Pattern Discovery in Data: Linear Bound on Irredundant Motifs & a Polynomial Algorithm

Laxmi Parida
Computational Biology Center,
IBM Thomas J. Watson Research Center,
Yorktown Heights, NY10598, USA
email: parida@us.ibm.com

March 19, 1999

Abstract

Given an input sequence of data, a motif is a repeating pattern (possibly interspersed with "dont care" characters). The *number* of motifs could potentially be exponential in the size (number of characters) of the input. In this paper, we show, for any sequence with n characters, there exists only a *linear* (or no more than $3n$) number of special motifs and every other motif can be generated from this. We term these special motifs as *irredundant* motifs. The result is meaningful also from an algorithmic viewpoint, since the ideas from the proof has been used to design a polynomial time algorithm to detect these irredundant motifs. This bound on the number of useful motifs gives validation to motif-based approaches, since the total number of irredundant motifs does not explode. This result is of significance to most applications that use pattern discovery as the basic engine such as data mining, clustering, matching and others. From an information-theoretic viewpoint, redundant motifs represent the sets with very high co-relation, hence of lower informational value. Besides, it has been shown in some motif based applications, such as multiple sequence alignment [PFR99b, PFR99a], these irredundant motifs capture all the relevant information.

1 Introduction

Given an input sequence of data, a motif is a repeating pattern (possibly interspersed with dont-care characters) that occurs in the sequence. The number of motifs could potentially be exponential in the size (number of characters) of the input (see Examples 1 and 2 in the following section). Typically, the higher the self similarity in the sequence, greater is the number of motifs in the data. Motif discovery on such data, such as repeating DNA or protein sequences, is a source of concern since these exhibit a very high degree of self-similarity (repeating patterns). Usually, this problem of an exploding number of motifs is tackled by pre-processing the input, using heuristics, to remove the repeating or self-similar portions of the input. Another way of trimming down the number of motifs is to use a "statistical significance" measure. However, due to the absence of a good understanding of the domain, there is no consensus over the right model to use. Thus there is a trend towards model-less motif discovery in different fields. To keep the problem manageable, it is useful to identify a small number of motifs that capture important information about the family of motifs.

In this paper, we show that, for any sequence, there exists only a linear number of special motifs and every other motif can be simply *generated* from them. We name these special motifs *irredundant* motifs. The result is meaningful also from an algorithmic viewpoint, since the ideas from the proof can be used to design a polynomial time algorithm to detect these irredundant motifs. This bound on the number of useful motifs gives validation to motif-based approaches, since the total number of irredundant motifs does not explode. This result is of significance to most applications that use pattern discovery as the basic engine such as data mining, clustering and matching. This family of irredundant motifs is also very characteristic of the family of all the motifs: in applications such as multiple sequence alignment, we have shown that the irredundant motifs suffice to obtain the alignment [PFR99a, PFR99b]. However, in applications that use the motifs to extract signature motifs of sets of sequences, all the motifs, including the redundant ones, may be of relevance.

Roadmap. In Section 2 we define motifs and some basic related concepts (such as maximality) and in Section 3 we introduce the notion of irredundancy and in Section 4 show that the number of such motifs is only linear in the input length. In Section 5 we describe a low polynomial time algorithm to extract the irredundant motifs and in Section 6 we give a systematic and efficient way to compute all the redundant motifs. We conclude with two examples of possible applications of the notion of redundancy and the polynomial time algorithm in Section 7.

2 Notation

Let s be a sequence on an alphabet Σ, '.' $\notin \Sigma$. A character from Σ, say σ, is called a solid character and '.' is called a "dont care" character. For brevity of notation, if x is a sequence, then $|x|$ denotes the length of the sequence and if x is a set of elements then $|x|$ denotes the cardinality of the set. The j^{th} $(1 \leq j \leq |s|)$ character of the sequence is given by $s[j]$.

Definition 1 *($\sigma_1 <, =, \leq \sigma_2$) If σ_1 is a "dont care" character and $\sigma_2 \in \Sigma$, then $\sigma_1 < \sigma_2$. If both σ_1 and σ_2 are identical characters in Σ, then $\sigma_1 = \sigma_2$. If either $\sigma_1 < \sigma_2$ or $\sigma_1 = \sigma_2$ holds, then $\sigma_1 \leq \sigma_2$.*

Definition 2 *(p occurs at l, cover) A string, p, on $\Sigma \cup$ '.', occurs at position l in s if $p[j] \leq s[l+j]$*

holds for $1 \leq j \leq |p|$. p *is said to cover the interval* $[l, l+|p|-1]$ *on* s.

Definition 3 *(k-motif m, location list* $\mathcal{L}_m$*) Given a string* s *on alphabet* Σ *and a positive integer* k, $k \leq |s|$, *a string* m *on* $\Sigma \cup$ '.' *is a* k*-motif with location list* $\mathcal{L}_m = (l_1, l_2, \ldots, l_p)$, *if all of the following hold:*

1. $m[1], m[|m|] \in \Sigma$.
   (The first and last characters of the motif are solid characters; if "dont care" characters are allowed at the ends, the motifs can be made arbitrarily long in size without conveying any extra information.)
2. $p \geq k$.
3. *there does not exist a location* l, $l \neq l_i$, $1 \leq i \leq p$ *such that* m *occurs at* l *on* s *(the location list is of maximal size).*
   (This ensures that any two distinct location lists must correspond to distinct motifs.)
4. *for every "dont care" character at position* j *in* m, *there exist at least two distinct occurrences* l_{i_1} *and* l_{i_2}, $1 \leq i_1, i_2 \leq p$, *such that* $s[l_{i_1}+j] \neq s[l_{i_2}+j]$.
   (No "dont care" character can be replaced by a solid character, otherwise an arbitrary number of "dont care" characters could be introduced without conveying any extra information.)

If m is a string on Σ, m is called a *simple motif.* If m is a string of $\Sigma \cup \{\text{'.'}\}$, m is called a *rigid motif.* In the rest of the discussion a k-motif is referred to as a motif.

Consider $s = ABCDABCD$. Using the definition of motifs, the different 2-motifs are as follows:

1. $m_1 = AB$ with $\mathcal{L}_{m_1} = \{1, 5\}$,
2. $m_2 = BC$ with $\mathcal{L}_{m_2} = \{2, 6\}$,
3. $m_3 = CD$ with $\mathcal{L}_{m_3} = \{3, 7\}$,
4. $m_4 = ABC$ with $\mathcal{L}_{m_4} = \{1, 5\}$,
5. $m_5 = BCD$ with $\mathcal{L}_{m_5} = \{2, 6\}$ and
6. $m_6 = ABCD$ with $\mathcal{L}_{m_6} = \{1, 5\}$.

Notice that $\mathcal{L}_{m_1} = \mathcal{L}_{m_4} = \mathcal{L}_{m_6}$ and $\mathcal{L}_{m_2} = \mathcal{L}_{m_5}$. Using the notation $\mathcal{L} + i = \{x + i | x \in \mathcal{L}\}$, $\mathcal{L}_{m_5} = \mathcal{L}_{m_6} + 1$ and $\mathcal{L}_{m_3} = \mathcal{L}_{m_6} + 2$ hold. We call the motif m_6 *maximal* as $|m_6| > |m_1|, |m_4|$ and $|m_5| > |m_2|$. Motifs m_1, m_2, m_3, m_4 and m_5 are non-maximal motifs.

We give the definition of maximality below.

Definition 4 *(Maximal Motif) Let p_1, p_2, ..., p_k be the motifs in a sequence s. A motif p_i is maximal if and only if there is no p_j $(j \neq i)$ such that p_i is a substring of p_j, or, if p_i is a substring of p_j, then there exists at least one occurrence of p_i in s that is not covered by p_j in s.*

However, the notion of maximality alone does not suffice to bound the number of motifs. We now motivate the need for defining irredundant maximal motifs by giving two simple examples of strings that have an unusually large number of maximal motifs without conveying extra information about the input.

Example 1 Let the input string s have the following form:

$$ac_1c_2c_3baXc_2c_3bYac_1Xc_3bYYac_1c_2Xb$$

Then the maximal motifs (which are $2^{\Omega(\sqrt{n})}$ in number) are as follows.

motif	location list			
	[$ac_1c_2c_3b$	aXc_2c_3bY	ac_1Xc_3bYY	ac_1c_2Xb]
$a...b$	+	+	+	+
$a..c_3b$	+	+	+	
$a.c_2.b$	+	+		+
$ac_1..b$	+		+	+
$\mathbf{a.c_2c_3b}$	+	+		
$\mathbf{ac_1.c_3b}$	+		+	
$\mathbf{ac_1c_2.b}$	+			+

Example 2 Let $s = aaaaaaaaaaa$ and $k = 2$. By the definition, the motifs with the location lists shown as positions on the input string are as follows.

size	motif	location list										
		[a	a	a	a	a	a	a	a	a	a	a]
2	aa	+	+	+	+	+	+	+	+	+	+	
3	aaa	+	+	+	+	+	+	+	+	+		
4	aaaa	+	+	+	+	+	+	+	+			
5	aaaaa	+	+	+	+	+	+	+				
6	aaaaaa	+	+	+	+	+	+					
7	aaaaaaa	+	+	+	+	+						
8	aaaaaaaa	+	+	+	+							
9	aaaaaaaaa	+	+	+								
10	aaaaaaaaaa	+	+									

In other words, a sequence of n identical characters has $n - 2$ maximal motifs.

Consider a minor variation of the original string $s' = aaaaaXaaaaa$. Note that the number of motifs increase drastically. The motifs, in increasing order of size, along with the locations list are as follows. We also give the non-maximal motifs (for instance all the motifs of size 6) for the sake of completeness.

Motif size $= 2$

	[a	a	a	a	a	X	a	a	a	a	a]
aa	+	+	+	+			+	+	+	+	

Motif size $= 3$

	[a	a	a	a	a	X	a	a	a	a	a]
aaa	+	+	+				+	+	+		
a.a	+	+	+		+		+	+	+		

Motif size $= 4$

	[a	a	a	a	a	X	a	a	a	a	a]
aaaa	+	+					+	+			
a.aa	+	+			+		+	+			
aa.a	+	+		+			+	+			
a..a	+	+		+	+		+	+			

Motif size $= 5$

	[a	a	a	a	a	X	a	a	a	a	a]
aaaaa	+						+				
a.aaa	+				+		+				
aa.aa	+			+			+				
aaa.a	+		+				+				
a..aa	+			+	+		+				
a.a.a	+		+		+		+				
aa..a	+		+	+			+				
a...a	+		+	+	+		+				

Motif size $= 6$

	[a	a	a	a	a	X	a	a	a	a	a]
a..aaa				+	+						
aa..aa			+	+							
aaa..a		+	+								
a.aa.a		+			+						
a.a.aa			+		+						
aa.a.a		+		+							
aa...a		+	+	+							
a...aa			+	+	+						
a.a..a		+	+		+						
a..a.a		+		+	+						
a....a		+	+	+	+						

Motif size = 7

	[a	a	a	a	a	X	a	a	a	a	a]
a..aaaa				+	+						
a.a.aaa			+		+						
a.aa.aa		+			+						
a.aaa.a	+				+						
aa..aaa			+	+							
aa.a.aa		+		+							
aa.aa.a	+		+								
aaaa.aa		+	+								
aaa.a.a	+		+								
aaaa..a	+	+									
a...aaa			+	+	+						
a.a..aa		+	+		+						
a.aa..a	+	+			+						
a.a.a.a	+		+		+						
a..a.aa		+		+	+						
a..aa.a	+			+	+						
a....aa		+	+	+	+						
aa....a	+	+	+	+							
a.....a	+	+	+	+	+						

Motif size = 8

	[a	a	a	a	a	X	a	a	a	a	a]
aa..aaaa			+	+							
aa.aa.aa	+			+							
aaa.a.aa	+		+								
aaaa..aa	+	+									
aaa..aaa		+	+								
aa.a.aaa		+		+							
aa.a..aa	+	+		+							
aa..a.aa	+		+	+							
aaa...aa	+	+	+								
aa...aaa		+	+	+							
aa....aa	+	+	+	+							

Motif size = 9

	[a	a	a	a	a	X	a	a	a	a	a]
aaa.a.aaa	+		+								
aaaa..aaa	+	+									
aaa..aaaa		+	+								
aaa...aaa	+	+	+								

Motif size = 10

	[a	a	a	a	a	X	a	a	a	a	a]
aaaa..aaaa	+	+									

3 Notion of Redundancy

We saw in the last section an example where a small change in the input string (replacing just one character by another) increases the number of maximal motifs from linear to exponential. This suggests that using a notion stronger than maximality would be meaningful. However, the notion of redundancy needs to be meaningful as well! Informally speaking, we call a motif m redundant if m (and its location list $\mathcal{L}_m$) can be deduced from the other motifs *without* studying the input string s. We introduce such a notion below and the section on "generating operation" describes how the redundant motifs and the location lists can be computed from the irredundant motifs [Par98].

Definition 5 *($m_1 \leq m_2$) Given two motifs m_1 and m_2 with $|m_1| \leq |m_2|$, $m_1 \leq m_2$ holds if $m_1[j] \leq m_2[j]$, $1 \leq j \leq |m_1|$.*

For example, let $m_1 = AB..E$, $m_2 = AK..E$ and $m_3 = ABC.E.G$. Then $m_1 \leq m_3$, and $m_2 \not\leq m_3$. The following lemmas are straightforward to verify.

Lemma 1 *If $m_1 \leq m_2$, then $\mathcal{L}_{m_1} \supseteq \mathcal{L}_{m_2}$.*

Lemma 2 *If $m_1 \leq m_2$ and $m_2 \leq m_3$, then $m_1 \leq m_3$.*

Definition 6 *(Redundant motif) A maximal motif m, with location list $\mathcal{L}_m$, is redundant if there exist maximal motifs m_i, $1 \leq i \leq p$, such that $\mathcal{L}_m = \mathcal{L}_{m_1} \cup \mathcal{L}_{m_2} \ldots \cup \mathcal{L}_{m_p}$, (i.e., every occurrence of m on s is already covered by one of the motifs $m_1, m_2, \ldots, m_p$).*

Definition 7 *(fixed-size redundant) A maximal motif m, with location list $\mathcal{L}_m$, is redundant if there exist maximal motifs m_i, $1 \leq i \leq p$, with $\mathcal{L}_m = \mathcal{L}_{m_1} \cup \mathcal{L}_{m_2} \ldots \cup \mathcal{L}_{m_p}$, and $|m| = |m_i|$, $\forall i$ motif m is called* fixed-size *redundant* [1].

Definition 8 *(Irredundant motif) A maximal motif that is not redundant is called an* irredundant *motif.*

Notice that the notion of fixed-size redundancy is a little weaker than the general irredundancy as in Definition 8 since if a motif is fixed-size redundant, it is redundant but the converse is not true. We discuss the example of irredundancy after describing the generating operations.

[1] This notion of redundancy has also been used by the authors in [GYW+97].

3.1 Generating operations

The redundant motifs need to be generated from the irredundant ones, if required. We define the following generating operations. Let m, m_1 and m_2 be motifs.

1. Prefix operator, $P^\delta(m)$, $1 < \delta < |m|$.
   This is a valid operation when δ is an integer and $m[\delta]$ is a solid character, since all the operations are closed under motifs. $P^\delta(m)$ is the string given by $m[1 \ldots \delta]$.

   For example, if $m = AB..CDE$, then $P^3(m)$ is not a valid operation since $m[3]$ is a dot-character. Also, $P^5(m) = AB..C$.

2. The following binary operations are valid only if
   - $|m_1| = |m_2|$ and
   - For each i, $1 \leq i \leq |m_1|$, $m_1[i] \leq m_2[i]$ or $m_2[i] \leq m_1[i]$.

(. $\bigoplus$.) $m = m_1 \bigoplus m_2$, where m is such that every solid-character in m is a solid-character in both m_1 and m_2, otherwise it is a dot-character. For example if $m_1 = A..D..G$ and $m_2 = AB...FG$. Then, $m = m_1 \bigoplus m_2 = A.....G$.

(. $\bigotimes$.) $m = m_1 \bigotimes m_2$, where m is such that every solid-character in m is a solid-character in at least one of m_1 and m_2, otherwise it is a dot-character. For example if $m_1 = A..D..G$ and $m_2 = AB...FG$. Then, $m = m_1 \bigotimes m_2 = AB.D.FG$.

The operations satisfy the following properties.

1. If $m = m_1 \bigoplus m_2$, then $m \leq m_1$ and $m \leq m_2$.
   If $m = m_1 \bigotimes m_2$, then $m_1 \leq m$ and $m_2 \leq m$.

2. (symmetric)
   $m_1 \bigoplus m_2 = m_2 \bigoplus m_1$.
   $m_1 \bigotimes m_2 = m_2 \bigotimes m_1$.

3. (associative)
   $m_1 \bigoplus (m_2 \bigoplus m_3) = (m_1 \bigoplus m_2) \bigoplus m_3$.
   $m_1 \bigotimes (m_2 \bigotimes m_3) = (m_1 \bigotimes m_2) \bigotimes m_3$.

4. The P^δ operator distributes over $\bigoplus$, i.e., $P^\delta(m_1 \bigoplus m_2) = P^\delta(m_1) \bigoplus P^\delta(m_2)$.

These properties are straightforward to verify.

Back to Examples 1 and 2. Consider Example 1 discussed in the last section. The motifs not in bold, in the example, are fixed size redundant. Each of the redundant motif can be constructed from the motifs in the these using the generating operations. For example, $a..c_3b = a.c_2c_3b \bigoplus ac_1.c_3b$ and $\mathcal{L}_{a..c_3b} = \mathcal{L}_{a.c_2c_3b} \cup \mathcal{L}_{ac_1.c_3b}$.

Consider Example 2. For a fixed size of the motif f, $2 \leq f \leq 10$, in this example, the motifs shown in italics are redundant with respect to the other motifs of the same size f. The reader may check the location lists of these motifs to verify the redundancy of the motifs. The basis with respect to fixed-size redundancy consists of all the motifs shown in bold which is $O(n^2)$ in number. However, a basis using the general notion of redundancy for the input string is shown below. To compare it with the basis for the original string s, we reproduce the basis for s as well.

$s' =$ aaaaaXaaaaa

size	motif	[a	a	a	a	a	X	a	a	a	a	a]
2	aa	+	+	+	+			+	+	+	+	
3	aaa	+	+	+				+	+	+		
4	aaaa	+	+					+	+			
5	aaaaa	+						+				
7	a.aaa.a	+				+						
8	aa.aa.aa	+			+							
9	aaa.a.aaa	+		+								
10	aaaa..aaaa	+	+									

$s =$ aaaaaaaaaaa

size	motif	[a	a	a	a	a	a	a	a	a	a	a]
2	aa	+	+	+	+	+	+	+	+	+	+	
3	aaa	+	+	+	+	+	+	+	+	+		
4	aaaa	+	+	+	+	+	+	+	+			
5	aaaaa	+	+	+	+	+	+	+				
6	aaaaaa	+	+	+	+	+	+					
7	aaaaaaa	+	+	+	+	+						
8	aaaaaaaa	+	+	+	+							
9	aaaaaaaaa	+	+	+								
10	aaaaaaaaaa	+	+									

Notice the similarity in the two bases (for s and s'). Irredundant motif of size 6 is missing in the basis for s'. The striking similarity suggests that the general notion of redundancy is perhaps a more natural notion.

Further, every redundant maximal motif of s' can be obtained from its basis using the generating operations. We give some examples for illustration.

1. $a..aa = aaaaa \bigoplus P^5(a.aaa.a) \bigoplus P^5(aa.aa.aa)$ and

$\mathcal{L}_{a..aa} = \mathcal{L}_{aaaaa} \cup \mathcal{L}_{a.aaa.a} \cup \mathcal{L}_{aa.aa.aa}$.

2. $aaa...aa = P^8(aaaa..aaaa) \bigoplus P^8(aaa.a.aaa)$ and $\mathcal{L}_{aaa...aa} = \mathcal{L}_{aaaa..aaaa} \cup \mathcal{L}_{aaa.a.aaa}$.

$\square$

4 Bounding the Irredundant Motifs

Definition 9 *(Basis) Given a sequence s on an alphabet Σ, let $\mathcal{M}$ be the set of all maximal motifs on s. A set of maximal motifs $\mathcal{B}$ is called a basis of $\mathcal{M}$ iff the following hold:*

1. *for each $m \in \mathcal{B}$, m is irredundant with respect to $\mathcal{B} - \{m\}$, and,*
2. *let $\mathbf{G}(\mathcal{X})$ be the set of all the redundant maximal motifs generated by the set of motifs $\mathcal{X}$, then $\mathcal{M} = \mathbf{G}(\mathcal{B})$.*

In general, $|\mathcal{M}| = \Omega(2^n)$. The natural attempt now is to obtain as small a basis as possible. In the following theorem we show that using the notion of fixed-size irredundancy there can be no more than n^2 irredundant motifs where n is the size of the input.

Theorem 1 *Let s be a string with $n = |s|$ and let $\mathcal{B}^f$ be a basis using the fixed-size redundancy. Then $|\mathcal{B}^f| < n^2$.*

Proof. From the definition of irredundancy, it follows that if j motifs are incident on the same location on the input string, then there must exist j distinct locations with at least one of the j motifs incident on each position. Hence, no location can have more than n motifs incident on it. Thus, it follows that the number of these motifs can be no more than n^2 i.e., $|\mathcal{B}^f| < n^2$. $\square$

We prove a much stronger result for the irredundant motifs in the following theorem.

Theorem 2 *Let s be a string with $n = |s|$ and let $\mathcal{B}$ be a basis or a set of irredundant motifs (see Definition 8). Then $|\mathcal{B}| \leq 3n$.*

Proof. The proof is not very straightforward and we begin by giving a few definitions that we use in the argument.

Definition 10 *($\mathcal{L}_a$ straddles $\mathcal{L}_b$) A set $\mathcal{L}_a$ straddles a set $\mathcal{L}_b$ if $\mathcal{L}_a \cap \mathcal{L}_b \neq \phi$, $\mathcal{L}_a - \mathcal{L}_b \neq \phi$ and $\mathcal{L}_b - \mathcal{L}_a \neq \phi$.*

Notice that if $\mathcal{L}_a$ straddles $\mathcal{L}_b$, then $\mathcal{L}_b$ straddles $\mathcal{L}_a$.

Let $\mathcal{L}_a = \{1,2,3\}$, $\mathcal{L}_b = \{2,3,4\}$, $\mathcal{L}_c = \{2,3\}$ and $\mathcal{L}_d = \{5,6\}$. Then $\mathcal{L}_a$ straddles $\mathcal{L}_b$. However $\mathcal{L}_c$ does not straddle any of the others and $\mathcal{L}_d$ does not straddle any of the others.

Definition 11 *(sub-motifs of motif m, $\mathcal{S}^{[j_1,j_2]}(m)$) Given a motif m let $m[j_1], m[j_2], \ldots m[j_l]$ be the l solid characters in the motif m. Then the sub-motifs of m are given as $\mathcal{S}^{[j_i,j_k]}(m)$, $1 \leq i < k \leq l$, which is obtained by dropping all the characters before (to the left of) j_i and all characters after (to the right of) j_k in m.*

For example, let $m_1 = x...yzw..x$, $m_2 = xy$. Then $\mathcal{S}^{[1,10]}(m_1) = m_1$, $\mathcal{S}^{[5,10]}(m_1) = yzw..x$, $\mathcal{S}^{[5,7]}(m_1) = yzw$, $\mathcal{S}^{[6,10]}(m_1) = zw..x$, $\mathcal{S}^{[7,10]}(m_1) = w..x$, $\mathcal{S}^{[10,10]}(m_1) = x$, and $\mathcal{S}^{[2,2]}(m_2) = y$. Notice that $\mathcal{S}^{[1,|m|]}(m) = m$ and $|\mathcal{S}^{[|m|,|m|]}(m)| = 1$ [2]. Also $\mathcal{L}_{\mathcal{S}^{[j_l,|m|]}(m)} \subseteq \mathcal{L}_m + (j_l - 1)$.

Consider $\mathcal{B}^*$ ($\mathcal{B} \subseteq \mathcal{B}^*$) where the motifs in $\mathcal{B}^*$ are *not* maximal and *redundant*. The idea here is to use the non-maximal motifs m (note that $m \notin \mathcal{B}$) in $\mathcal{B}^*$ to count some of the irredundant motifs to which a unique position in s cannot be assigned immediately.

The argument proceeds in two steps.

Step 1. This is the initialization step where we carry out the following:

1. Every position, $x \in \mathcal{L}_{m_1}, \mathcal{L}_{m_2}, \ldots, \mathcal{L}_{m_l}$, is assigned ON/OFF with respect to m as follows: If m_i, $1 \leq i \leq l$, is such that there exists no $j \neq i$, $1 \leq j \leq l$ so that $\mathcal{L}_{m_j} \subset \mathcal{L}_{m_i}$ holds, then x is marked ON, otherwise it is marked OFF w.r.t. m_i. We make the following claims due to this ON/OFF marking:

   Claim 1 *At the end of this step, every motif m that is* not *redundant, has at least one location $x \in \mathcal{L}_m$ marked ON w.r.t. m.*

   Proof. This follows directly from the definition of redundancy. Also a redundant motif m may not have any position marked ON w.r.t m. See Figure 1 for an illustration. □

   Claim 2 *If location x is marked ON w.r.t. motifs $m_1, m_2, \ldots, m_l$, then every pair of location lists $\mathcal{L}_{m_i}, \mathcal{L}_{m_j}$, $i \neq j$ must straddle.*

   Proof. Assume this does not hold, then $\mathcal{L}_{m_i} \subseteq \mathcal{L}_{m_j}$, for some i and j. In that case the location x is marked OFF w.r.t. m_j which is a contradiction. □

   This is straightforward to see; however, it is the critical property that allows us to proceed further with the argument.

[2] A motif m with $|m| = 1$ *is* a valid motif by the Definition 3 where the first and last characters are not distinct. In practice, this may not be a useful motif, however we use this in the argument of the proof.

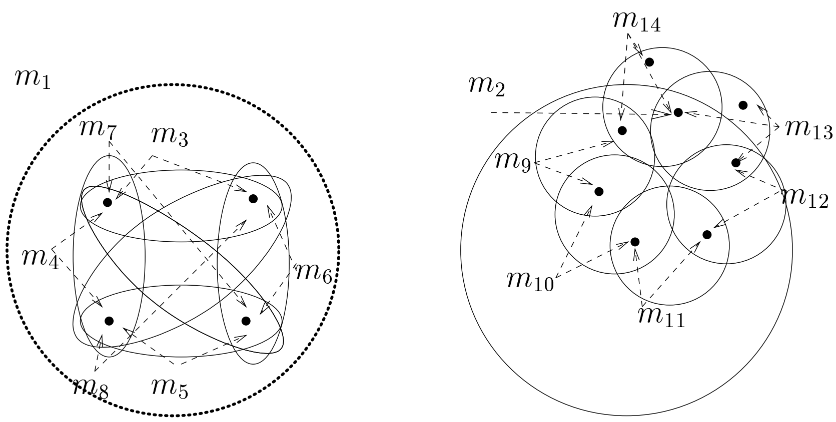

Figure 1: An example that illustrates the marking of the positions as ON/OFF w.r.t. each motif. The black dot denotes a position in the location list and the closed curves denote the location list (as a set) of a motif. The locations that are marked ON w.r.t. a motif are shown by a dashed line segment in the figure. m_1 is a redundant motif since $\mathcal{L}_{m_1} = \mathcal{L}_{m_3} \cup \mathcal{L}_{m_4} \cup \ldots \mathcal{L}_{m_8}$. Notice that none of the locations can be marked ON w.r.t m_1. However, m_2 is *not* redundant and there exists a location marked ON w.r.t m_2. (Notice that if this position did not exist in its location list, m_2 would have been redundant.)

2. For each motif m, define $c(m)$ to be the charge which is a positive integer. This is initialized to 1 for every motif. In the counting process, when there is difficulty in accounting for a motif m, a charge or count for m is assigned ot some other other motif m': thus m' would account for itself and all the other motifs whose charge it is carrying (thus m' is the *banker*, as defined in the next step, for all these other motifs).

3. For each motif m, define $B(m)$ to be the *banker* of m, which is a motif m' that is carrying the charge for m. For each m, initialize $B(m) = m$.

4. Every motif is marked LIVE/DEAD. At the initialization step every motif that is not redundant (see Claim 1) is marked LIVE. If there exists a position x that is marked ON w.r.t only one LIVE motif m, m is marked DEAD. Repeat this process until no more motifs can be marked DEAD.

   In some sense, every DEAD motif at this stage is such that there is a unique position (x of last paragraph), that can be uniquely assigned to it. The number of DEAD motifs $\leq n$.

Step 2. We begin by introducing some more definitions.

Definition 12 *(instance) An* instance *of a motif m is the motif at some location $x \in \mathcal{L}_m$ on the input string s.*

For example, let $s = abcdabed$ and let $m = ab.d$. Then one instance of m on s, shown in bold, is **abcd***abed* and the other instance is *abcd***abed**. The solid characters in the instances of m, shown with a bar, are as follows: $\overline{\mathbf{ab}}\mathbf{c}\overline{\mathbf{d}}abed$ in the first and $abcd\overline{\mathbf{ab}}\mathbf{e}\overline{\mathbf{d}}$ in the second.

Definition 13 *(i-connected) An instance of a motif m_1 is i-connected to an instance of motif m_2 if the two instances have at least i common solid characters.*

For example let $s = eabcdgeababgd$ and the motifs are $m_1 = eab..g$ and $m_2 = ab.d$. The first instance of m_1 is $\overline{\mathbf{eab}}\mathbf{cd}\overline{\mathbf{g}}exbabgd$ and the first instance of m_2 is $e\overline{\mathbf{ab}}\mathbf{c}\overline{\mathbf{d}}geababgd$ with the solid characters as shown. The two instances are 2-connected since the second (a) and third character (b) in the string are solid characters in the instances of both m_1 and m_2. The second instance of m_1 is $eabcdg\overline{\mathbf{eab}}\mathbf{ab}\overline{\mathbf{g}}d$ and the second instance of m_2 is $eabcdgeab\overline{\mathbf{ab}}\mathbf{g}\overline{\mathbf{d}}$. These two instances are *not* 1-connected since they do not share any common solid character, although they share some dot characters.

Let $\bar{m}_a^x$ be an instance of m_a where $x \in \mathcal{L}_{m_a}$. To avoid clutter we refer to an instance of motif m_a simply as $\bar{m}_a$.

Lemma 3 *Consider an instance each of a set of motifs $m_1, m_2, \ldots, m_l$ such that*

- *for that instance of m_i, the starting position $x \in \mathcal{L}_{m_i}$ is marked ON w.r.t m_i, $1 \le i \le l$, and,*
- *for every motif m_i there exists a motif m_j, $j \ne i$, $1 \le i, j \le l$, such that the two instances are 2-connected,*

then there exist distinct *positions $j_1, j_2, \ldots, j_l$ on the input string s, with the corresponding positions, $j'_1, j'_2, \ldots, j'_l$ such that $m_1[j'_1]$, $m_2[j'_2]$, $\ldots$, $m_l[j'_l]$ are solid characters.*

Proof. Assume this does not hold, then there exists instances of motifs m_{j_a}, m_{j_b}, $1 \le j_a, j_b \le l$ with $m_{j_b} \le m_{j_a}$. Consider m'_{j_a}, the sub-motif of m_{j_a} which starts at the starting position on m_{j_b} and ends at the ending position of m_{j_b}. If the position w.r.t. m'_{j_a} is ON, it is a contradiction since then the position at which m_{j_b} is incident must be marked OFF. However, if the position w.r.t. m'_{j_a} is OFF, then there exists an instance of m_{j_c} such that $m'_{j_a} \le m_{j_c}$ and that instance of m_{j_c} is marked ON. But $m_{j_b} \le m_{j_c}$ and both are ON, which is again a contradiction. □

Next, we define an *operational connectedness* on ON-marked instances of LIVE motifs m_a and m_b, called the *o-connectedness* which holds if

1. $\bar{m}_a$ is 2-connected to $\bar{m}_b$, or.
2. there exists $\bar{m}_c$, where the instance is ON-marked w.r.t LIVE motif m_c, and $\bar{m}_a$ is o-connected to $\bar{m}_c$ and $\bar{m}_c$ is o-connected to $\bar{m}_b$.

Lemma 4 *o-connectedness is an equivalence relation.*

Proof. It can be easily verified that o-connectedness is reflexive, symmetric and transitive. Thus all the ON-marked instances of the LIVE motifs can be partitioned into equivalence classes. . □

Now, we make the following claim.

Claim 3 *Using Lemmas 3 and 4, every instance of a LIVE motif m_a has a solid character at position j_a associated with it. Let $D(\bar{m}_a) = S^{[j_a, |m_a|]}(m_a)$.*

Charging Scheme. We next describe a charging (or counting) scheme by which we count the number of motifs. This is best described as an iterative process as follows.

<u>While</u> there exists position x on s such that x is marked ON w.r.t LIVE motifs $m_1, m_2, \ldots, m_l$, $l > 1$, <u>do</u> the following for $1 \le i \le l$:

1. Let $B(m_i) = D(\bar{m}_i)$ (see Step 1.3 and Claim 3).
2. $c(B(m_i)) = c(B(m_i)) + c(m_i)$ (see Step 1.2).

3. Mark m_i DEAD (see Step 1.4) [3].

We make the following claims about the while loop.

Claim 4 *The loop terminates.*

Proof. At every iteration at least two distinct LIVE motifs are marked DEAD, hence the loop must terminate. □

Claim 5 *At the end of the loop, all the LIVE motifs are such that for every pair m_i, m_j:*

1. *$\mathcal{L}_{m_i}$ and $\mathcal{L}_{m_j}$ do not straddle and*
2. *$\mathcal{L}_{m_i} \not\subset \mathcal{L}_{m_j}$, without loss of generality.*

Proof. The first condition holds obviously since otherwise the loop would not terminate since $x \in \mathcal{L}_{m_i} \cap \mathcal{L}_{m_j}$ would be marked ON w.r.t m_i and m_j. The second condition also hold obviously since if $\mathcal{L}_{m_i} \subset \mathcal{L}_{m_j}$, then motif m_i is marked DEAD (Step 1). □

Next, we need to show that the charge $c(m)$ carried by every LIVE motif m at the end of the loop, can be accounted for by $\mathcal{L}_m$. In this context we make the following observation about the charge: the iterative assignment of charges to a motif has a tree structure. This is best described using an example: see Figure 2. Each level of the tree corresponds to an iteration in the loop. For instance, the top level in the left tree denotes that at iteration 1, $B(a.bxyc..ab) = B(xyxyc..ab) = B(c...dxyc..ab) = xyc..ab$ and $c(xyc..ab) = 1 + c(a.bxyc..ab) + c(xyxyc..ab) + c(c...dxyc..ab)$. At the end of this iteration motifs $a.bxyc..ab$, $xyxyc..ab$ and $c...dxyc..ab$ are marked DEAD. At the second iteration, $B(xyc..ab) = yc..ab$ and $c(yc..ab) = 1 + c(xyc..ab)$ and motif $xyc..ab$ is marked DEAD and so on.

Claim 6 *Let L denote the number of leaf nodes (nodes with no incoming edges) in the charge tree of motif m at the end of the while loop, then $|\mathcal{L}_m| \geq L$.*

Proof. Such a claim holds since we know that by our choice of $B(.)$, if $B(m_1) = B(m_2) = \ldots = B(m_l) = m'$ then by Lemma 3 m' must have l distinct instances, each instance in a distinct equivalent class of motif instances (Lemma 4). However, the instance of m' may not be distinct from each of these instances; hence the non-leaf nodes may not be accounted for but the leaf nodes are. Hence $|\mathcal{L}_m| \geq L$. □

[3] The only exception is made when $B(m_i) = m_i$. In this case m_i remains LIVE.

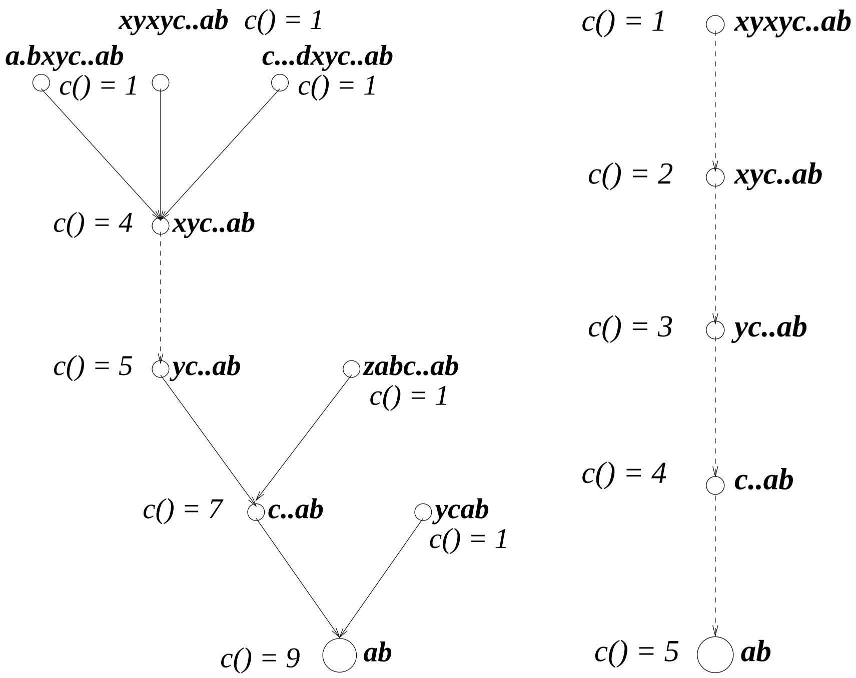

Figure 2: Two examples showing the different steps in the assignment of charge to motif ab: Every level of the tree corresponds to an iteration in the while loop. The dashed edge indicates that the motif at the "to" end of the edge could possibly be non-maximal.

At an iteration if a motif m is charged by more than one motif (or in the charge tree, the node has more than one incident edge), m is certainly maximal. However if it is charged by exactly one motif then it may or may not be maximal; if it is maximal, it *must* have an extra instance. We use the following folk-lore lemma to bound the size of I, the number of non-leaf nodes in the charge-tree.

Lemma 5 *Given a tree $\mathcal{T}$, where each node, except the leaf nodes, must have at least two children, the number of non-leaf nodes, I is no more than the number of leaf nodes L.*

We are not interested in counting non-maximal motifs and these are the only motifs that contribute to a single child for a node in a tree. Thus the number of maximal motifs that were marked LIVE at the start of Step 2 is no more than $2n$, using Claim 6 and Lemma 5.

Using Step 1.4, we have number of maximal and non-redundant motifs $\leq (n + 2n)$. This concludes the proof of the theorem. $\mathbf{\Delta}$

Corollary 1 *Given a string s, the basis $\mathcal{B}$ is unique.*

This follows immediately by viewing motifs as location lists and treating these as overlapping sets. $\mathbf{\Delta}$

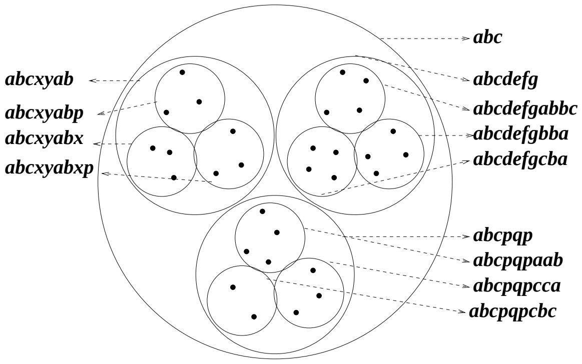

Figure 3: An example that illustrates the relationship between location lists (as sets) of motifs with *no* dot characters. Notice that the sets do not straddle and the motif corresponding to a subset of another set is such that the latter a sub-motif of the former.

Corollary 2 [4] *Given a string s, let $\mathcal{M}$ be the set of all maximal (but possibly redundant) motifs which are such that they have only solid characters in them. Then $|\mathcal{M}| \leq n$.*

When the motifs are such that they have no dot characters, it can be easily verified that the following hold. For any pair of motifs m_a and m_b,

1. $\mathcal{L}_{m_a}$ and $\mathcal{L}_{m_b}$ do not straddle, and,
2. if $\mathcal{L}_{m_a} \subset \mathcal{L}_{m_b}$, then there exist j such that $m_b = m_a[1..j]$. For example $m_a = abxyz$, and $m_b = ab$ with $j = 2$. See Figure 3 for an illustration. Thus every redundant motif has a unique solid position associated with it (note that this is not true in the general case).

Using the same arguments for the main theorem, but also counting the redundant motifs we obtain a bound of n on the total number of maximal motifs. $\mathbf{\Delta}$

Corollary 3 *Given a string s, let $\mathcal{M}$ be the set of all motifs. Let $\mathcal{M}' \subseteq \mathcal{M}$ be an arbitrary set of maximal motifs. Then the basis $\mathcal{B}'$ of $\mathcal{M}'$ is such that $|\mathcal{B}'| < 3n$.*

This follows immediately since the proof of the theorem does not use the fact that $\mathcal{M}$ is the set of *all* motifs of s. It simply works on the location lists (sets) of this special set of maximal motifs. $\mathbf{\Delta}$

[4] This result has been proved independently in [Gus97].

5 Algorithm to detect the O(n) Irredundant Motifs

The next natural question is whether the irredundant maximal motifs can be detected in polynomial time.

We present the following iterative algorithm. The input parameters are: s a string on some alphabet Σ, integers k and d. Each motif must appear at least k times in s and the maximum number of dot characters between two consecutive solid characters is at most d. Notice that $0 \leq d \leq n-2$ where $|s| = n$.

1. Construct all the motifs with exactly two solid characters. Let the set of these motifs be M, then $|M| \leq |\Sigma|^2 n$. Construct a graph with every vertex v_m corresponding to a motif m and a directed edge exists from v_{m_p} to v_{m_c} if $\mathcal{L}_{m_c} \subseteq \mathcal{L}_{m_p}$ (or $m_p \leq m_c$).
2. For every vertex v_{m_p}, consider the vertices incident on all the outgoing edges $v_{m_{c1}}, v_{m_{c2}}, \ldots, v_{m_{cl}}$. If $\mathcal{L}_{m_p} = \mathcal{L}_{m_{c1}} \cup \mathcal{L}_{m_{c2}} \cup \ldots \cup \mathcal{L}_{m_{cl}}$, then we do the following.
   (1) Case $l = 1$:
   Remove the vertex $v_{m_c 1}$ and replace the associated motif m_p of vertex v_{m_p} with $m = m_p \bigotimes m_{c1}$.
   (2) Case $l > 1$:
   Remove the vertex v_{m_p} and all its incident edges (since this is a redundant motif).
3. Define a pair on motifs m_a and m_b to be compatible if $m_a[|m_a|] = m_b[1]$ holds. For every pair of compatible motifs m_i, m_j (corresponding to vertices in the graph), let $\mathcal{L}'_{m_i} = \mathcal{L}_{m_i} + |m_i|$. Construct $\mathcal{L}'_{m_i} \cap \mathcal{L}_{m_j} = \mathcal{L}_{m_{new}}$. Motif m_{new} is obtained by concatenating the compatible motifs m_i and m_j: $m_{new} = m_i + m_j$. For example if $m_i = ab..d$ and $m_j = d.e.ac$, then $m_{new} = ab..d.e.ac$.
   Update the graph as follows.
   (a) Introduce a vertex corresponding to motif m_{new}
   (b) Introduce a directed edge from v_{m_i} to $v_{m_{new}}$.
4. Repeat Steps 2 and 3 until $\mathcal{L}_{new} = \{\}$, for every pair of compatible motifs m_i, m_j with $m_{new} = m_i + m_j$.

The following is a proof of correctness of the algorithm.

Lemma 6 *Let M be all the motifs produced by the algorithm. Then $M = \mathcal{B}$, the basis set.*

Proof. Let $\mathbf{G}(\mathcal{B})$ be the set of all the redundant maximal motifs generated by $\mathcal{B}$ and let $\mathbf{G}(M)$ be the set of all the redundant maximal motifs generated from M. We need to show $\mathbf{G}(\mathcal{B}) = \mathbf{G}(M)$.

Let $m \in \mathbf{G}(M)$, then clearly $m \in \mathbf{G}(\mathcal{B})$ since all the motifs produced by the algorithm are maximal motifs due to pruning at Step 2 of the algorithm and see Lemma 8 about enumerating $\mathbf{G}()$. Thus $\mathbf{G}(M) \subseteq \mathbf{G}(\mathcal{B})$.

Let $m \in \mathbf{G}(\mathcal{B})$. Let us assume the contrary, that is, $m \notin \mathbf{G}(M)$.

1. Case 1: m is not redundant.
   We make the following straightforward observations about the algorithm.

   Claim 7 *Let the algorithm produce the set of motifs M in p iterations. For every motif m_a on the input s, there exist motifs $m_a^1, m_a^2, \ldots, m_a^p = m_a$ at each iteration respectively such that the following hold:*

   (a) $m_a^{i+1} = m_a^i + m_a'$ for some motif m_a' or an empty string and
   (b) $\mathcal{L}_{m_a^1} \supseteq \mathcal{L}_{m_a^2} \supseteq \ldots \supseteq \mathcal{L}_{m_a^p}$

   Proof. At the very first iteration (Step 1 of the algorithm) the algorithm produces all the largest possible location lists, assuming that the motif m of interest has $|m| \geq 2$ [5]. In other words there exists no motif m_{queer} such that $\mathcal{L}_{m_{queer}} \supset \mathcal{L}_{m_b}$ where m_b is produced in Step 1. Hence there must exist m_k such that $\mathcal{L}_{m_k} \supset \mathcal{L}_m$ and m_k was pruned as a redundant motif (Step 2.2 of the algorithm) at some iteration i_1.

   Also $m = m_k + m'$ for some motif m'. If m' is an empty string, $m = m_k$ which is redundant and that is a contradiction.

   By Claim 7, there must exist m^{i_1} corresponding to m' at the end of iteration i_1. If m^{i_1} is redundant, it leads to a contradiction since then m is also redundant. Consider the case when m^{i_1} is not redundant. Then the algorithm must "concatenate" (Step 3 of the algorithm) compatible motifs m_k and m^{i_1} at iteration i_1 to produce m_b at the next iteration with $\mathcal{L}_m \subseteq \mathcal{L}_{m_b} \subseteq \mathcal{L}_{m_k}$, which is again a contradiction by the choice of m_k at iteration i_1.

   Hence $m \in \mathbf{G}(M)$.

2. Case 2: m is redundant.
   Then there exist motifs $m_1, m_2, \ldots, m_l$ each of which is not redundant that renders m redundant. By using Case 1, we can show that each of $m_1, m_2, \ldots, m_l$ must be in $\mathbf{G}(M)$. Hence m must be in $\mathbf{G}(M)$.

Also, it is easy to see that all the motifs in M are maximal and mutually irredundant due to the pruning at Step 2. Thus by Corollary 1 M is the basis set. This concludes the proof. Δ

[5] However, if it requires $|m| \geq 1$, at this first step we produce all the motifs of size one or single character motifs and "grow" them appropriately in the algorithm.

Lemma 7 *The algorithm takes $O(n^3 \log n)$ time.*

Proof. The point to note here is that the nodes in the graph or the motifs being constructed do not grow exponentially. At the very first iteration there are $O(n)$ motifs (Step 1) and at the end of Step 2, there are again $O(n)$ motifs, using Corollary 3. Thus there are $O(n^2)$ comparisons made at each iteration. Since the location list can be no more that n, the amount of work done is $O(n^3 I)$, where I is the number of iterations. But $I = logL$ where L is the maximum number of solid characters in a motif, since the motif grows by concatenating two smaller motifs. But $L < n$, hence the result. Δ

6 Generating Redundant Motifs: Annotated Trie

The last section describes a polynomial time algorithm to produce the unique basis set $\mathcal{B}$ for a string s. The elements of $\mathcal{B}$ are maximal and not redundant. However, it is possible that there exist applications where the user requires all the motifs – either non-maximal or redundant, or, both. For such applications, we present an efficient and systematic way of generating non-maximal or redundant motifs or both by storing the motifs in $\mathcal{B}$ in a trie structure [AHU83], that is suitably annotated to produce the location lists. This procedure will implicitly use the generating operators described in Section 3.1.

We store the motifs of the basis $\mathcal{B}$ in a trie structure as shown in Figure 4. This trie is annotated with the location lists so that every redundant motif can be read off the trie along with the location list. The data structure ensures that the redundant motifs can be obtained in time linear with respect to the output. The algorithm is relatively straightforward and we describe it below.

For each path (reading off the labels on the edges) $\mathcal{P}_1$ from the root node of the trie $\sigma s_1 s_2 \ldots s_l$, $\sigma \in \sum$, $s_i \in \sum + \{.\}$

For each distinct and longest [6] path $\mathcal{P}_2$ from the root node, $\sigma s'_1 s'_2 \ldots s'_k$, $s'_i \in \sum + \{.\}$, satisfying the following:

(a) $n \leq l$,

(b) $s'_k = s_k \in \sum$ (since the last character of a valid motif must be a solid character), and

(c) either $s_i = s'_i$ or one of them is the dot character.

1 Let $m_1 = \sigma s_1 s_2 \ldots s_k$ and $m_2 = \sigma s'_1 s'_2 \ldots s_k$. Construct $m_{new} = m_1 \bigoplus m_2$. Further $\mathcal{L}_{new} = \mathcal{L}_1 \cup \mathcal{L}_2$ where $\mathcal{L}_1$ is the annotation at the node with the incoming edge labeled with s_l and similarly $\mathcal{L}_2$. Notice that this is the generating operator as defined in Section 3.1 and when $n < l$, the Prefix operator has been implicitly used on m_2.

[6] This means that the path cannot be grown any further without having to make a choice from alternate paths.

2 Project m_{new} onto the trie updating the new node with $\mathcal{L}_{new}$.

See Figure 5 for an illustration of the algorithm. We make the following claim about the algorithm.

Lemma 8 *The algorithm produces all maximal redundant motifs.*

Proof. It is easy to see that the motif is maximal since $\mathcal{P}_2$ is the *longest* compatible (so that the binary operator $\bigoplus$ is valid) path.

Next, we show that *all* the maximal and redundant motifs are generated. Any redundant motif m can be generated from $m_1, m_2, \ldots, m_l \in \mathcal{B}$ for some $l > 1$. But the binary operator $\bigoplus$ is commutative and associative and the unary prefix operator $P^\delta()$ distributes over the binary operator (see Section 3.1). Thus, although we consider only pairwise paths in the algorithm (for ease of implementation), the new motif is projected onto the trie and this process can be successively used to obtain m. Δ

7 Applications

In this section we give two examples of possible applications of the results of the preceding sections.

7.1 Association Rules

We describe an important problem in the context of data mining which has been addressed in [AMS$^+$95]. See the citations therein for related literature.

Definition 14 *(k-association rule) Given n transactions (or records) each with F fields. Each of the field f_j, $1 \leq j \leq F$, takes one of d^j discrete values. A two tuple (f, v_f) denotes the assignment of the value v_f to the field f. An association rule or a pattern r, is a collection of two tuples that hold in at least k transactions, given a positive integer k. $\mathcal{L}_r$ is the set of transactions in which r holds.*

r is maximal if and only if there exists no other association rule r' such that $r \subset r'$ and $\mathcal{L}_r \subseteq \mathcal{L}_{r'}$.

It is straightforward to see that the problem of extracting all maximal k-association rules from n transactions with F fields can be mapped onto a special case of discovering k-motifs from a string of size nF. This string can be defined by fixing the order of fields in a transaction and considering only the values each field takes In this context we can introduce the notion of *irredundant k-association rules.* In a strict sense, the total number of maximal k-association rules can be no more than 2^{d^F} where the F fields take d values, which for the purposes of complexity analysis is only a constant (!)

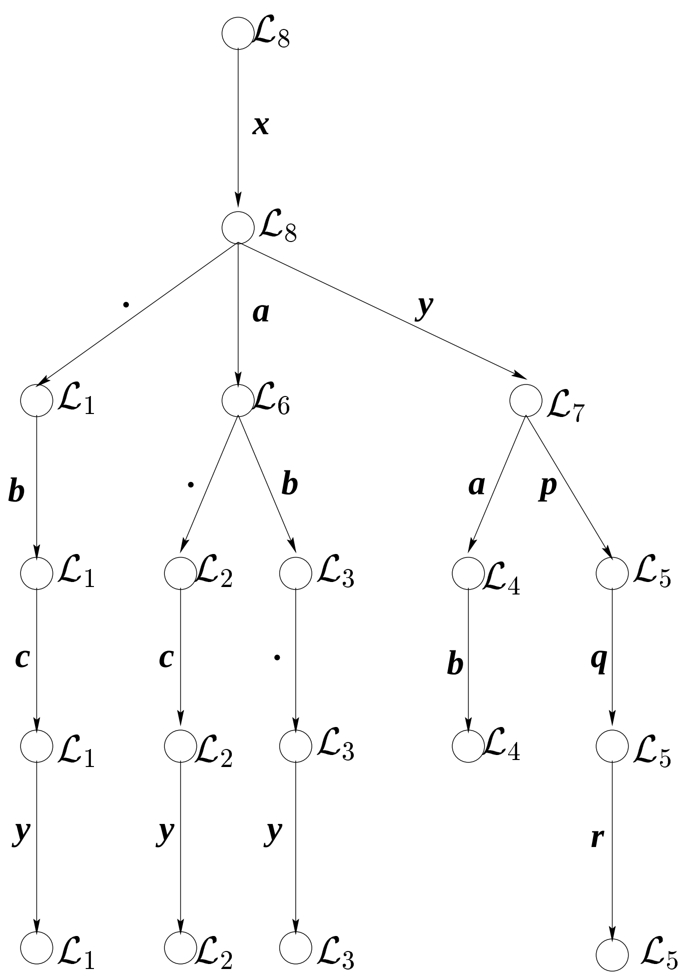

Figure 4: Annotated trie corresponding to maximal and non-redundant motifs $x.bcy$, $xa.cy$, $xab.y$, $xyab$, $xypqr$. Each leaf node of this edge-labeled trie is annotated with the location lists as shown. $\mathcal{L}_1 = \mathcal{L}_{x.bcy}$, $\mathcal{L}_2 = \mathcal{L}_{xa.cy}$, $\mathcal{L}_3 = \mathcal{L}_{xab.y}$, $\mathcal{L}_4 = \mathcal{L}_{xyab}$, $\mathcal{L}_5 = \mathcal{L}_{xypqr}$, $\mathcal{L}_6 = \mathcal{L}_2 \cup \mathcal{L}_3$, $\mathcal{L}_7 = \mathcal{L}_4 \cup \mathcal{L}_5$, $\mathcal{L}_8 = \mathcal{L}_1 \cup \mathcal{L}_6 \cup \mathcal{L}_7$,

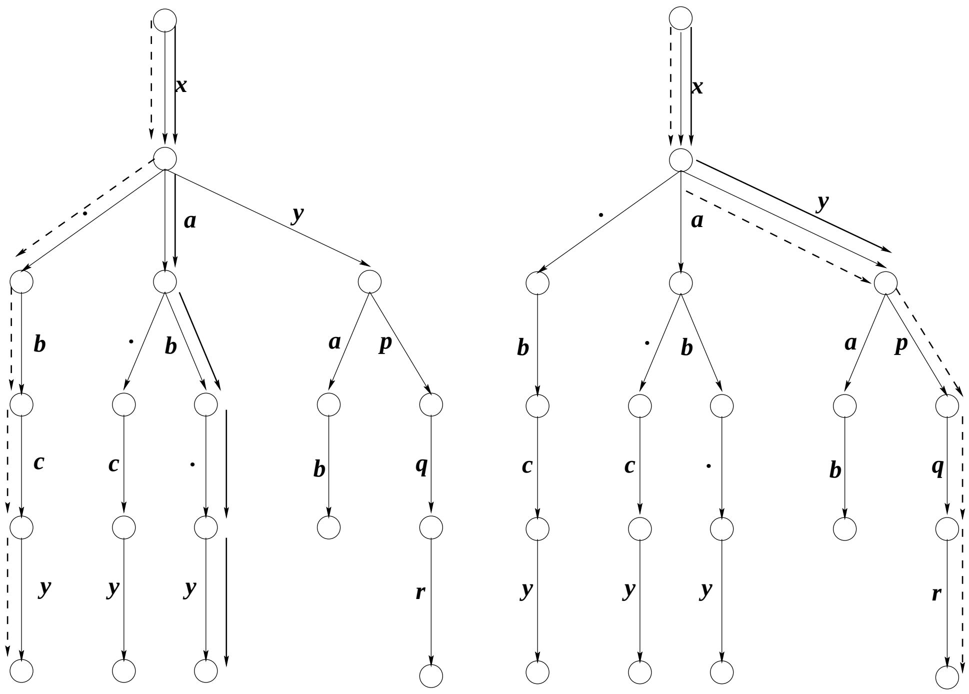

Figure 5: An example showing the construction of the redundant motifs from the trie of the basis motifs. $\mathcal{P}_1$ is shown by a dashed solid line and $\mathcal{P}_2$ is shown by solid line superimposed on the trie of the previous figure. In the first case $\mathcal{P}_1 = x.bcy$ and $\mathcal{P}_2 = xab.y$ giving the new motif $x.b.y$. In the second case $\mathcal{P}_1 = xypqr$, $\mathcal{P}_2 = xy$ giving the new motif xy and the associated location list is $\mathcal{L}_7$. Note that in the latter case $\mathcal{P}_2$ can be extended in at least two ways (that may or may not be compatible with $\mathcal{P}_1$).

unless d or F is a function of n. However, for practical purposes this number may be considered extremely large. In this context, it might be economical to use the notion of irredundancy to cut down the large number of rules by using a suitable modification of the algorithm of Section 5. All the redundant k-association rules anyways can be constructed from the *basis* set.

7.2 Multiple Sequence Alignment

Another useful application is alignment of multiple sequences based on motif discovery. It has been shown in [PFR99b] that irredundant motifs give the same best alignment as that of all maximal motifs and since their number is only polynomial as opposed to exponential, it offers validity to this approach. See [PFR99a] for further details on this application.

Acknowledgments

I am grateful to Rohit Parikh, Saugata Basu and Ian Jermyn for very fruitful discussions. I am indebted to Auckland, the city of sails, for providing the inspiration to overcome one of the major bottlenecks in the main theorem of the paper. Special thanks to Gustavo Stolovitzky and Andrea Califano for their unrelenting enthusiasm and keen interest in the work.

References

[AHU83] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structure and Algorithms.* Addison-Wesley Publishing Company, 1983.

[AMS+95] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Advances in knowledge discovery and data mining, chapter 12,. In *Fast Discovery of Association Rules.* AAAI/MIT Press, MA, 1995.

[Gus97] D. Gusfield. *Algorithms on strings, trees and sequencess: computer science and computational biology.* Cambridge University Press, New York, 1997.

[GYW+97] Y. Gao, M. Yang, X. Wang, K. Mathee, and G. Narasimhan. Detection of HTH motifs via data mining. *Int'l Conference on Bioinformatics*, 1997.

[Par98] L. Parida. *Algorithmic Techniques in Computational Genomics.* PhD thesis, Courant Institute of Mathematical Sciences, New York University, September 1998.

[PFR99a] L. Parida, A. Floratos, and I. Rigoutsos. An approximation algorithm for alignment of mulitple sequences using motif discovery. *Journal of Combinatorial Optimization*, 1999.

[PFR99b] L. Parida, A. Floratos, and I. Rigoutsos. MUSCA: An algorithm for constrained alignment of multiple data sequences. *Genome Informatics*, No 9:112–119, 1999.