

# Partial Words for DNA Coding

Peter LEUPOLD\*

Research Group on Mathematical Linguistics  
Rovira i Virgili University  
Pça. Imperial Tàrraco 1, 43005 Tarragona, Catalunya, Spain  
eMail: klauspeter.leupold@estudiants.urv.es

**Abstract.** A very basic problem in all DNA computations is finding a good encoding. Apart from the fact that they must provide a solution, the strands involved should not exhibit any undesired behaviour, especially they should not form secondary structures. Various combinatorial properties like repetition-freeness and involution-freeness have been proposed to exclude such misbehaviour. Another option, which has been considered, is requiring a big Hamming distance between the codewords.

We propose to consider partial words for the solution of the coding problem. They, in some sense, already include the Hamming distance in the definition of compatibility and are investigated for many combinatorial properties. Thus, they can be used to guarantee a desired distance and simultaneously other properties. As the investigations on partial words are attracting more and more attention, they might be able to provide an ever-growing toolbox for finding good DNA encodings.

## 1 Introduction

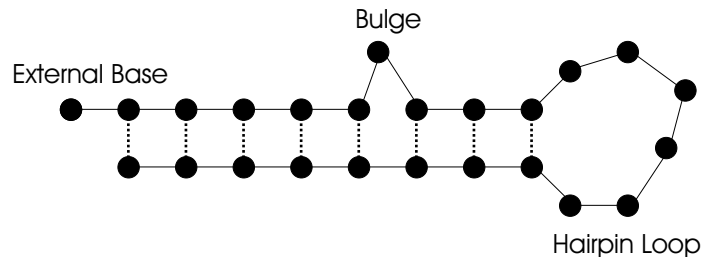
Partial words were introduced by Berstel and Boasson in 1998 [3]. One of their main motivations came from the behaviour of DNA strands. Two strands complementing each other very closely, but having a few mismatches can still align with each other. In such a double strand of DNA with a few mismatches, one cannot tell which of the two non-matching bases is the right or original one. Thus one might consider the respective position as one without information, a whole, and then see what can still be said about the resulting word.

On the other hand, DNA or RNA strands are the basic building blocks and the carriers of information in all DNA computations. There, a major issue is finding the right words to encode the problem under consideration. Computations like the famous, seminal experiment by Adleman depend on the usage of sequences, which will recombine exactly in the ways intended in the design.

Of course, the central problem is finding a suitable set of linear sequences, whose recombinations will constitute the computation. However, in addition the designer faces further problems due to the fact that in reality nucleic acid strands are not linear but three-dimensional objects. Thus the selected strands might form three-dimensional secondary structures like loops and hairpins. This should be avoided by all means, i.e. no parts of the strand should align with other parts of the same strand to form structures like the ones depicted in Figure 1, which we have taken from an article by Adronescu et al. [1]. Such secondary structures obviously hinder further combinability, readability etc., and thus can render useless a computation designed very well for one-dimensional strings, but without consideration of the actual behaviour of longer strands of nucleic acids in the three-dimensional real world.

---

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



**Fig. 1.** Various possible secondary structures of an RNA strand; the line indicates the backbone, the dotted ones paired bases.

Adronesco et al. also mention a rather simple model trying to predict such structures, the *no repeated  $k$ -strings* model [1]. There, a sequence is supposed to have non-empty secondary structure, if there is a repetition without overlap of a factor of length  $k$  in the string. It should be emphasized that this need not be a direct repetition, usually termed square in combinatorics on words – in this case an arbitrarily long sequence can separate the two repetitions; it should be added that would one really be looking for *reverse complement* strings instead of repetitions, which could be done with essentially the same algorithms.

In a different approach Deaton et al. have considered the Hamming distance between the strands of a coding as an approximate measure for the reliability of a computation [9]. The farther apart the code words are, the less probable undesired bindings are.

Partial words seem a good tool to combine properties like these two: if we replace equality of words by compatibility, we in some sense get the Hamming distance for free. If then we define a property like repetition-freeness also for partial words, we can guarantee both properties in a unified way.

We will proceed to illustrate this considering another property, namely involution-freeness, which was introduced by Hussini et al. [13] and further developed with numerous variants in subsequent work [11, 12, 14]. Before this, we mention yet another motivation to use partial words in the context of DNA computation: if such a computation lasts for a longer time and involves some recombinations and especially copying of strands, errors are bound to be introduced in some of the strands; for example, copying processes never work with absolute perfection. If one wants to choose a set of DNA words fulfilling a property, even after some bases may have been changed, it is an appropriate approach to check the following: does the desired property still hold, if the originally chosen language of code words and their possible catenations is punctured up to a certain degree with holes like in a partial word. Such a code language would be more robust to errors and external influence and would thus promise more reliable computations. Also the production of not exactly complementary but very similar sequences, which still might align to each other, would be outruled.

## 2 Partial Words

The main motivation for the introduction of partial words mentioned by Berstel and Boasson [3] came from molecular biology of nucleic acids. There, among other things, one tries to determine properties of the DNA or RNA sequences encountered in nature. 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 only perfect pairings of bases is supposing an ideal world. As long as the number of mismatches is not very high, similar strands will still align due to the affinity of their matching parts.

If one then encounters a pair of strands as depicted in Figure 2, 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



**Fig. 2.** Part of an RNA sequence with two mismatches.

them as concisely as possible. To this end, it seems a plausible choice to regard the positions 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. Looking at 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.

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  $\vartheta(u) := \frac{\text{Hole}(u)}{|u|}$ .

A final notion we will need concerns the intersection of two sets of partial words. For it to contain also words not in either language but compatible to at least one

word from each language, we define a modified intersection:

$$K \sqcap L := \{w : \exists u \in K, \exists v \in L[w = u \vee v]\}$$

An important question is how to obtain such languages of partial words to start with. Because, in this context, holes are considered as some type of defect, which might occur just about anywhere, we choose the following approach: we start out from a language  $L$  of total words.

**Definition 2.1** *For some puncturedness coefficient  $r$  with  $0 < r \leq 1$  the language  $L^{r-\diamond}$ , called  $L$ 's  $r$ -puncturing, is the one that contains all words of  $L$  and all the partial words one can obtain from these obeying the bound imposed by the coefficient  $r$ .*

As already mentioned, the Hamming distance between code words has been considered as a measure for the quality of a code [9]. To make evident the close connection between compatibility and the Hamming distance we close this section by stating a rather obvious equivalence.

**Proposition 2.1** *Let  $k, m$  be two natural numbers with  $m > 2k$ . Two words  $u, v \in (\Sigma^m)^{k-\diamond}$  are compatible, if and only if their Hamming distance is less than or equal  $2k$ .*

Notions from classical Formal Language Theory are not explained here; we only mention that  $\Sigma$  shall always denote the alphabet under consideration and that  $\Sigma^*$  is the set of all words over this alphabet; finally  $\Sigma^+ := \Sigma^* \setminus \{\lambda\}$ .

### 3 Involutions and DNA

We now introduce a special class of mappings, so-called involutions. These enjoy some special interest in the context of DNA computing, because the Watson-Crick complementarity corresponds to a specific involution to be introduced further down.

In general, an *involution* is a mapping  $\theta$  such that  $\theta^2$  is the identity mapping. A mapping such that always  $\mu(uv) = \mu(u)\mu(v)$  is a *morphism*; an involution also fulfilling this property will be called a *morphic involution*.

We now recall a few special involutions acting over the DNA-alphabet  $\Delta = \{A, C, G, T\}$ , which were introduced, for example, by Hussini et al [11]; their specific importance in the context of DNA is that a strand and its image under  $\tau$  align. Thus, strands which are not supposed to align with themselves to form secondary structures should not contain at the same time some factor and its image as explained with more detail in the cited source.

The complement involution  $\gamma$  is defined by  $\gamma(A) := T$ ,  $\gamma(T) := A$ ,  $\gamma(G) := C$ , and  $\gamma(C) := G$ ; additionally we define  $\gamma(\diamond) := \diamond$  to extend the mapping from total to partial words. A second involution is the mirror involution  $\mu$  mapping every word into its mirror image; i.e. it reverses the word's order. Their combination  $\mu\gamma$  is also an involution and will be called the DNA involution  $\tau$ . Thus, for example  $\tau(GTAT) = ATAC$ .

### 4 Involution Compliance and Freedom

For various reasons it seems reasonable to consider in this context only puncturedness bounds relative to the words' length. First, whether two strands align or not, depends not on the absolute number of mismatches, but more on their frequency. While two strands of length eight will almost certainly not align, if there are four

mismatches, the same number is negligible for strands of lengths greater than one hundred.

Secondly, computation means that something is happening; so computing with DNA molecules means that they are changed, at the very least rearranged in some way. Often enough this involves catenation; especially in the matters treated here, catenation plays a central role. And while relative bounds are preserved under catenation, absolute ones are not, because the number of holes is simply summed up for two catenated words.

**Proposition 4.1** *For a rational number  $r$  with  $0 < r < 1$  the inclusion  $(L^{r-\diamond})^* \subseteq (L^*)^{r-\diamond}$  holds.*

It is easily seen that the two sets are in general not equal. Consider a non-empty language  $L$  with only words shorter than  $\frac{1}{r}$ ; its  $r$ -puncturing is just  $L$  itself, it does not contain any words with holes. Thus also the Kleene iteration is a total language. The  $r$ -puncturing of  $L^*$ , however, contains words of arbitrary length and therefore also words with holes.

To make notation a little easier and more readable we make the following convention: when the puncturing symbol  $\diamond$  will be used without giving either a constant or a relative bound in the form  $r-\diamond$ ; this shall mean that all occurrences within a definition, theorem, etc. have the same relative bound. In general, the respective statements will not be true or make sense without this unstated assumption.

**Definition 4.1** A language of partial words  $L$  is  $\theta$ -compliant for a morphic involution  $\theta$ , if for words  $u, v, w, w' \in \Sigma^{*\diamond}$  we have that  $w, u\theta(w')v \in L^\diamond$  and  $w \uparrow \theta(w')$  imply  $uv = \lambda$ ; if also  $(L)^\diamond \cap \theta(L)^\diamond = \emptyset$ , then  $L$  is strictly  $\theta$ -compliant.

A rather easy to see property of compliance is the following.

**Proposition 4.2** *For every  $\theta$ -compliant language  $L^\diamond$ , also  $\theta(L^\diamond)$  is  $\theta$ -compliant.*

Further, we can see quickly a necessary and sufficient condition for the strictness of compliance.

**Proposition 4.3** *A  $\theta$ -compliant language  $L^\diamond$  is strictly  $\theta$ -compliant, if and only if  $(L^*)^\diamond \cap (\theta(L)^*)^\diamond = \{\lambda\}$ .*

*Proof.* The empty word is in any iteration of a language, and thus always  $\lambda \in (L^\diamond)^* \cap (\theta(L)^\diamond)^*$ . Now suppose there is another word in this set for some strictly  $\theta$ -compliant language  $L$ . This means there are words  $u_1, \dots, u_n$  and  $v_1, \dots, v_m$  all from  $L$  such that

$$(u_1 \cdot u_2 \dots u_n)^\diamond \uparrow \theta(v_1 \cdot v_2 \dots v_m)^\diamond.$$

If  $|u_1| < |\theta(v_1)|$  or  $|u_1| > |\theta(v_1)|$  this leads to a contradiction to  $L$ 's  $\theta$ -compliance or to  $\theta(L)$ 's  $\theta$ -compliance, which by Proposition 4.2 is equivalent. For  $|u_1| = |\theta(v_1)|$  we must have  $u_1 \uparrow \theta(v_1)$ ; so only in this case we need the *strict*  $\theta$ -compliance of  $L$  to reach a contradiction.

The other direction of the implication is rather obvious. □

After compliance, freeness is a second interesting property related to involutions.

**Definition 4.2** A language  $L$  is  $\theta$ -free, if  $(L^2)^\diamond \cap (\Sigma^+\theta(L)\Sigma^+)^\diamond = \emptyset$ ; if also  $(L)^\diamond \cap \theta(L)^\diamond = \emptyset$ , then  $L$  is strictly  $\theta$ -free; it suffices, if  $L \setminus \{\lambda\}$  is strictly  $\theta$ -free.

It is quite clear that the notions just defined carry over from punctured languages to total ones, in exactly the sense of the original definitions [11]. We state this explicitly only in one exemplary case.

**Proposition 4.4** *If any puncturing of a language  $L$  is  $\theta$ -free, then  $L$  itself is  $\theta$ -free.*

*Proof.* For languages of only total words,  $\sqcap$  becomes simply conventional set intersection; thus the two definitions of  $\theta$ -freeness are equivalent, and consequently hold for the same class of languages,  $\square$

As is to be expected, the contrary is not true. To show this, we investigate an example provided by Hussini et al. Here the DNA-involution  $\tau$  and the DNA-alphabet  $\Delta$  are used as introduced in Section 3.

**Example 4.1**  $ACC\Delta^2$  is  $\tau$ -free [11]. With a puncturing factor of  $\frac{1}{10}$  this is not true any more. Consider the two words

$$\begin{aligned} w_1 &= \text{ACCGGACCTG} \\ w_2 &= \text{ACCGGTCCTG}, \end{aligned}$$

where  $w_1$  is from  $ACC\Delta^2ACC\Delta^2$ , and  $\tau(w_2)$  is from the set  $\Delta^+(\Delta^2GGT)\Delta^+$ , which is equal to  $\Delta^+\theta(ACC\Delta^2)\Delta^+$ . They are identical except for the sixth position. As they have length ten, one hole is allowed, and thus the  $\frac{1}{10}$ -puncturing of  $ACC\Delta^2$  is not  $\tau$ -free, because  $(ACC\Delta^2ACC\Delta^2)^\diamond \sqcap (\Delta^+(\Delta^2GGT)\Delta^+)^\diamond$  contains, for example,  $\text{ACCGG}\diamond\text{CCTG}$ , because this word is contained in both languages.

So we see that in general things must be reinvestigated. However, we have chosen our definitions in a way that leaves most results for total languages valid with only the reformulations necessary by the slight differences in definition. We illustrate this with the first result from Hussini et al.

**Proposition 4.5** *For a language  $L$  and a morphic involution  $\theta$  the following hold true:*

- (i) *If  $L^\diamond$  is  $\theta$ -free, then both  $L^\diamond$  and  $\theta(L^\diamond)$  are  $\theta$ -compliant.*
- (ii) *If  $L^\diamond$  is strictly  $\theta$ -free, then  $(L^\diamond)^\diamond \sqcap (\Sigma^*\theta(L)\Sigma^*)^\diamond = \emptyset$*

*Proof.* We give only the proof for (i); it is very analogous to the proof of the original lemma just as the proof for part (ii). So suppose that  $L$  is a language such that  $L^\diamond$  is  $\theta$ -free but at the same time not  $\theta$ -compliant. The latter implies that either  $\Sigma^+\theta(L^\diamond)\Sigma^* \sqcap L^\diamond \neq \emptyset$  or  $\Sigma^*\theta(L^\diamond)\Sigma^+ \sqcap L^\diamond \neq \emptyset$ . Catenating  $L^\diamond$  on the right side in the second case, we obtain  $L^\diamond\Sigma^*L^\diamond\Sigma^+ \sqcap L^\diamond L^\diamond \neq \emptyset$ . With Proposition 4.1 we see that then also  $(L\Sigma^*L\Sigma^+)^\diamond \sqcap (LL)^\diamond \neq \emptyset$ ; this contradicts  $\theta$ -freeness. In the first case an analogous contradiction is reached. Finally, for  $\theta(L^\diamond)$  the inclusion now follows with Proposition 4.2.  $\square$

As already mentioned, there is some special interest in the behaviour of certain properties with respect to catenation. For the case of compliance, we can state a positive result in this respect.

**Proposition 4.6** *If two languages  $L_1^\diamond$  and  $L_2^\diamond$  are both  $\theta$ -compliant a morphic involution  $\theta$ , then also their catenation  $L_1 \cdot L_2$  is  $\theta$ -compliant.*

*Proof.* We assume that there exist two  $\theta$ -compliant languages  $L_1$  and  $L_2$ , whose catenation is not  $\theta$ -compliant. This means there are partial words  $u, v, w, w'$  such that  $w, u\theta(w')v \in L_1^\diamond \cdot L_2^\diamond$  and  $w \uparrow \theta(w')$ . Let  $w$  be composed from  $w_1 \in L_1^\diamond$  and  $w_2 \in L_2^\diamond$ , and  $u\theta(w')v$  from  $z_1 \in L_1^\diamond$  and  $z_2 \in L_2^\diamond$ . The border between  $z_1$  and  $z_2$  must be inside the factor  $\theta(w')$ ; otherwise we obtain an immediate contradiction to the  $\theta$ -compliance of either  $L_1^\diamond$  or  $L_2^\diamond$  looking at  $w_1$  respectively  $w_2$ .

So  $w'$  has a factorization  $w'_1w'_2$  such that  $u\theta(w_1) \in L_1^\diamond$  and  $\theta(w_2)v \in L_2^\diamond$ . Now if  $|w_1| \leq |w'_1|$ , then  $w_1$  is compatible to a prefix of  $\theta(w'_1)$  in the word  $u\theta(w'_1)$ , which is in contradiction to the  $\theta$ -compliance of  $L_1^\diamond$ , because  $u\theta(w_1) \in L_1^\diamond$ . If, on the other hand,  $|w_1| > |w'_1|$ , then  $|w_2| \leq |w'_2|$ ; we obtain an analogous contradiction to the  $\theta$ -compliance of  $L_2^\diamond$ .  $\square$

## 5 Constant Length Codings

Now we will restrict our attention to a class of languages, which seems of special interest in the context of DNA computations. Many times all the original strands employed at the beginning of an experiment have the same length. Among other advantages this allows, for example, telling how far the experiment has proceeded in a simple way: the length of the present strands corresponds directly to the number of catenations, which created them. And determining strand length via gel electrophoresis is a very reliable standard procedure.

Thus it seems reasonable to consider languages all of whose words have equal length. This property allows the statement of some results, which are not true in general. We note further that the DNA involution  $\tau$  as well as its components  $\mu$  and  $\gamma$  all are length-preserving (this means the length of original and image is always the same); therefore this is a sensible restriction to put on the involutions under consideration.

First off, we note that strictness loses its meaning for involution compliance.

**Proposition 5.1** *A language  $L \subset \Sigma^{n\Diamond}$  is  $\theta$ -compliant for a length-preserving involution  $\theta$ , if and only if it is strictly  $\theta$ -compliant.*

*Proof.* Immediate from the definition: For non- $\theta$ -compliance  $u\theta(w')v$  (variable names referring to Definition 4.1) must have the same length as  $w$ . Because always  $|w| = |w'| = |\theta(w')|$  for  $w, w' \in L$ ,  $uv$  is empty in any counterexample to  $\theta$ -compliance.  $\square$

For involution-freeness, the analogous statement is not true as shown by the following example.

**Example 5.1** The set  $\{TGGT, ATAC\}$  is strictly  $\tau$ -free. If we add the word  $ACCA = \tau(TGGT)$ , the resulting set is still  $\tau$ -free, the strictness, however, is lost.

**Proposition 5.2** *A constant-length language  $L^\Diamond$  is strictly  $\theta$ -free for a length-preserving involution  $\theta$ , if and only if it is  $\theta$ -free and  $\theta$ -compliant.*

*Proof.* From strict  $\theta$ -freeness,  $\theta$ -freeness follows by definition;  $\theta$ -compliance follows by Proposition 4.5.

The inverse inclusion follows immediately from Proposition 5.1 and the fact that the strictness condition is the same for both freeness and compliance.  $\square$

Now we provide a sufficient condition for a constant-length language to be  $\theta$ -free, and also for strict  $\theta$ -freeness. Here  $\text{pref}(L)$  denotes the set of all proper prefixes of words in  $L$  and  $\text{suff}(L)$  denotes the set of all proper suffixes including the empty word. For one word the sets of prefixes and suffixes are both finite. Therefore the conditions provided can be checked very easily for any finite language.

However, here we need to give the puncturedness bounds explicitly, because they are not uniform for all languages involved – there is even a mixture of relative and absolute bounds. The proof will make clear, why this is necessary.

**Proposition 5.3** *For  $L \subset \Sigma^n$ , the language  $L^{r\Diamond}$  is  $\theta$ -free, if and only if  $(\text{pref}(L)\text{suff}(L))^{k\Diamond} \cap L^{r\Diamond} = \emptyset$  for  $k := \lfloor 2 \cdot r \cdot n \rfloor$ .*

*Proof.* If  $L^{r\Diamond}$  is not  $\theta$ -free, then there exist total words  $u, v, w \in L$ , such that a word from  $\theta(w^{r\Diamond})$  is compatible to a subword of one from  $(uv)^{r\Diamond}$ . This subword is neither entirely from  $u^{2r\Diamond}$  nor entirely from  $v^{2r\Diamond}$  due to the definition of  $\theta$ -freeness. Further, because all three words have equal length the subword touches both factors.

Since this subword is shorter than  $uv$  itself, the local puncturedness coefficient might be higher than the one for the entire word. The maximum number of holes in this word is the following: the length of  $uv$  is  $2n$ , thus with a puncturedness coefficient of  $r$  there can be at most  $\lfloor 2 \cdot r \cdot n \rfloor$  holes in the entire word from  $(uv)^{r-\diamond}$  – in the extreme case all of them can be in the subword considered above. Thus the set  $(\text{pref}(L)\text{suff}(L))^{k-\diamond} \sqcap L^{r-\diamond}$  is non-empty. After these considerations also the inverse inclusion follows easily.  $\square$

By extending the prefix and suffix sets by the original words themselves, we immediately obtain an analogous characterization of strictly  $\theta$ -free languages.

**Corollary 5.3.a** *For  $L \subset \Sigma^n$ , the language  $L^{r-\diamond}$  is strictly  $\theta$ -free, if and only if  $((L \cup \text{pref}(L))^{2r-\diamond} (L \cup \text{suff}(L))^{2r-\diamond}) \sqcap L^{r-\diamond} = \emptyset$ .*

## 6 Outlook

What was presented here is only one example for possible usage of partial words in the context of biological computation, or more general, in dealing with DNA sequences with certain desired or undesired properties. In some contexts also a variation of partial words might be useful.

As suggested by both Gh. Păun and G. Lischke, regarding a base pair like  $A-G$  as a hole is to some extent giving away some more information than necessary. Although theoretically possible, it is in practice improbable that both bases have been produced by an error. So instead of regarding the position as a complete unknown, one might attribute to it a type like  $\{A-T, C-G\}$ , supposing that only the  $A$  or only the  $G$  is wrong. Considering only one strand we would use  $\{A, C\}$ ; either way, this should then be treated as a letter compatible to all its elements, but not to the other letters.

For the DNA alphabet  $\Delta$ , the version extended in this sense would be

$$\{A, C, G, T, \{A, C\}, \{A, G\}, \{T, C\}, \{T, G\}\}.$$

The other possible binary combinations like  $\{A, T\}$  are, of course well-matched pairs and do not appear here explicitly, but are already represented by the single letters. To our knowledge this type of partiality has not been investigated yet.

We have stated before that using partial words in the original form for the encoding of a DNA computation is very related to the use of Hamming distances between the code words. Already now combinatorial investigations on partial words offer quite a number of tools concerning periodicity [7],[17], primitivity [5], codes [6],[15] etc. Thus, and as the combinatorial theory around partial words is growing, their use might have the advantage over the plain Hamming distance that many properties have already been investigated and may provide ways to guarantee some desired properties. Of course, taking into account the peculiarities of DNA, also some tailor-made restrictions of partial words might be defined and investigated for special purposes.

## Acknowledgement

The author is thankful to Max Garzon for pointing him to the coding problem. Further thanks are due to an anonymous referee for close reading of and detailed comments about the manuscript.



## References

1. M. ANDRONESCU, D. DEES, L. SLAYBAUGH, Y. ZHAO, A.E. CONDON, B. COHEN, and S. SKIENA: *Algorithms for Testing That Sets of DNA Words Concatenate without Secondary Structure*. In: [10], pp. 182–195.
2. W. BAUER, H. EHRIG, J. KARHUMÄKI and A. SALOMAA (EDS.): *Formal and Natural Computing*. Lecture Notes in Computer Science 2300, Springer-Verlag, Berlin, 2002.
3. J. BERSTEL and L. BOASSON: *Partial Words and a Theorem of Fine and Wilf*. In: *Theoretical Computer Science*, Vol. 218, 1999, pp. 135–141.
4. J. BERSTEL and D. PERRIN: *Theory of Codes*. Academic Press, 1985.
5. F. BLANCHET-SADRI: *Primitive Partial Words*. Preprint 2003.
6. F. BLANCHET-SADRI: *Codes, Orderings, and Partial Words*. Preprint 2003.
7. 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.
8. J. CHEN and J.H. REIF (EDS.): *DNA Computing, 9th International Workshop on DNA Based Computers*. Lecture Notes in Computer Science 2943, Springer-Verlag, Berlin, 2004.
9. R. DEATON, M. GARZON, R.C. MURPHY, J.A. ROSE, D.R. FRANCESCHETTI and S.E. STEVENS JR.: *On the Reliability and Efficiency of a DNA-based Computation*. In: *Physical Review Letters* 80:2, 1998, pp. 417–420.
10. M. HAGIYA and A. OHUCHI (EDS.): *DNA Computing — 8th Int. Workshop on DNA-Based Computers*. Lecture Notes in Computer Science 2568, Springer-Verlag, Berlin, 2003.
11. S. HUSSINI, L. KARI and S. KONSTANTINIDIS: *Coding Properties of DNA Languages*. In: *Theoretical Computer Science*, Vol. 290, 2003, pp. 1557–1579.
12. N. JONOSKA and K. MAHALINGAM: *Languages of DNA Based Code Words*. In: [8], pp. 61–73.
13. L. KARI, R. KITTO and G. THIERRIN: *Codes, Involutions and DNA Encodings*. In: [2].
14. L. KARI, S. KONSTANTINIDIS, E. LOSSEVA and G. WOZNIAK: *Sticky-free and Overhang-free DNA Languages*. In: *Acta Informatica* 40(2), 2003, pp. 119–157.
15. P. LEUPOLD: *Languages of Partial Words*. Submitted.
16. G. ROZENBERG and A. SALOMAA (EDS.): *Handbook of Formal Languages*. Springer-Verlag, Berlin, 1997.
17. A.M. SHUR and Y.V. GAMZOVA: *Periods' Interaction Property for Partial Words*. In: *Preproceedings of WORDS'03*, TUCS General Publications, Turku, 2003.
18. H.J. SHYR: *Free Monoids and Languages*. Hon Min Book Company, Taichung, 1991.