



## Languages of Partial Words How to Obtain Them and What Properties They Have

PETER LEUPOLD

Research Group on Mathematical Linguistics

Rovira i Virgili University

Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain

E-mail: klauspeter.leupold@estudiants.urv.es

### Abstract

Partial words were introduced by Berstel and Boasson in 1999. Since then various of their combinatorial properties have been studied, the foremost being periodicity. Lately Blanchet-Sadri has made a first step in also investigating languages of partial words by introducing the concept of *pcodes*. With a slightly different approach we define new ways to obtain such languages by puncturing conventional ones. Then we present some first results on properties like the finiteness of their root or the property of being a code for these newly defined punctured languages.

## 1 Introduction

A partial word is a word with some positions, of which it is not known which letter they hold. With some motivation from molecular biology, partial words were introduced by Berstel and Boasson in 1999 [1]. In their seminal, and also in most subsequent work, mainly periodicity and a few other combinatorial properties such as conjugacy and primitivity were investigated (see e.g. [3], [5], [6] and [16]).

Quite recently, Blanchet-Sadri [4] has made a first step towards also investigating languages of partial words by introducing the concept of *pcodes*, sets of partial words fulfilling a code-like property. We now take another step and consider languages of partial words in general. A first topic is, how to obtain such a language – we contrast allowing known mechanism to generate holes against subsequent puncturing of an already generated language.

Then we investigate some combinatorial properties of such languages, which are analogous to well-known properties for conventional languages. The first is the finiteness of the root. Here, things turn out to be rather different from the total word case, because already the definition of the root of a word or a language has to be modified to apply to partial words.

An even more intensively investigated property of languages is that of being a code. We define punctured codes in such a way that they are conventional codes, which preserve the uniqueness of factorization, even if the codewords are riddled with holes to a certain degree. Then some fundamental properties of such punctured codes are presented.

## 2 Preliminaries

Before actually turning our attention to partial words, we recall some basic and important notations concerning (conventional) words. The latter we will also call total words, if we want to emphasize that a word is not partial. In general, however, *word* without any specification shall mean a total word.

In a second section, we provide some motivation from molecular biology for the investigation of partial words and subsequently we formally define this notion along with several other new concepts connected with it.

### 2.1 Words

When we speak of a *word*, we mean a sequence of letters taken from a finite, non-empty *alphabet*  $\Sigma$ . The free monoid generated by this alphabet with concatenation is  $\Sigma^*$ , and  $\lambda$ , the *empty word*, is its neutral element. The *length* of a word  $w \in \Sigma^*$ , i.e. the number of its letters, is denoted by  $|w|$ . For counting the number of occurrences of letters from a subset  $B$  of  $\Sigma$ , we use the notation  $|w|_B$ . If we want to refer to one specific letter at the  $i$ -th position, we write  $w[i]$  and also denote longer factors in an analogous way. The first letter of a word has number zero.

If  $u = w[0 \dots k]$  for some  $k < |w|$ , we say that  $u$  is a *prefix* of  $w$  and write  $u \sqsubseteq_p w$ ; the prefix is *proper*, if  $u \neq w$ . More generally,  $u$  is a *factor* of  $w$  (written  $u \sqsubseteq w$ ), if there are  $i \leq j < |w|$  such that  $u = w[i \dots j]$ ; the factor is *proper*, if  $i \neq 0$  or  $j \neq |w| - 1$ . It is *internal*, if both  $i \neq 0$  and  $j \neq |w| - 1$ . If  $j = |w| - 1$ , then  $u$  is a *suffix* of  $w$ , written  $u \sqsubseteq_s w$ .

A *factorization* of a word  $w$  is a sequence of words  $w_1, w_2, \dots, w_n$  such that  $w = w_1 w_2 \dots w_n$ . If the factors are all taken from a set of words  $W$ , then such a factorization is called a  $W$ -factorization.

A less common way to look at a word  $w$  is to see it as a total function from  $\{0, \dots, |w| - 1\}$  into  $\Sigma$ , but obviously this gives us the same notion as above. We only mention this, because the definition of a partial word will be stated in comparable terms. Also this further motivates our notation  $w[i]$  from above as well as the term *total* for non-partial words.

A word is *primitive*, if it is not a non-trivial (i.e. with exponent one) power of any word. Thus  $u$  is primitive, if  $u = v^k$  implies  $u = v$  and  $k = 1$ ; this means that  $\lambda$  is not primitive, because, for example,  $\lambda^4 = \lambda$ . For every non-empty word  $w$  there exists a unique primitive word  $p$  such that  $w \in p^+$ ; this primitive word is called the *root* of  $w$  and we will denote it by  $\sqrt{w}$ . In a canonical way, we extend this notion to languages, such that  $\sqrt{L} := \{\sqrt{w} : w \in L\}$ .

Although they will be used later on, we refrain here from giving further definitions for notions like regular or context-free languages from classical Formal Language Theory. If the reader encounters such notions unknown to him, we refer him to classical textbooks like the ones by Harrison [8] or Rozenberg and Salomaa [15] for the definitions not provided here.

## 2.2 Partial Words and Their Motivation

The main motivation for the introduction of partial words mentioned by Berstel and Boasson [1] came from the molecular biology of nucleic acids. There, among other things, one tries to determine properties of the DNA or RNA sequences encountered in the genome of organisms on Earth. These are usually seen as strings over the alphabet  $\{A, T, C, G\}$  respectively  $\{A, U, C, G\}$  of four bases. In nature they mostly occur paired with their Watson-Crick complements; all these concepts belong more to the realm of biology and will not be explained in any depth here.

But supposing a perfect pairing of bases is supposing an ideal world. In reality, the complement is often not completely perfect. For example, copying operations on single strands do not work without errors and thus result in mismatches. However, as long as the number of such mismatches is not very high, the strands will still align due to the affinity of the matching parts.

If one then encounters a pair of strands as depicted in Figure 1, there is in general no telling, which one of the mismatched bases is the correct or original one. However, one still wants to investigate the properties of such a sequence and state them as concisely as possible. To this end, it seems a plausible choice to regard the positions

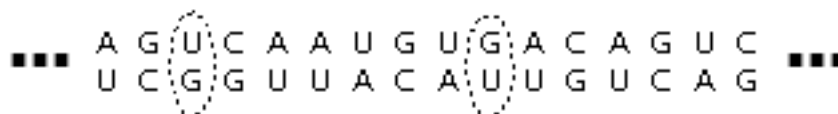


Figure 1: Part of an RNA sequence with two mismatches.

in question as unknown, or holes, and to see what then still can be said about such a sequence.

In the given example we would consider (for the upper strand) a string composed of the parts  $\dots AG, CAAUGU,$  and  $ACAGUC \dots$  in this order with one hole inbetween each of the parts.

Thus, intuitively, a partial word is very much like a conventional word, only at some positions we do not know, which letter it has. Coming back to the definition of a word as a total function from  $\{0, \dots, |w| - 1\}$  to  $\Sigma$ , we then except these unknown positions from the mapping's domain and define a *partial word*  $w$  as a partial function from  $\{0, \dots, |w| - 1\}$  to  $\Sigma$ . The positions, where  $w[n]$  is not defined for  $n < |w|$  are called the word's *holes*. The numbers in  $\{0, \dots, |w| - 1\} \setminus D(w)$  are the set of holes of  $w$  and are written  $\text{Hole}(w)$ . Here  $D(w)$  denotes the domain of  $w$ .

For a partial word  $w$  we define its *companion* as the total word  $w_\diamond$  over the extended alphabet  $\Sigma \cup \{\diamond\}$  where

$$w_\diamond[i] := \begin{cases} w[i] & \text{if } i \in D(w) \\ \diamond & \text{if } i \notin D(w) \wedge 0 \leq i < |w| \end{cases} .$$

When it is more convenient, we will also refer to the companion as a partial word to simplify the syntax of our sentences. Thus we will say for example “the partial word  $\diamond a \diamond b$ ” instead of “the partial word with companion  $\diamond a \diamond b$ ”.

For two partial words  $u$  and  $v$  of equal length, we say that  $u$  is contained in  $v$ , if  $D(u) \subset D(v)$  and  $i \in D(u) \rightarrow u[i] = v[i]$ ; this is written  $u \subset v$ , a rather natural notation, if we adopt the view of a function  $f$  as a set of ordered pairs  $[n, f(n)]$ . If there exists a partial word  $w$  such that for two other partial words  $u$  and  $v$  we have  $u \subset w$  and  $v \subset w$ , then  $u$  and  $v$  are called *compatible*, written  $u \uparrow v$ . For two such words,  $u \vee v$  is the smallest word containing both  $u$  and  $v$ ; smallest here means that its domain is  $D(u \vee v) = D(u) \cup D(v)$ , its values are defined in the obvious way.

**Example 2.1** *Illustrating the above definitions, we see that for  $u = b \diamond a b a \diamond a$  and  $v = \diamond a a \diamond a \diamond a$  we have the compatibility  $u \uparrow v$ . Further,  $u \vee v = b a a b a \diamond a$ .*

The following rules for computing with partial words already stated by Berstel and Boasson [1] are rather obvious and can be given without any proof:

- Multiplication: if  $u_1 \uparrow v_1$  and  $u_2 \uparrow v_2$ , then  $u_1 u_2 \uparrow v_1 v_2$ ,
- Simplification: if  $u_1 u_2 \uparrow v_1 v_2$  and  $|u_1| = |v_1|$ , then  $u_1 \uparrow v_1$  and  $u_2 \uparrow v_2$ ,
- Weakening: if  $u \uparrow v$  and  $w \subset u$ , then  $w \uparrow v$ ,

From the simplification rule, one can quite directly deduce an analogue to Levi’s Lemma. This also was done first by Berstel and Boasson [1].

**Lemma 2.1 ([1])** *For partial words  $r, s, u, v$  such that  $rs \subset uv$  and  $|r| \leq |u|$ , there exists a factorisation  $u = u_1 u_2$  such that  $r \uparrow u_1$  and  $s \uparrow u_2 v$ .*

At times it will be interesting to in some sense measure to what degree a partial word is riddled with holes. For example a nucleic acid sequence would certainly not align with its complement any more, if more than half of its bases had been changed. To formally denote the degree to which a partial word  $u$  is undefined we will use the *puncturedness coefficient* defined as  $\vartheta(u) := \frac{|Hole(u)|}{|u|}$ .

Finally we shall remark that partial words over a one-letter alphabet do not make much sense, because any hole could just stand for the unique letter. Therefore in what follows we shall always assume that the alphabets under consideration have at least two distinct letters  $a$  and  $b$ .

### 3 Generating Holes vs. Puncturing

For total words, even more interest than on combinatorial properties of single words has been paid to the properties of languages of such words. As, for example, any DNA computation produces a language of all sequences involved in its process, naturally in many cases such an entire language rather than just the initial sequences will be required to fulfill certain combinatorial properties [13]. Thus, after the motivation we have provided for partial words in general, it now is a systematic next step, to investigate such properties for languages of partial words, too.

To do so, one has first to find a way to define languages of partial words appropriately. One possibility would be to use classical generative devices, like generative grammars, where holes can be generated as normal non-terminals. Generating something, which one does not know of, what it is, however, seems somewhat artificial.

As described in Section 2.2, the main motivation for partial words is that in certain contexts wear and tear can make a letter in a previously total word unreadable. Therefore we introduce the concept of a *punctured language* as follows: such a language is defined in a classical way, e.g. by a grammar or an accepting device. The holes are put into its words only in a second step of the definition. Again looking back to the original motivation for partial words this might correspond to looking at a language of DNA sequences after a while, when transcriptions or simply time might have done some damage and introduced mismatches corresponding to our holes.

When there are no restrictions on the number of holes, we will call such a language a *punctured language*. If there is a bound of  $k$  holes per word, it will be called *k-punctured*, and similarly if the puncturedness coefficient for all words is bounded by a positive rational number  $q < 1$ , it will be called *q-punctured*.

**Definition 3.1** *Formally, the k-puncturing of a word w is*

$$w^{(k-\diamond)} := \{u : u \uparrow w \wedge \vartheta(u) \leq k\}.$$

*Then the k-puncturing of a language L is*

$$L^{(k-\diamond)} := \bigcup_{w \in L} w^{(k-\diamond)}.$$

*For classes of languages we use an analogous notation, denoting by  $\mathcal{C}^{(k-\diamond)}$  the class of all k-punctured languages obtained by puncturing languages from a class  $\mathcal{C}$  of languages.*

Another possibility is generating the holes as a special type of non-terminal. Thus the degree of puncturedness is not enforced by any external control, but must be intrinsic in the grammar's way of deriving. With  $REG_{(k-\diamond)}$ ,  $CF_{(k-\diamond)}$ ,  $CS_{(k-\diamond)}$ , and  $RE_{(k-\diamond)}$  we denote the classes generated by regular, context-free, context-sensitive, and general grammars with  $\diamond$  as a non-terminal respectively. Here a constant bound on the number of holes can be kept by all generative grammars relatively easily. A bound relative to the length is in general rather difficult to obey for regular and context-free grammars.

We now provide an example for the generation of a language of partial words, which in this case coincides with a punctured language. Then we state some relations between the classes of languages obtained by the two mechanisms introduced in this section.

**Example 3.1** *We write a grammar generating the 2-puncturing of the language generated by  $[\{S\}, \{a, b\}, S, \{S \rightarrow SS, S \rightarrow aSb, S \rightarrow ab\}]$ . The rules of the new grammar are very much the same, only in an index the non-terminal carries the number of holes that can still be generated in its subtree of the derivation. Thus the start symbol is  $S_2$  now.*

*Then there are the branching rules  $S_2 \rightarrow S_2S_0$ ,  $S_2 \rightarrow S_1S_1$ , and  $S_2 \rightarrow S_0S_2$ , which allow all possible distributions of the two holes to the subtrees. Further, terminals and holes are generated by  $S_2 \rightarrow aS_2b \mid \diamond S_1b \mid aS_1\diamond \mid \diamond S_0\diamond$  and  $S_2 \rightarrow ab \mid \diamond b \mid a\diamond \mid \diamond\diamond$ . For  $S_1$  analogous rules just not generating two holes in one step are added,  $S_0$  behaves just like the original  $S$ .*

**Proposition 3.1** *For  $\mathcal{C} \in \{REG, CF, CS, RE\}$  the class  $\mathcal{C}^{(k-\diamond)}$  is properly contained in  $\mathcal{C}_{(k-\diamond)}$  for any positive integer  $k$ .*

*Proof.* [Proof] All non-empty languages from the classes  $\mathcal{C}^{(k-\diamond)}$  contain also total words; it is obvious how to construct grammars generating exclusively partial words. Therefore in none of the cases equality holds.

Example 3.1 shows exemplarily, how to construct for any regular or context-free language from its grammar a new one, which generates the original language's  $k$ -puncturing. Of course, linear bounded automata and Turing machines can do the same thing for context-sensitive and recursively enumerable languages, and thus the inclusions hold.  $\square$

Concerning closure properties, all classes — punctured or generated, with an absolute or a relative bound on the number of holes — are closed under union and intersection, if their total counterparts are; for total languages, these properties are all well-established [15]. A relevant difference exists between absolute and relative bounds on the number of holes, when we consider the catenation of languages. Here a relative bound is preserved, while an absolute one might be violated. Thus, in contexts where languages are catenated, one should work with relative bounds.

## 4 Languages of Partial Words

Not every question about languages obtains a new meaning in this way, but some properties can very well be formulated. As a first example, we take a look at a topic which was investigated by S. Horvth and G. Lischke together with the current author [9]: the finiteness of the root of a language. For this we will suppose that the alphabet  $\Sigma$  has at least two elements; otherwise all such questions become trivial, because the root of any language containing non-empty words would be the one-element set  $\Sigma$  itself.

To give the central result, we first recall the definition of *Kleene terms*, also known as regular expressions. For an alphabet  $\Sigma$  we denote by  $\Sigma_T$  its extension by the following symbols:  $\Sigma_T := \Sigma \cup \{\emptyset, (, ), \circ, \vee, \langle, \rangle\}$ . All the new symbols are not elements of  $\Sigma$ . The set of Kleene terms over the alphabet  $\Sigma$  is composed as follows and has the following interpretation  $\phi$ , which maps the Kleene terms to languages:

- (i)  $\emptyset$  is a Kleene term, and  $\phi(\emptyset) := \emptyset$ .
- (ii) Every letter of the alphabet  $\Sigma$  is a Kleene term, and  $\phi(a) := \{a\}$ .
- (iii) If  $t_1$  and  $t_2$  are Kleene terms, so is  $(t_1 \circ t_2)$ , and  $\phi(t_1 \circ t_2) := \phi(t_1) \cdot \phi(t_2)$ .
- (iv) If  $t_1$  and  $t_2$  are Kleene terms, so is  $(t_1 \vee t_2)$ , and  $\phi(t_1 \vee t_2) := \phi(t_1) \cup \phi(t_2)$ .
- (v) If  $t$  is a Kleene term, so is  $\langle t \rangle$ , and  $\phi(\langle t \rangle) := \phi(t)^*$ .

There are no other Kleene terms.

Under items (iii) and (iv), strictly speaking  $\phi((t_1 \circ t_2))$  and  $\phi((t_1 \vee t_2))$  should be written; however, we omit this doubling of parentheses for reasons of conciseness. We will also omit other parentheses, unnecessary e.g. for reasons of associativity,

Now we can proceed to define a *root term*. This is a Kleene term having the form

$$((t_{1,1} \circ \langle t_{1,2} \rangle) \vee (t_{2,1} \circ \langle t_{2,2} \rangle) \vee \dots \vee (t_{k,1} \circ \langle t_{k,2} \rangle)) \quad (1)$$

for a natural number  $k$  where each  $t_{i,j}$  is a product of elementary terms, and for every  $i$ , either  $\sqrt{\phi(t_{i,1})} = \sqrt{\phi(t_{i,2})}$  or one of the two roots is empty. Additionally,  $\emptyset$  and  $\langle \emptyset \rangle$  are root terms. After this preparation we are ready to recall the significance of root terms for the finiteness of the roots of regular languages [9].

**Theorem 4.1** *A regular language has finite root, if and only if it can be described by a root term.*

Before looking at an analogous question for languages of partial words, we need to define, what we mean by their roots. While each total word has one word as a root, one partial word already has a set of words as root. We define

$$\sqrt{w} := \{p : p \text{ is primitive and total} \wedge \exists k (w \subset p^k)\}.$$

This leads to some ambiguity for total words, where for example  $abab$  has now root  $\{ab\}$ , where earlier and in the literature the root would have been the word  $ab$ ; however, this difference should not lead to confusion.

Again, generalizing the concept of root, we define for a language  $L$  of partial words its root as  $\sqrt{L} := \bigcup_{w \in L} \sqrt{w}$ , where it turns out to be convenient that also total words have a set as root.

Now, even putting just one hole into the words of such a language can change the finiteness of its root from finite to infinite; thus Theorem 4.1 cannot directly be extended to languages of partial words.

**Example 4.1** Consider the language  $L = \phi(\langle a \rangle \vee \langle b \rangle)$ , which has the finite root  $\{a, b\}$ . If we now put one hole into, for example, the word  $aaaaa$  at position 2, then the word's root is changed from  $a$  to  $\{a, abaaa\}$ ; thus the root of  $L$  with up to one hole per word is the infinite language  $\phi(\langle a \rangle b \langle a \rangle \vee \langle b \rangle a \langle b \rangle)$ .

In the same line of investigations as Theorem 4.1, the current author made a next step and looked to the context-free languages with finite roots [12]. Rather easy considerations show that these never have finite root, if they are not regular.

**Theorem 4.2** All context-free languages with finite root are regular.

This result allowed, after some further considerations, to give a new decision procedure for a problem shown earlier to be decidable by Ito and Horvth [10].

**Theorem 4.3** For a context-free language  $L$  it is decidable, whether its root is finite.

We now investigate the same problems for languages of partial words. With the help of a well-known lemma, we first obtain an auxiliary result. Lemma 4.1 itself is a straightforward consequence of two well known facts: for a word  $w$  and a letter  $a$  at least one of  $w$  and  $wa$  is primitive [17]; primitivity is invariant under cyclic permutation of words [2].

**Lemma 4.1** For total words  $u$  and  $v$  and for two distinct letters  $a$  and  $b$ , at least one of  $uav$  and  $ubv$  is primitive.

It remains to also state the notion of primitivity for partial words.

**Definition 4.1** A non-empty partial word  $w$  is primitive, if and only if  $w \subset u^n$  implies  $n = 1$ .

In the way defined here, primitivity is in some sense inherited upwards along the inclusion relation: if  $u$  is primitive and  $u \subset v$  holds, then also  $v$  is primitive.

**Proposition 4.1** For every word with one hole, its root contains at least two words.

*Proof.* [Proof] Any partial word with one hole has a unique factorization  $u \diamond v$ . Let  $w$  be primitive. Then by the definition of primitivity, both  $uav$  and  $ubv$  must be primitive. Therefore both are contained in  $\sqrt{w}$ .

For a non-primitive  $w$ ,  $\sqrt{w}$  contains a word  $w'$  shorter than  $w$ , again by the definition of primitivity. At the same time, by Lemma 4.1 at least one of  $uav$  and  $ubv$  must be primitive and due to length consideration distinct from  $w'$ . Thus also in this case  $\sqrt{w}$  contains at least two words.  $\square$

With this we are now able to state a result for partial words corresponding to Theorems 4.1 and 4.2 and even exceeding them in scope, because it holds true for any language rather than only regular or context-free ones; as hinted in the example above, the situation here turns out to be quite different from the one for languages of total words.

**Theorem 4.4** *A 1-punctured language's root is finite, if and only if the language itself is finite.*

*Proof.* [Proof] It is obvious that a finite language has a finite root, no matter whether it is punctured or not. If an infinite language  $L$  contains infinitely many primitive words, all these will also be in  $\sqrt{L}$  as well as in the root of  $L$ 's punctured version. On the other hand, every non-primitive word  $u$  in  $L$  will contribute one word of length  $|u|$  to the root of the 1-punctured  $L$ : if the one hole in  $u$  is replaced by a letter different from the original one (before puncturing), the result is in the language's root, because it is primitive according to Lemma 4.1. The observation that a language with infinitely many non-primitive words, has also infinitely many non-primitive words of pairwise unequal length concludes the proof.  $\square$

Because every type of puncturedness allows one hole per word (at least starting from a certain length) we obtain immediately a statement about languages punctured to an arbitrary degree.

**Corollary 4.4.a** *For any integer  $k > 0$  or rational number  $0 < k < 1$ , a  $k$ -punctured language's root is finite, if and only if the language itself is finite.*

With the fact that for context-free languages the question of their finiteness is decidable, we obtain an immediate consequence for their punctured counterparts. For context-sensitive languages, however, finiteness is not a decidable property, which provides us with a second consequence.

**Corollary 4.4.b** *For any punctured context-free language it is decidable, whether its root is finite.*

**Corollary 4.4.c** *For a punctured context-sensitive language it is in general undecidable, whether its root is finite.*

For some other properties of languages, puncturing does not make a big difference. One example is density; a language  $L$  is dense, if for every  $w \in \Sigma^*$  there is a word  $w' \in \Sigma^*w\Sigma^*$  such that  $w' \in L$  — in other words, the set of factors of  $L$  is entire  $\Sigma^*$ . We adapt this a little bit and say that a  $k$ -punctured language  $P$  is dense, if for every  $w \in (\Sigma^*)^{(k-\diamond)}$  there is a partial word  $w' \in (\Sigma^*w\Sigma^*)^{(k-\diamond)}$  such that  $w' \in P$ . Now we obtain immediately an equivalence.

**Proposition 4.2** *A language is dense, if and only if its 1-puncturing is dense.*

*Proof.* [Proof] If  $L$  is dense, then for every word  $w$  with one hole, there are total words  $w', u, v$  such that  $uw'v \in L$  and  $w \subset w'$ . Thus  $uwv \in (\Sigma^*w'\Sigma^*)^{(1-\diamond)}$ , when we put the hole in  $w'$  in the position such that we obtain  $w$ .

Now let  $L^{(1-\diamond)}$  be dense. Here all total words are just a subset of all words fulfilling the density condition, and conventional density is obvious.  $\square$

In a similar manner, boundedness can be shown not to be affected by puncturing.

## 5 Punctured Codes

For sets of total words probably the best-investigated property is that of being a code [2], [17]. Following similar lines for partial words, Blanchet-Sadri [4] has already defined the concept of *pcode*, which is basically a code consisting of arbitrary partial words.

We take a slightly different approach and consider codes in the following way: they must contain all puncturings (up to the respective bound) of the total words involved. This way the language generated by the code contains all the catenations of partial words, which might have resulted from the original code by wear and tear. Different occurrences of one codeword can be punctured in completely different ways.

We now proceed to formally define the notion of a  $k$ -punctured code. It is closely oriented after the definition of a conventional code — only total words are replaced by partial ones and compatibility substitutes equality.

**Definition 5.1** *A set of words  $W$  is called a  $k$ -punctured code, if*

$$u_0u_1 \dots u_n \uparrow v_0v_1 \dots v_m$$

*for  $u_i, v_i \in W^{(k-\diamond)}$  implies that  $m = n$  and  $u_i = v_i$  for all  $i \leq n$ .*

The definition of  $k$ -punctured code can be given also in a stronger form, considering two lists of equal length only. The necessary property is stated in the following proposition.

**Proposition 5.1** *For any set of words  $W$  that is not a  $k$ -punctured code, there are a natural number  $m$  and words  $u_0, u_1, \dots, u_m$  and  $v_0, v_1, \dots, v_m$  all from  $W^{(k-\diamond)}$  such that  $u_0u_1 \dots u_m \uparrow v_0v_1 \dots v_m$  and not  $u_i = v_i$  for all  $i \leq m$ .*

*Proof.* [Proof]For any set of words  $W$ , which is not a  $k$ -punctured code, there are natural numbers  $l$  and  $n$ , and there are words  $u_0, u_1, \dots, u_l$  and  $v_0, v_1, \dots, v_n$  all from  $W$  such that

$$u_0u_1 \dots u_l \uparrow v_0v_1 \dots v_n$$

and not  $u_i = v_i$  for all  $i \leq \min\{l, n\}$ . Now we set  $m := l + n$  and looking at the words  $u_0u_1 \dots u_lv_0v_1 \dots v_n$  and  $v_0v_1 \dots v_nu_0u_1 \dots u_l$ , which are compatible according to the multiplication rule, we see that the lemma's statement is true.  $\square$

We will work only with integer coefficients; for sets of words with a rather homogenous length, the results will also say something about relative puncturedness. For example, in the very homogenous case of a set of words all of length six is a 2-punctured code, if and only if it is a  $\frac{2}{3}$ -punctured code. Here the two notions simply coincide.

It is clear already from the definition that the original language of a punctured code is also a conventional code. The converse, however, does not hold true.

**Example 5.1** *Trivially, the code  $\{a, b\}$  is not a 1-punctured code, because any hole can originate from either  $a$  or  $b$ . A slightly less trivial example is the code  $a^+b$ . Here, already for the 1-punctured set a word  $a^m \diamond a^n b$  is compatible to both  $a^m b \cdot a^n b$  and  $a^{m+n+1} b$ .*

For sets of two words, the following proposition might simplify a first step in testing for the punctured code property.

**Proposition 5.2** *If a set of non-empty words  $\{u, v\}$  such that  $|u| > |v|$  is not a  $k$ -punctured code, then  $\{v, u[0 \dots |u| - |v|]\}$  is not a  $k$ -punctured code either.*

*Proof.* [Proof] Let  $\{u, v\}$  be a set, which is not a  $k$ -punctured code, such that  $|u| > |v|$ . In that case there exists a shortest word  $w$  with two factorizations over  $\{u, v\}^{(k-\diamond)}$ . Clearly, the two factorizations must begin and end with partial words originating from different words (thus one from  $u$ , one from  $v$ ); otherwise there would be a shorter word with two factorizations.

With the above considerations it follows that there exist  $u', u'' \in u^{(k-\diamond)}$ ,  $v', v'' \in v^{(k-\diamond)}$  and non-empty partial words  $r', s'$  such that  $u' \uparrow v' r'$  and  $u'' \uparrow s' v''$ . Furthermore,  $r$  and  $s$  can be chosen in such a way as not to contain no holes by just setting them to  $r := u[|v| \dots |u| + 1]$  and  $s := u[0 \dots |u| - |v|]$ . Now neither  $\{v, r\}$  nor  $\{v, s\}$  is a  $k$ -punctured code – for both also the word  $w$  from above provides a counterexample.  $\square$

The converse does not hold true, some more sophisticated condition would be needed to establish equivalence.

**Example 5.2** *The set  $\{aaaa, bbbaaaa\}$  is a 2-punctured code. However, the set reduced after the fashion of Proposition 5.2 is  $\{aaaa, aaa\}$ , and this is not even a conventional code.*

When the two words involved have equal length, things are much easier.

**Proposition 5.3** *Let  $W$  be a set of non-empty words  $\{u, v\}$  such that  $|u| = |v|$ . Then  $W$  is a  $k$ -punctured code, if and only if  $u^{(k-\diamond)} \cap v^{(k-\diamond)} = \emptyset$ .*

*Proof.* [Proof] By definition, every  $k$ -punctured code  $\{u, v\}$  fulfills the latter condition. For the inverse inclusion, we suppose there exists a set  $\{u, v\}$  such that  $u^{(k-\diamond)} \cap v^{(k-\diamond)} = \emptyset$ , which is not a  $k$ -punctured code. In that case as in the proof of Proposition 5.2 there exists a shortest word  $w$  with two factorizations over  $\{u, v\}^{(k-\diamond)}$ , and one factorization must begin with a word originating from  $u$ , the other with one originating from  $v$ . If now  $u$  and  $v$  had the same length, then  $u^{(k-\diamond)} \cap v^{(k-\diamond)}$  would have to be non-empty contradicting the assumptions.  $\square$

Proposition 5.3 can be generalized to an arbitrary number of words. For constant-length codes, i.e. codes of words of equal lengths, the Hamming distance between these words is the decisive factor for determining, whether they are also punctured codes. The Hamming distance between two strings is the number of positions in which they differ. For total words all constant-length sets are codes.

**Proposition 5.4** *A constant-length code is a  $k$ -punctured code, if and only if the Hamming distance between all of its elements is pairwise at least  $k + 1$ .*

*Proof.* [Proof] Since the constant length of the codewords already provides us with the borders of the factorization, clearly a Hamming distance of  $k + 1$  suffices to distinguish them, even if up to  $k$  holes are put into them. If on the other hand two words have a Hamming distance of less than  $k + 1$ , putting holes in all distinguishing positions will make the origin indeterminable.  $\square$

When speaking about codes, one is normally also interested in whether some more words could be added without violating the code property. A  $k$ -punctured code  $W$  is maximal, if for all words  $w$  from  $\Sigma^+ \setminus W$  the set  $W \cup \{w\}$  is not a  $k$ -punctured code.

**Theorem 5.1** *Every  $k$ -punctured code over an alphabet  $\Sigma$  is contained in a maximal  $k$ -punctured code over  $\Sigma$ .*

*Proof.* [Proof] We consider the partial order on all  $k$ -punctured codes created by set inclusion. It suffices to show that any chain in this partial order has a least upper bound, which is again a  $k$ -punctured code. Clearly, the union of all the chain's elements is its least upper bound. Let us call it  $W$ .

If the set  $W$  were not a  $k$ -punctured code, then by definition there would be a positive integer  $k$  and words  $w_0, w_1, \dots, w_k, w'_0, w'_1, \dots, w'_k$  from  $W^{(k-\diamond)}$ , such that  $w_0 w_1 \dots w_k \uparrow w'_0 w'_1 \dots w'_k$ . Since  $W$  is the union of all elements of the chain, there is a maximal element in this chain containing all these  $w_i$  and  $w'_i$ . The chain, however, consists only of  $k$ -punctured codes, and thus this situation is impossible, also  $W$  is a  $k$ -punctured code.  $\square$

As one might expect, the more holes are allowed, the less sets of words preserve the uniqueness of the factorization.

**Proposition 5.5** *For every positive integer  $k$  the class of all  $k + 1$ -punctured codes is strictly included in the class of  $k$ -punctured codes.*

*Proof.* [Proof] The inclusion is obvious. For the strictness consider the language  $W = \{a^{k+1}, b^{k+1}\}$ . It is a  $k$ -punctured code by Proposition 5.4. By the same proposition it is not a  $k + 1$ -punctured code.  $\square$

With this we close our first brief survey of elementary properties of punctured codes.

## 6 Outlook

One task that certainly remains is to further investigate codes of partial words with a special focus on properties which are often required in DNA computations. Such properties are for example involution-freeness and involution-compliance [11].

Guaranteeing these properties even for a certain degree of puncturedness would make the coding of a problem even to some extent error-resistant. This might be

quite desirable in computations involving many steps and thus many possibilities for the introduction of errors.

Of course, also other combinatorial properties of languages, which have no connection to bio-computing should be adapted to punctured languages and should be investigated. This would provide further investigations with a set of tools like the ones already existing for total languages. An example for an interesting topic might be local testability, which quite certainly changes, if some holes appear.

#### Acknowledgement

This work was done, while the author was funded by the Spanish Ministry of Culture, Education and Sport under the Programa Nacional de Formacin de Profesorado Universitario.

## References

- [1] J. BERSTEL and L. BOASSON: *Partial Words and a Theorem of Fine and Wilf*. In: Theoretical Computer Science, Vol. 218, 1999, pp. 135–141.
- [2] J. BERSTEL and D. PERRIN: *Theory of Codes*. Academic Press, 1985.
- [3] F. BLANCHET-SADRI: *Periodicity on Partial Words*. In: Computers and Mathematics with Applications, to appear.
- [4] F. BLANCHET-SADRI: *Codes, Orderings, and Partial Words*. submitted.
- [5] F. BLANCHET-SADRI: *Primitive Partial Words*. submitted.
- [6] F. BLANCHET-SADRI and A. HEGSTROM: *Partial Words and a Theorem of Fine and Wilf Revisited*. In: Theoretical Computer Science, Vol. 270, No. 1/2, 2002, pp 401–419.
- [7] F. BLANCHET-SADRI and D.K. LUHMANN: *Conjugacy on Partial Words*. In: Theoretical Computer Science, Vol. 289, No. 1, 2002, pp 297–312.
- [8] M.A. HARRISON: *Introduction to Formal Language Theory*. Reading, Mass., 1978.
- [9] S. HORVTH, P. LEUPOLD and G. LISCHKE: *Roots and Powers of Regular Languages*. In: Proceedings of Developments in Language Theory 2002, Lecture Notes in Computer Science 2450, Springer-Verlag, Berlin, 2003.
- [10] S. HORVTH and M. ITO: *Decidable and Undecidable Problems of Primitive Words, Regular and Context-free Languages*. In: Journal of Universal Computer Science, Nov. 1999.
- [11] S. HUSSINI, L. KARI and S. KONSTANTINIDIS: *Coding Properties of DNA Languages*. In: Theoretical Computer Science, Vol. 290, 2003, pp. 1557-1579.

- [12] P. LEUPOLD: *Some Properties of Context-Free Languages Related to Primitive Words*. Proceedings of WORDS'03, TUCS General Publications, Turku, September 2003.
- [13] P. LEUPOLD: *Partial Words for DNA Coding*. Presented at DNA 10, June 2004, Milano.
- [14] G. LISCHKE: *The Root of a Language and its Complexity*. Proceedings of Developments in Language Theory 2001, LNCS 2295, Springer-Verlag, pp. 272–280.
- [15] G. ROZENBERG and A. SALOMAA (EDS.): *Handbook of Formal Languages*. Springer-Verlag, Berlin, 1997.
- [16] A.M. SHUR and Y.V. GAMZOVA: *Periods' Interaction Property for Partial Words*. Proceedings of WORDS'03, TUCS General Publications, Turku, September 2003.
- [17] H.J. SHYR: *Free Monoids and Languages*. Hon Min Book Company, Taichung, 1991.