *and show that its conditions cannot be weakened. Let* $\Sigma = \{a,b,c\}$.

- *Let* $x_\circ = (a^2)_\circ$, $y_\circ = (a^3)_\circ$ *and* $r = (a,a;\lambda,a)$. *Here* $u_1u_2 = a^2$ *and* $u_3u_4 = a$, *so Case* 1 *holds, and* r *produces* $(a^5)_\circ = \{a^5\} = x_\circ \cdot y_\circ$. *However, for* $r' = (a^3,a;\lambda,a)$ *we have* $u_1u_2 = a^4$, *whose length exceeds* $|x| = 2$; *hence* $a^4 \notin Fact(x_\circ)$, *the rule is not applicable, and it produces the empty set instead of* $x_\circ \cdot y_\circ$. *Thus, the length bound in Case* 1 *is necessary.*
- *Let* $x_\circ = (abc)_\circ$, $y_\circ = \lambda_\circ$ *and* $r = (b,c;\lambda,\lambda)$. *Since* $bc \in Fact(x_\circ)$, *Case* 2 *holds: r linearizes* $x_\circ$ *to cab and produces* $(cab)_\circ = \{abc,bca,cab\} = x_\circ \cdot \lambda_\circ$.
- *Let* $x_\circ = (ab)_\circ$ *and* $y_\circ = (a)_\circ$, *which satisfy none of the three cases. Then* $x_\circ \cdot y_\circ = \{aba,baa\}$: *the letter b, which does not occur in y, is not the last letter of any of its words, so the conjugate aab is missing and* $x_\circ \cdot y_\circ$ *is not a circular word language. Since every Păun splicing produces circular words, no Păun rule yields* $x_\circ \cdot y_\circ$; e.g., $(\lambda,\lambda;\lambda,\lambda)$ *produces* $(aab)_\circ = \{aab,aba,baa\} \neq x_\circ \cdot y_\circ$.

In the next part, we characterize when the Pixton circular splicing operation reproduces the one-wheel concatenation.

Theorem 4. *Let* $x_\circ = (x_1 \ldots x_n)_\circ$ *and* $y_\circ = (y_1 \ldots y_m)_\circ$ *be any arbitrary circular words over the finite alphabet* Σ *with indices taken modulo n and m respectively. Then applying Pixton's circular splicing with rules* $r = (\lambda,\lambda;\lambda)$ *and* $\bar{r} = (\lambda,\lambda;\lambda)$ (*in fact, this rule is associated with itself*) *leads to the same set as the one-wheel concatenation* $x_\circ \odot y_\circ$.

Proof. On the one hand, in a Pixton rule $r = (\alpha,\alpha';\beta)$ setting both α and α' to the empty word, the rule can be applied anywhere to cut and linearize the two circular input words. Further, by setting β to the empty word we do not permit changes by rewriting, and thus, the conjugates of the original circular words are concatenated, and the result is closed to be a circular word again. Thus, in this case, exactly the one-wheel concatenation is obtained. □

Example 11. *Let* $x_\circ = (ab)_\circ$ *and* $y_\circ = (c)_\circ$. *By Corollary 1, the Păun rule* $(\lambda, \lambda; \lambda, \lambda)$ *linearizes* $x_\circ$ *to ab or ba and* $y_\circ$ *to c, producing the two distinct circular words* $(abc)_\circ$ *and* $(bac)_\circ$; *this is exactly* $x_\circ \odot y_\circ$. *By Theorem 6, the Pixton rule pair* $r = \bar{r} = (\lambda,\lambda;\lambda)$ *produces the same set.*

Now, analogously to Theorem 3, we can also state a result when applying Pixton's circular splicing results in the same set as the two-wheel concatenation of the input circular words.

Theorem 5. *Let* $x_\circ = (x_1 \ldots x_n)_\circ$ *and* $y_\circ = (y_1 \ldots y_m)_\circ$ *be any arbitrary circular words over the finite alphabet* Σ *with indices taken modulo n and m respectively. The Pixton circular splicing applied to them has the same result as their two-wheel concatenation if and only if* (*at least*) *one of the following three cases holds*:

1. $x,y \in \{a\}^*$ *for any* $a \in \Sigma$, *and* $r,\bar{r}$ *such that* $r = (a^p,a^q;\beta)$ *and* $\bar{r} = (a^q, a^p; \beta\prime)$, *where* $p,q \in \mathbb{N}_0$ *with* $0 \leq p \leq |x|$, $0 \leq q \leq |y|$ *and* $\beta\beta' = a^{p+q}$;
2. $x_\circ = (x_1 \ldots x_n)_\circ \neq \lambda_\circ$, $y_\circ = \lambda_\circ$ *and* $r = (\alpha,\lambda;\beta)$ *and* $\bar{r} = (\lambda, \alpha; \beta\prime)$ *such that* $\alpha \in Fact(x_\circ)$ *and* $\beta\beta' = \alpha$;
3. $x_\circ = \lambda_\circ$, $y_\circ = (y_1 \ldots y_m)_\circ \neq \lambda_\circ$ *and* $r = (\lambda,\alpha';\beta)$ *and* $\bar{r} = (\alpha\prime, \lambda; \beta\prime)$ *such that* $\alpha' \in Fact(y_\circ)$ *and* $\beta\beta' = \alpha'$.

Proof. (Necessary condition) Assume that the Pixton circular splicing operations defined by a pair $(r,\bar{r})$ applied to $x_\circ$ and $y_\circ$ results in $x_\circ \cdot y_\circ$. As this operation also produces a circular word language, Lemma 1 applies exactly as in Theorem 3, so either $x,y \in \{a\}^*$ for some $a \in \Sigma$, or one of the two circular words is $\lambda_\circ$.

Let $x,y \in \{a\}^*$. Writing $r = (\alpha,\alpha';\beta)$ and $\bar{r} = (\alpha', \alpha; \beta')$, the applicability to $x_\circ$ and $y_\circ$ gives $\alpha = a^p$, $\alpha' = a^q$ with $0 \leq p \leq |x|$ and $0 \leq q \leq |y|$, and the produced circular word must be $\left(a^{|x|+|y|}\right)_\circ$; comparing the lengths of $\varepsilon\beta\varepsilon'\beta' = a^{|x|-p}\beta a^{|y|-q}\beta'$ and of $a^{|x|+|y|}$ yields $\beta\beta' = a^{p+q}$, which is alternative (1) in the theorem. This case also covers the case that both of $x_\circ$ and $y_\circ$ are the empty circular word.

However, if either $x_\circ = \lambda_\circ$ or $y_\circ = \lambda_\circ$, then the proof is immediate and yields alternatives (2) or (3). Indeed, suppose $y_\circ = \lambda_\circ$. Then every conjugate $u \in x_\circ$ must be obtainable as the output of the Pixton rule pair $(r, \bar{r})$ applied to representatives of $x_\circ$ and the empty circular word. Writing a general pair as $r = (\alpha,\alpha';\beta)$ with $\bar{r} = (\alpha', \alpha; \beta')$, we must have $\alpha' = \lambda$; hence $r = (\alpha,\lambda;\beta)$ and $\bar{r} = (\lambda, \alpha; \beta')$. To produce u, we must have a decomposition $u = \varepsilon\beta$ with $\alpha\varepsilon \in x_\circ$; comparing this equality for the various conjugates forces $\beta\beta' = \alpha$. This is exactly alternative (2). The symmetric argument gives (3) when $x_\circ = \lambda_\circ$.

(Sufficient condition) We show each alternative implies that applying the given pair of rules, the two-wheel concatenation $x_\circ \cdot y_\circ$ is obtained.

- If alternative (1) holds: $x = a^{|x|}$, $y = a^{|y|}$ and there exists $r = (a^p,a^q;\beta)$ with partner $\bar{r} = (a^q, a^p; \beta\prime)$ and $\beta\beta' = a^{p+q}$. Take any $u = a^s \in x_\circ$, $v = a^t \in y_\circ$. Choose $\varepsilon = a^{s-p}$, $\varepsilon' = a^{t-q}$ (these are nonnegative powers because $0 \leq p \leq |x|$, $0 \leq q \leq |y|$). Then the Pixton pair applied to $(a^p\varepsilon)_\circ$ and $(a^q\varepsilon\prime)_\circ$ produces:

$$\varepsilon\beta\varepsilon'\beta' = a^{s-p}\beta a^{t-q}\beta' = a^{s-p}a^{p+q}a^{t-q} = a^{s+t} = uv.$$

  Thus, every $uv \in x_\circ \cdot y_\circ$ is realized; conversely, any Pixton output is plainly a concatenation of a conjugate of $x_\circ$ and a conjugate of $y_\circ$. Hence, the equality holds.
- If alternative (2) holds: $y_\circ = \lambda_\circ$ and $r = (\alpha,\lambda;\beta)$, $\bar{r} = (\lambda, \alpha; \beta')$ with $\alpha \in Fact(x_\circ)$ and $\beta\beta' = \alpha$. Take any $u = \alpha\varepsilon \in x_\circ$, then applying the Pixton pair to $(\alpha\varepsilon)_\circ$ and $\lambda_\circ$ yields:

$$\varepsilon\beta\varepsilon'\beta' = \varepsilon\beta\beta' = \varepsilon\alpha = u.$$

  So, every conjugate u of x is produced; the concatenation with the empty word yields exactly $x_\circ \cdot y_\circ$.
- The proof of alternative (3) is identical to (2). □

Example 12. *Let* $x_\circ = \left(a^2\right)_\circ$ *and* $y_\circ = (a)_\circ$, *and take* $p = q = 1$. *The pair* $r = (a,a;a)$, $\bar{r} = (a, a; a)$ *satisfies* $\beta\beta' = a^{p+q} = a^2$, *so Case 1 of Theorem* 5 *holds: with* $\varepsilon = a$ *and* $\varepsilon' = \lambda$, *it produces* $(\varepsilon\beta\varepsilon'\beta')_\circ = \left(a^3\right)_\circ = x_\circ \cdot y_\circ$. *In contrast, the pair* $r = (a,a;a^2)$, $\bar{r} = (a, a; a)$ *has* $\beta\beta' = a^3 \neq a^2$, *and it produces* $\left(a^4\right)_\circ \neq x_\circ \cdot y_\circ$, *since Pixton's operation may change the length.*

4. Computing by Circular Insertion–Deletion Operations

The insertion of linear DNA strands into linear DNA strands is modeled by contextual insertion [14]. In the circular framework, insertion–deletion operations are particularly relevant for representing transformations of circular DNA, leading to computational models grounded in circular molecular structures (and, in the theoretical description, by circular word structures).

4.1. Circular Insertion

In [15], the circular context-guided insertion was introduced. It was first defined as the insertion, between the factors u and v of a linear word x, of each of the linear words representing a circular permutation of y. We recall a form of the insertion operation [15]:

Definition 6 (Circular Insertion into Linear Word). *Let Σ be a finite alphabet. A circular insertion rule is represented by a triple of the form $r_{ins} = (u, \lambda/\beta_\circ, v)$, where $u, v \in \Sigma^*$ and $\beta \in \Sigma^+$. Then, given a linear word $x = x_1uvx_2$, the application of the rule inserts any conjugate γ of $\beta_\circ$ between u and v, obtaining $y = x_1u\gamma vx_2$. We denote this operation by:*

$$x \vdash_{r_{ins}} y = \{ x_1u\gamma vx_2 \mid x = x_1uvx_2,\ \gamma \in \beta_\circ, x_1, x_2 \in \Sigma^* \} \tag{4}$$

Although an application of the rule may give only one particular word, as we have already noted for the splicing operations, in general a set of possible results can be obtained, as we have written here in the definition. A graphical explanation is also shown in Figure 4. On the one hand, if x does not contain uv as a factor, then the result of the circular contextual insertion of any word into x is nothing, i.e., the empty set. The cardinality of the result of $x \vdash_{r_{ins}} y$ ranges from 0 to the number of occurrences of uv times $|\beta_\circ|$. We note here that in some practical applications the inserted circular strands are short, and in some models only some of the conjugates can be inserted based also on the context where the cycle is broken.

Based on the above definition, since we are interested in having circular words as results of an operation, we introduce the insertion of a circular strand of DNA into another circular strand (see Figure 5). In order to simulate this process, we examine here a generalization of word insertions.

Definition 7 (Circular Insertion into Circular Word). *Let Σ be a finite alphabet. A circular insertion rule is represented by a triple of the form $r_{ins} = (u,\lambda/\beta_\circ,v)$, where $u, v \in \Sigma^*$ and $\beta \in \Sigma^+$. Then, having a circular word $x_\circ = (uvx_1)_\circ$, the application of the rule inserts any conjugate γ of $\beta_\circ$ between u and v, obtaining $y_\circ = (u\gamma vx_1)_\circ$. We denote this operation by:*

$$x_\circ \vdash_{r_{ins}} y_\circ = \{z \in (u\gamma vx_1)_\circ \mid x = uvx_1,\ \gamma \in \beta_\circ, x_1 \in \Sigma^*\} \tag{5}$$

In Definition 6, the contexts u and v are searched for in a linear word x, i.e., in one fixed linearization of the molecule, and the result $x_1u\gamma vx_2$ is again a linear word (see Figure 4). In Definition 7, they are searched for in the circular word $x_\circ$, i.e., in all of its linearizations at the same time, and the result is a circular word (see Figure 5). The two operations therefore differ in three respects. First, Definition 7 is closed on circular words, so it can be iterated on cyclic molecules and compared with the circular splicing operations and with the concatenations of circular words (Theorems 13 and 14), which is not possible for Definition 6. Second, Definition 7 is a conservative extension of Definition 6: by Lemma 2, it can be obtained by applying Definition 6 to all linearizations of $x_\circ$ and closing the results under conjugation. Third, and most importantly, this closure is a genuine enlargement and

not a reformulation: as we will show, by Lemma 3, we only have the inclusions $Z \subseteq Z' \subseteq Z''$, and Theorem 6 characterizes when equality holds, the inclusions being strict as soon as $x_\circ$ has an occurrence of the context uv that spans the position at which x was cut, i.e., an occurrence that no linear representative of the molecule exhibits (see Example 13, where $x = cdaab$ yields nothing, while $x_\circ = (cdaab)_\circ$ yields three circular words). In biological terms, a cyclic molecule has no distinguished starting point, so the recognition site of the enzyme may lie across any point of the cycle; a model based on a fixed linearization would discard exactly these occurrences.

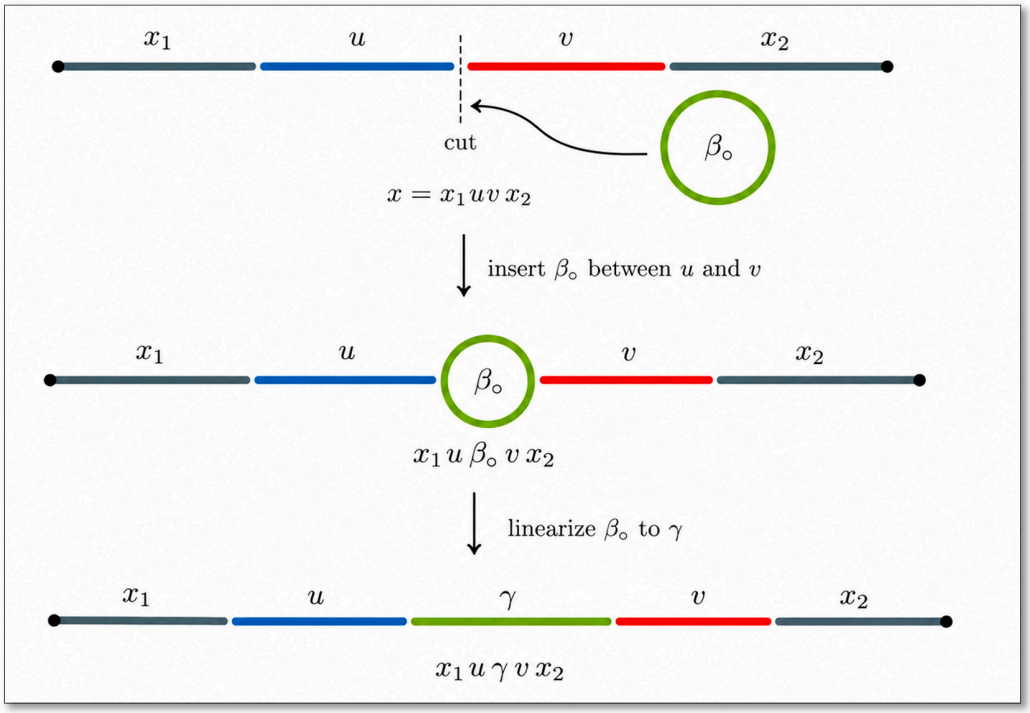


Figure 4. Circular Insertion into a Linear Word.

Observe that, since it is clear from the context, we use the same notation for a circular insertion rule whether we apply it to a linear or to a cyclic word.

Now, as a preliminary result about circular insertions, we show that our new definition is a natural extension of Definition 6 to circular words (i.e., it makes the insertion applicable to the case when both the original words are cyclic words, not only the one that is inserted).

Lemma 2. *Applying an insertion rule $r_{ins} = (u,\lambda/\beta_\circ,v)$ to a circular word $x_\circ$ yields the same result as the following process: linearize the circular word $x_\circ$ in all possible ways; apply the rule r_{ins} to each linear word by inserting any conjugate γ of $\beta_\circ$ between u and v in them (according to Definition 6); then close each resulting word back into its circular form.*

Proof. The result is a direct consequence of the new definition when it is compared with the original definition applied for inserting circular words into linear words. □

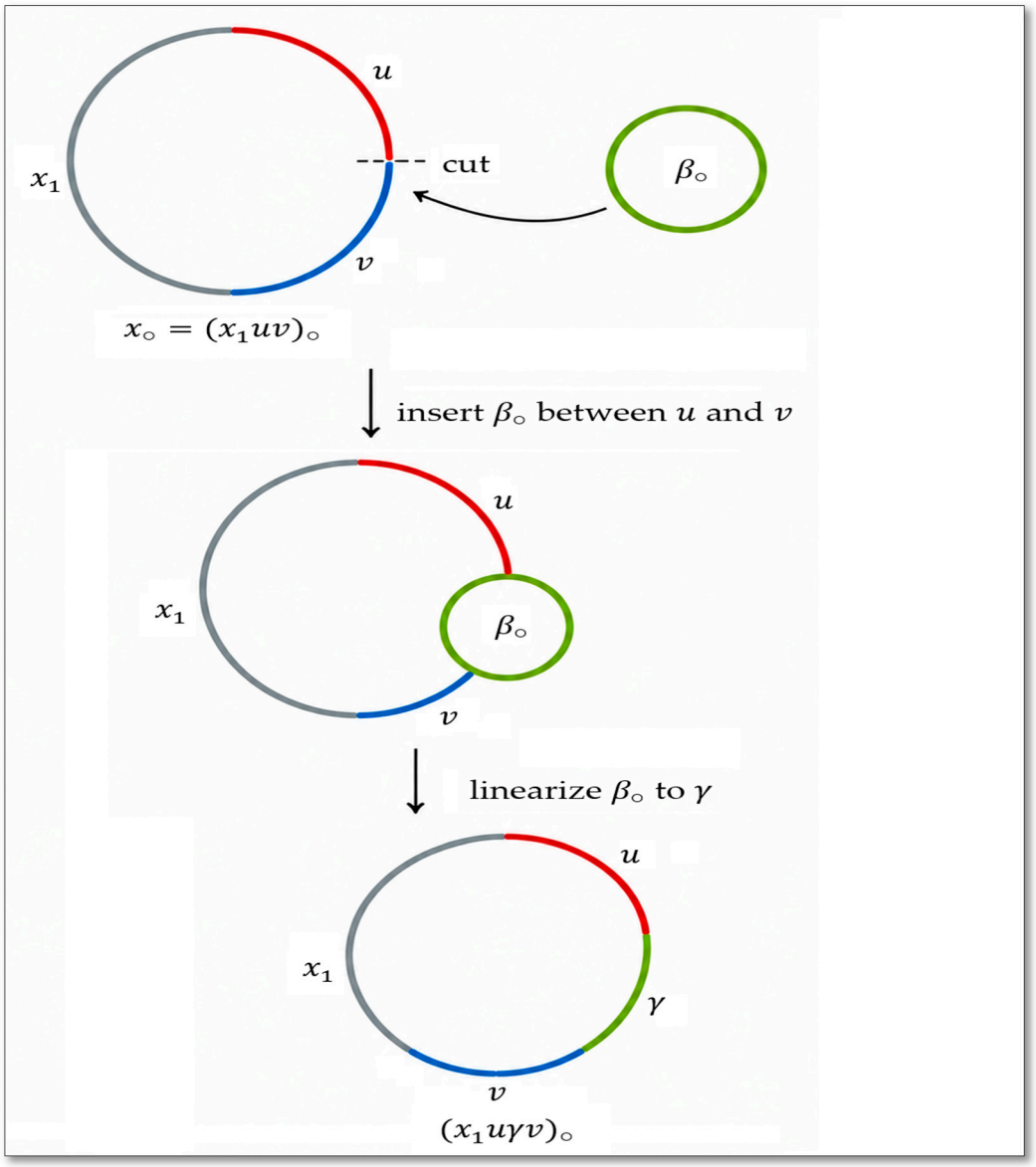


Figure 5. Circular Insertion into a Circular Word.

Lemma 3. *Let $r_{ins} = (u,\lambda/\beta_\circ,v)$ be a circular insertion rule over the alphabet Σ and a linear word x be given. Let the sets Z, Z' and Z'' represent the results of operations r_{ins} to x, the set of all conjugates of this set and the application of r_{ins} to the circular word $x_\circ$ as a circular insertion to a circular word, respectively. Then, we have the following relation:*

$$Z \subseteq Z' \subseteq Z'' \tag{6}$$

Proof. The first part of the proof is trivial, since Z′, as the set of all conjugates of Z, must be a superset of Z. On the other hand, each element of Z′ is a conjugate of a word obtained by inserting a conjugate γ of $\beta_\circ$ between u and v in x (if such position exists). However, in this case, this word is also a conjugate of a linear word which is in $x_\circ \vdash_{r_{ins}} y_\circ$. □

The previous lemma shows a real hierarchy of the obtained sets. To see and prove that these sets may not be equal, we present the following examples.

Example 13. *Let* $\Sigma = \{a,b,c,d,0,1,2\}$ *be an alphabet, and let* $r_{ins} = (ab,\lambda/(012)_\circ,cd)$ *be a circular insertion rule. Considering the linear word* $x = cdaab$ *over* Σ*, the rule* r_{ins} *applied to* x *generates nothing other than the empty set since it does not contain abcd as a factor. Hence,* $x \vdash_{r_{ins}} y = \{\}$ *and* $(x \vdash_{r_{ins}} y)_\circ = \{\}$*. However, the rule* r_{ins} *applied to the circular word* $x_\circ = (cdaab)_\circ$ *generates* $x_\circ \vdash_{r_{ins}} y_\circ = (ab012cda)_\circ \cup (ab120cda)_\circ \cup (ab201cda)_\circ$.

Moreover, applying the rule r_{ins} *to the linear word* $w = abcd$*, it produces* $w \vdash_{r_{ins}} y = \{ab012cd, ab120cd, ab201cd\}$ *while the set of all its conjugates is*

$$(w \vdash_{r_{ins}} y)_\circ = \left\{ \begin{matrix} ab012cd, b012cda, 012cdab, 12cdab0, 2cdab01, cdab012, dab012c \\ ab120cd, b120cda, 120cdab, 20cdab1, 0cdab12, cdab120, dab120c \\ ab201cd, b201cda, 201cdab, 01cdab2, 1cdab20, cdab201, dab201c \end{matrix} \right\}$$

and

$$w_\circ \vdash_{r_{ins}} y_\circ = (ab012cd)_\circ \cup (ab120cd)_\circ \cup (ab201cd)_\circ$$

Now, we turn to an important result by further analyzing this second part of the relation.

Theorem 6. *Let* Σ *be a finite alphabet, and* $r_{ins} = (u,\lambda/\beta_\circ,v)$ *be a circular insertion rule, where* $u, v \in \Sigma^*$ *and* $\beta \in \Sigma^+$*. For any linear word* x*,*

1. *If* $(x \vdash_{r_{ins}} y)_\circ = x_\circ \vdash_{r_{ins}} y_\circ$*, then* $uv \in Fact(x)$ *or* $uv \notin Fact(x_\circ)$.
2. *If* $uv \notin Fact(x_\circ)$*, or the number of occurrences of* uv *in* x *is maximal in the sense that the suffix of* x *concatenated to the prefix of* x *does not provide any extra occurrences of* uv*, then* $(x \vdash_{r_{ins}} y)_\circ = x_\circ \vdash_{r_{ins}} y_\circ$.

Proof. Case 1: Assume that $(x \vdash_{r_{ins}} y)_\circ = x_\circ \vdash_{r_{ins}} y_\circ$. Let us suppose that $uv \notin Fact(x)$ and $uv \in Fact(x_\circ)$. Then, $(x \vdash_{r_{ins}} y)_\circ$ is the empty set, whereas $x_\circ \vdash_{r_{ins}} y_\circ$ is not, which leads to a contradiction. Therefore, $uv \in Fact(x)$ or $uv \notin Fact(x_\circ)$.

Case 2: Assume that the linear word x has the factor uv, i.e., x can be written in the form of $x = x_1uvx_2$. By Lemma 3, $(x \vdash_{r_{ins}} y)_\circ \subseteq x_\circ \vdash_{r_{ins}} y_\circ$. Conversely, each word of $x_\circ \vdash_{r_{ins}} y_\circ$ arises from an occurrence of uv in $x_\circ$ and some $\gamma \in \beta_\circ$. If the number of occurrences of uv in x is maximal as it is described, then the result is $y_\circ = (u\gamma vx_2x_1)_\circ$ for each conjugate of $x_\circ$ having an occurrence of uv as a prefix, whose words belong to $(x \vdash_{r_{ins}} y)_\circ$. This leads to the equality $(x \vdash_{r_{ins}} y)_\circ = x_\circ \vdash_{r_{ins}} y_\circ$.

On the other hand, if $uv \notin Fact(x_\circ)$, then both $(x \vdash_{r_{ins}} y)_\circ$ and $x_\circ \vdash_{r_{ins}} y_\circ$ are the empty sets. Hence, the equality holds. □

Notice that the condition $uv \in Fact(x)$ itself is not sufficient for the set equality of the previous theorem since $x_\circ$ can also contain occurrence(s) that run(s) across the junction between the last and first letters of x, as we show in the next example.

Example 14. *Let* $r_{ins} = (a,\lambda/(0)_\circ,b)$ *and* $x = babca$*. In* x*, the factor* ab *occurs only once, giving* $ba0bca$*. So* $(x \vdash_{r_{ins}} y)_\circ = (ba0bca)_\circ$*. In* $x_\circ$*,* ab *also occurs across the junction (last* a*, first* b*), giving the extra circular word* $(babca0)_\circ$*. Since* $ba0bca$ *is not a conjugate of* $babca0$*, then* $(x \vdash_{r_{ins}} y)_\circ$ *and* $x_\circ \vdash_{r_{ins}} y_\circ$ *differ even though* $uv \in Fact(x)$.

The following results relate the circular insertion to Păun circular splicing operations and the one-wheel concatenation.

Theorem 7. *Let Σ be a finite alphabet, and $r_{ins} = (u,\lambda/\beta_\circ,v)$ be a circular insertion rule, where $u, v \in \Sigma^*$ and $\beta \in \Sigma^+$. Then, the circular insertion by r_{ins} can be simulated by a Păun circular splicing operation.*

Proof. Consider the Păun rule $r = (u,v;\lambda,\lambda)$. Let $x_\circ = (uvx_1)_\circ$ (for some $x_1 \in \Sigma^*$ if any) and $y_\circ = \beta_\circ$. Applying r to $x_\circ$ and $y_\circ$, the Păun operation linearizes the words to uvx_1 (for any occurrence of uv in $x_\circ$) and to any conjugate γ of $\beta_\circ$, concatenates them, and circularizes the result, yielding $(u\gamma vx_1)_\circ$, whose words are exactly the results of applying the circular insertion rule r_{ins} to $x_\circ$. □

Example 15. *Let $r_{ins} = (a,\lambda/(01)_\circ,b)$ and $x_\circ = (abc)_\circ$. The circular insertion produces $(a01bc)_\circ$ and $(a10bc)_\circ$. The Păun rule $(a,b;\lambda,\lambda)$ linearizes $x_\circ$ to bca and $(01)_\circ$ to 01 or 10, producing $(bca01)_\circ = (a01bc)_\circ$ and $(bca10)_\circ = (a10bc)_\circ$,* i.e., *the same set.*

Theorem 8. *Let Σ be a finite alphabet, and $r_{ins} = (\lambda,\lambda/\beta_\circ,\lambda)$ be a circular insertion rule, $\beta \in \Sigma^+$. Consider the circular word $x_\circ$. Then, the set of circular words obtained by applying the circular insertion rule r_{ins} to $x_\circ$ coincides with the one-wheel concatenation $x_\circ \odot \beta_\circ$.*

Proof. Let $y_\circ$ be obtained from $x_\circ$ by r_{ins}. Since λ appears anywhere in x as a subword, any conjugate x_1 of x can be taken as a linearization, i.e., any point can be used to cut the circular word $x_\circ$. By the definition of circular insertion, there exists a conjugate $\gamma \in \beta_\circ$ such that $y_\circ = (\gamma x_1)_\circ$. Hence, $y_\circ \subset x_\circ \odot \beta_\circ$.

For the other inclusion, let $y \in x_\circ \odot \beta_\circ$. Then, there exist conjugates $x_1 \in x_\circ$ and $\gamma \in \beta_\circ$ and a circular word $y'_\circ = (x_1\gamma)_\circ$ such that $y \in y'_\circ$. Thus, actually, y is a conjugate of a concatenation of the form $x_1\gamma$. Applying the insertion rule r_{ins} to $x_\circ$, as the empty word is used as context, may cut the cycle of $x_\circ$ to get the conjugate x_1. The insertion of $\gamma \in \beta_\circ$ at this point yields exactly to $(x_1\gamma)_\circ = y'_\circ$. Thus, as $y \in y'_\circ$, it is obtainable by this circular insertion. This completes the second direction of the proof. □

Example 16. *Let $r_{ins} = (\lambda,\lambda/(01)_\circ,\lambda)$ and $x_\circ = (ab)_\circ$. The circular insertion produces the four distinct circular words $(ab01)_\circ$, $(ab10)_\circ$, $(ba01)_\circ$ and $(ba10)_\circ$, which form exactly $x_\circ \odot (01)_\circ$.*

4.2. Circular Deletion

For the sake of completeness, we also consider the deletion operation on circular words. Symmetrically to Definition 7, we define it as follows:

Definition 8 (Circular Deletion from a Circular Word). *Let Σ be a finite alphabet. A circular deletion rule is represented by a triple of the form $r_{del} = (u,\alpha_\circ/\lambda,v)$, where $u, v \in \Sigma^*$ and $\alpha \in \Sigma^+$. Then, having a circular word $x_\circ = (u\gamma vx_1)_\circ$ with $\gamma \in \alpha_\circ$, the application of the rule deletes the occurrence of γ between u and v, obtaining $y_\circ = (uvx_1)_\circ$. We denote this operation by:*

$$x_\circ \vdash_{r_{del}} y_\circ = \{z \in (uvx_1)_\circ \mid x = u\gamma vx_1,\ \gamma \in \alpha_\circ, x_1 \in \Sigma^*\} \tag{7}$$

As in the case of insertion, the result may be the empty set (if no conjugate of x contains $u\gamma v$ as a factor for any $\gamma \in \alpha_\circ$), and it may also contain the circular words obtained from several occurrences. We first make precise why this operation cannot be equivalent to any of the concatenation operations.

Both concatenation operations are length-increasing, whereas the deletion operation is strictly length-decreasing. Hence, no equivalence of the type given in Theorem 2, Corollary 1 and Theorems 4 and 8 can hold for deletion. Nevertheless, the deletion operation is

closely related to the one-wheel concatenation, namely, as its inverse relation, as the next result shows.

Theorem 9. *Let* Σ *be a finite alphabet, and* $r_{del} = (\lambda,\alpha_\circ/\lambda,\lambda)$ *be a circular deletion rule,* $\alpha \in \Sigma^+$. *Then, for any circular words* $x_\circ$ *and* $y_\circ$,

$$x_\circ \vdash_{r_{del}} y_\circ \text{ if and only if } x \in y_\circ \odot \alpha_\circ \tag{8}$$

Proof. Assume first that $y_\circ$ is obtained from $x_\circ$ by r_{del}. Since the contexts are empty, by Definition 8 there are a conjugate $\gamma \in \alpha_\circ$ and a word $x_1 \in \Sigma^*$ such that $x_\circ = (\gamma x_1)_\circ$ and $y_\circ = (x_1)_\circ$. As $x_1 \in y_\circ$ and $\gamma \in \alpha_\circ$, the word $x_1\gamma$ belongs to the set described in Definition 5, and x, being a conjugate of γx_1, is also a conjugate of $x_1\gamma$. Hence $x \in y_\circ \odot \alpha_\circ$.

Conversely, assume that $x \in y_\circ \odot \alpha_\circ$. By Definition 5, x is a conjugate of a word of the form $y_1\gamma$ with $y_1 \in y_\circ$ and $\gamma \in \alpha_\circ$. Therefore $x_\circ = (y_1\gamma)_\circ = (\gamma y_1)_\circ$, i.e., $x_\circ$ has an occurrence of the conjugate γ of $\alpha_\circ$. As the contexts of r_{del} are empty, the rule can be applied at this occurrence, and it deletes γ, yielding $(y_1)_\circ = y_\circ$. □

Theorem 9 is the exact counterpart of Theorem 8: while the circular insertion with empty contexts realizes the one-wheel concatenation, the circular deletion with empty contexts realizes its inverse relation, i.e., a kind of one-wheel quotient by $\alpha_\circ$. Finally, we show that circular deletion can also be related to the splicing operations.

Theorem 10. *Let* Σ *be a finite alphabet, and* $r_{del} = (u,\alpha_\circ/\lambda,v)$ *be a circular deletion rule,* $\alpha \in \Sigma^+$. *Then, for each* $\gamma \in \alpha_\circ$, *the Pixton circular splicing operation defined by the rule pair* $r = (u\gamma v,\lambda;uv)$ *and* $\bar{r} = (\lambda, u\gamma v;\lambda)$, *applied to* $x_\circ$ *and to* $\lambda_\circ$, *produces exactly those circular words that are obtained by* r_{del} *applying it to* $x_\circ$ *using the occurrences of* $u\gamma v$. *Hence, the finite set of rule pairs* $(r, \bar{r})$ *simulates* r_{del}.

Proof. Let $\gamma \in \alpha_\circ$ be fixed. In the notation of Definition 3, the rule pair $r = (\alpha,\alpha';\beta)$ with $\alpha = u\gamma v$, $\alpha' = \lambda$, $\beta = uv$ and $\bar{r} = (\alpha', \alpha; \beta\prime)$ with $\beta' = \lambda$ is applicable to $x_\circ$ and $\lambda_\circ$ if and only if $x_\circ = (u\gamma v\varepsilon)_\circ$ for some $\varepsilon \in \Sigma^*$, i.e., if and only if x has a conjugate of the form $u\gamma v\varepsilon$; the second circular word is linearized as $\varepsilon' = \lambda$. In this case, the operation linearizes $x_\circ$ to $\varepsilon u\gamma v$, replaces the suffix $u\gamma v$ by $\beta = uv$ and the suffix $\alpha' = \lambda$ by $\beta' = \lambda$, concatenates the obtained words and circularizes the result, producing $(\varepsilon uv)_\circ = (uv\varepsilon)_\circ$.

On the other hand, by Definition 8, the rule r_{del} is applicable to $x_\circ$ at an occurrence of $u\gamma v$ if and only if $x_\circ = (u\gamma v\varepsilon)_\circ$ for some $\varepsilon \in \Sigma^*$ (writing ε for x_1), and then it produces $(uv\varepsilon)_\circ$. Thus, the applications of the pair $(r, \bar{r})$ to $x_\circ$ and $\lambda_\circ$ and the applications of r_{del} at the occurrences of $u\gamma v$ correspond to each other and give the same circular words. Taking the union for all $\gamma \in \alpha_\circ$, which is a finite set, the whole result of r_{del} is obtained. □

Note that similar results as Theorem 10 cannot be stated for the Head or the Păun circular splicing operations: as we have already recalled, these operations are length-increasing, that is, actually, the length of the result equals the sum of the lengths of the two input circular words, and therefore they can never produce a shorter circular word. In contrast, Pixton's definition allows rewriting, which makes the simulation of deletion possible.

Example 17. *Let* $\Sigma = \{a,b,0,1\}$, $r_{ins} = (\lambda,(01)_\circ/\lambda,\lambda)$ *and* $x_\circ = (ab01)_\circ$. *The only occurrence of a conjugate of* $(01)_\circ$ *in* $x_\circ$ *is* 01, *hence* $x_\circ \vdash_{r_{del}} y_\circ = (ab)_\circ$. *Accordingly,* $x \in (ab)_\circ \odot (01)_\circ$, *in agreement with Theorem* 9. *Further, by Theorem* 10, *the Pixton rule pair* $r = (01,\lambda;\lambda)$, $\bar{r} = (\lambda, 01; \lambda)$ *applied to* $x_\circ$ *and* $\lambda_\circ$ *linearizes* $x_\circ$ *to* $ab01$, *replaces the suffix* 01 *by* λ, *and produces* $(ab)_\circ$, i.e., *the same circular word.*

https://doi.org/10.3390/e28101113

5. Implications for Language Generation

The results of the previous sections concern single applications of the operations. Since the operations are used in systems that generate circular languages by iterating them, we indicate here what our equivalences imply at that level. Throughout, a circular language L is called regular if its linearization, i.e., the set of all the linear words belonging to its circular words, is a regular language.

Corollary 2. *Let $L_1$and L_2be regular circular languages over Σ, and let $r = (u_1, u_2; u_3, u_4)$ be a Păun circular splicing rule. Then the set of circular words obtained by applying r to the pairs of $L_1 \times L_2$ is a regular circular language, and it is contained in $L_1 \odot L_2$ which is itself regular and is obtained for $r = (\lambda,\lambda;\lambda,\lambda)$.*

Proof. Let $Lin(L_1)$ and $Lin(L_2)$ denote the linearizations, which are regular by assumption. By Definition 2, the rule r uses exactly the linearizations belonging to $Lin(L_1) \cap u_2\Sigma^* u_1$ and to $Lin(L_2) \cap u_4\Sigma^* u_3$, which are regular, and it concatenates them, so the linear words obtained produce a regular language; its closure under conjugation is regular as well, i.e., the obtained set is a regular circular language. The inclusion in $L_1 \odot L_2$ and the equality for the rule with empty contexts are the two parts of Theorem 2, applied to each pair of circular words, while the regularity of $L_1 \odot L_2$ was proved in [17]. □

Thus, one application of a Păun circular splicing rule (and hence, by Proposition 1, of the corresponding Head rule) cannot leave the class of regular circular languages, and, by Corollary 1, the largest set it can produce, $L_1 \odot L_2$, is already reached by a single rule with empty contexts. Consequently, the generative power of the corresponding systems does not come from the operation itself, but from the iteration and from the restriction imposed by the sites: the contexts select, at each step, a subset of a set that is fixed in advance by the inputs.

At the level of single operations, Theorems 7 and 10 relate insertion and deletion to splicing: one application of a circular insertion rule can be performed by one application of a Păun circular splicing rule, provided that the inserted circular word is available as an input, and one application of a circular deletion rule can be performed by Pixton circular splicing rule pairs applied together with the empty circular word $\lambda_\circ$. Whether, and how, these relations extend to the languages obtained by iterating the operations is not addressed here; for insertion, this is the first of the tasks listed in the Conclusions. We note also that the operations behave differently with respect to length, and hence with respect to the descriptional resources needed: Head's and Păun's operations are length-increasing, so iterating them on a finite set of initial circular words yields only circular words whose lengths are sums of the lengths of the initial ones, while Pixton's operation permits rewriting and, by Theorem 10, deletion, so that this restriction disappears.

Finally, our equivalences have a descriptional reading. By Proposition 1, the translations between Head triples and Păun rules change the size of a rule only by the length of the common middle word x, so the two formalisms are equally succinct up to an additive term. By Theorem 10, one circular deletion rule is simulated by at most $\|\alpha_\circ\| \leq |\alpha|$ Pixton rule pairs, a linear increase whose optimality is open. In the opposite direction, Theorem 1 shows that nine different rules may have exactly the same effect, so the sets of rules describing a given operation are far from unique, and the most succinct description, a single rule with empty contexts, is precisely the one producing the largest set of circular words.

6. Conclusions

The results of this paper give an algebraic reading of the biologically inspired operations defined on circular words. When the contexts prescribed by the rules are empty, circular splicing and circular insertion degenerate to a purely algebraic operation, the one-wheel concatenation of circular words; when they are not empty, they only select which linearizations of the input circular words, i.e., which cut points of the cyclic molecules, may be used, and the results obtained in this way always belong to the same one-wheel concatenation. This dependence on the contexts is generally monotone: prescribing longer sites yields fewer products, the rule with empty contexts being the maximum of this ordering and the union over all rules recovering the whole concatenation. In this sense, the one-wheel concatenation is the common algebraic core of all the operations considered here, and the contexts of the rules carry their whole computational content. The coincidence with the two-wheel concatenation is, in contrast, exceptional: it occurs only in the degenerate unary and empty cases.

As single operations, those of Head and of Păun are equivalent, and both are length-increasing; Pixton's definition, which permits rewriting, is the only one that is able to shorten the input, and it is therefore also the only one that is able to simulate deletions. Together with our generalization of the insertion and of the deletion operations to act on circular words and to produce circular words, this shows that insertion and deletion are not independent of splicing at the level of single operations: insertion is the contextual form of the one-wheel concatenation, while deletion is its inverse relation. One application of a Păun circular splicing rule to regular circular languages yields a regular circular language; the corresponding questions for the other operations are left open. We believe that our results help to study also iterated applications of the mentioned operations.

These operations have recognizable biological counterparts, and it is worth stating how far the correspondence goes. The insertion of a circular word into a circular word models the integration of a cyclic molecule, such as a plasmid, a circular viral genome or a circular transposition intermediate, into another cyclic molecule, while the deletion models the reverse process, the excision of such a segment; that deletion is the inverse relation of the one-wheel concatenation and is the formal counterpart of the integration–excision duality. In this picture, the contexts u and v of a rule represent the sites at which the enzymes act, and they are the realistic part of the model, since site-specific recombination and the action of restriction enzymes are determined precisely by such sites.

The limitations of the model should also be stressed. As usual in this branch of formal language theory, a molecule is represented by a word over a finite alphabet, so the double-stranded structure, the Watson–Crick complementarity and the orientation of the strands are not fully represented, and neither are the topological properties of cyclic DNA, such as supercoiling, knotting or catenation, which are known to influence recombination. Moreover, the operations are non-deterministic, and their results are the sets of all products that a rule may yield, whereas in a real reaction the products obtained depend on concentrations, kinetics and reversibility. Within these limits, the results have a practical reading for DNA computing and synthetic biology, where circular molecules are used as carriers of the computation and where standard assembly methods yield circular products: the one-wheel concatenation describes the complete library of circular molecules that a given pair of inputs may produce, and our characterizations delimit which of these products can be enforced by a suitable choice of the sites, which is exactly what has to be designed in a plasmid-based experiment.

Our results concern single applications of the operations, and the natural continuation is to pass to the systems built on them. We list the specific tasks we intend to address:

https://doi.org/10.3390/e28101113

(1) Define a circular insertion system as a triple (Σ, A, R), where A is a finite set of axiom circular words and R is a finite set of circular insertion rules, and compare the family of circular languages generated by such systems with the family generated by Păun circular splicing systems with finite sets of axioms and rules. Theorem 7 gives one inclusion for a single step, provided that the inserted circular words belong to the axioms; the question is whether the inclusion is strict for the generated families.

(2) Determine the sizes for which circular insertion–deletion systems are computationally complete, i.e., the tuples of parameters bounding the lengths of the inserted and deleted words and of their left and right contexts, in analogy with the known results obtained for insertion–deletion systems on linear words [13,20].

(3) Decide, for given circular words $x_\circ$ and $y_\circ$ and a given $\alpha \in \Sigma^+$, whether $y_\circ$ can be obtained from $x_\circ$ by iterated applications of the circular deletion rule $(\lambda,\alpha_\circ/\lambda,\lambda)$, and determine the complexity of this reachability problem for the one-wheel quotient; the corresponding problem for iterated circular insertion is the reachability problem for the one-wheel concatenation.

(4) Determine the descriptional complexity of the simulation given in Theorem 10: it uses one Pixton rule pair for each conjugate of $\alpha_\circ$, and the question is the minimal number of rule pairs needed to simulate one circular deletion rule, in particular, whether a single pair is always sufficient.

(5) Define an excision operation that cuts one circular word at two sites into two circular words, i.e., a splitting counterpart of the one-wheel concatenation modeling the excision of a cyclic molecule from another one, and decide whether adding it to circular splicing systems strictly increases their generative power.

Beyond these tasks, the classical problems concerning finite circular splicing systems remain open, in particular the characterization of the circular languages, especially the regular ones, generated by them, which is known only for restricted classes of systems, and the comparison of the systems based on the definitions of Head, Păun and Pixton.

Author Contributions: Conceptualization, B.N. and B.A.; methodology, B.N. and B.A.; validation, B.A. and B.N.; formal analysis, B.A. and B.N.; investigation, B.A. and B.N.; writing—original draft preparation, B.A.; writing—review and editing, B.N.; visualization, B.A.; supervision, B.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been funded by BAP-C 04-26-01 of Eastern Mediterranean University.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. McManus, S.A.; Li, Y. The structural diversity of deoxyribozymes. *Molecules* **2010**, *15*, 6269–6284. [CrossRef] [PubMed]
2. Hollenstein, M. DNA catalysis: The chemical repertoire of DNAzymes. *Molecules* **2015**, *20*, 20777–20804. [CrossRef] [PubMed]
3. Zhang, Y.; Ji, Z.; Wang, X.; Cao, Y.; Pan, H. Single-molecule study of DNAzyme reveals its intrinsic conformational dynamics. *Int. J. Mol. Sci.* **2023**, *24*, 1212. [CrossRef] [PubMed]
4. Bonizzoni, P.; De Felice, C.; Mauri, G.; Zizza, R. Circular splicing and regularity. *RAIRO—Theor. Inform. Appl.* **2004**, *38*, 189–228. [CrossRef]
5. Bonizzoni, P.; De Felice, C.; Mauri, G.; Zizza, R. DNA and circular splicing. In *DNA Computing. DNA 2000*; Lecture Notes in Computer Science; Condon, A., Rozenberg, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2054, pp. 117–129. [CrossRef] [PubMed]
6. Bonizzoni, P.; De Felice, C.; Mauri, G.; Zizza, R. On the power of circular splicing. *Discret. Appl. Math.* **2005**, *150*, 51–66. [CrossRef]
7. De Felice, C.; Bonizzoni, P.; Zizza, R. A characterization of regular circular languages generated by circular splicing systems. *Theor. Comput. Sci.* **2009**, *410*, 3382–3390. [CrossRef]

8. Head, T. Formal language theory and DNA: An analysis of the generative capacity of specific recombinant behaviours. *Bull. Math. Biol.* **1987**, *49*, 737–759. [CrossRef]
9. Head, T. Splicing schemes and DNA. In *Lindenmayer Systems: Impacts on Theoretical Computer Science and Developmental Biology*; Rozenberg, G., Ed.; Springer: Berlin/Heidelberg, Germany, 1992; pp. 371–383.
10. Păun, G. On the splicing operation. *Discret. Appl. Math.* **1996**, *70*, 57–79. [CrossRef]
11. Păun, G.; Rozenberg, G.; Salomaa, A. *DNA Computing—New Computing Paradigms*; Springer: Berlin/Heidelberg, Germany, 1998. [CrossRef]
12. Pixton, D. Regularity of splicing languages. *Discret. Appl. Math.* **1996**, *69*, 101–124. [CrossRef]
13. Kari, L.; Seki, S.; Sosík, P. DNA computing—Foundations and implications. In *Handbook of Natural Computing*; Rozenberg, G., Bäck, T., Kok, J.N., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 1073–1127. [CrossRef]
14. Kari, L.; Thierrin, G. Contextual insertions/deletions and computability. *Inf. Comput.* **1996**, *131*, 47–61. [CrossRef]
15. Daley, M.; Kari, L.; Gloor, G.; Siromoney, R. Circular contextual insertions/deletions with applications to biomolecular computation. In *Proceedings of the 6th International Symposium on String Processing and Information Retrieval (SPIRE '99) and the 5th International Workshop on Groupware*; IEEE Computer Society: Los Alamitos, CA, USA, 1999; pp. 47–54.
16. Kudlek, M. On Languages of Cyclic Words. In *Aspects of Molecular Computing; Lecture Notes in Computer Science*; Jonoska, N., Păun, G., Rozenberg, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2950, pp. 278–288.
17. Abdallah, B.; Nagy, B. On concatenations of regular circular word languages. *Mathematics* **2025**, *13*, 763. [CrossRef]
18. Bonizzoni, P.; De Felice, C.; Zizza, R. A characterization of (regular) circular languages generated by monotone complete splicing systems. *Theor. Comput. Sci.* **2010**, *411*, 4149–4161. [CrossRef]
19. Boasson, L.; Bonizzoni, P.; De Felice, C.; Fagnot, I.; Fici, G.; Zaccagnino, R.; Zizza, R. Splicing systems from past to future: Old and new challenges. In *Discrete Mathematics and Computer Science: In Memoriam Alexandru Mateescu (1952–2005)*; Păun, G., Rozenberg, G., Salomaa, A., Eds.; The Publishing House of the Romanian Academy: Bucharest, Romania, 2014; pp. 51–76.
20. Păun, G. DNA computing by splicing and by insertion–deletion. In *Handbook of Natural Computing*; Rozenberg, G., Bäck, T., Kok, J.N., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 1185–1202. [CrossRef]
21. Rozenberg, G.; Salomaa, A. (Eds.) *Handbook of Formal Languages*; Springer: Berlin/Heidelberg, Germany, 1996.
22. Head, T.; Păun, G.; Pixton, D. Language theory and molecular genetics: Generative mechanisms suggested by DNA recombination. In *Handbook of Formal Languages*; Rozenberg, G., Salomaa, A., Eds.; Springer: Berlin/Heidelberg, Germany, 1996; Volume 2, pp. 295–360.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.