Best Fitting Fixed-Length Substring Patterns for a Set of Strings*

Hirotaka Ono[1] and Yen Kaow Ng[2]**

[1] Kyushu University, Department of Computer Science and Communication Engineering, 6-10-1, Hakozaki, Fukuoka, 812-8581, Japan
ono@csce.kyushu-u.ac.jp

[2] Kyushu Institute of Technology, Graduate School of Computer Science and Systems Engineering, Iizuka, 820, Japan
kalngyk@daisy.ai.kyutech.ac.jp

Abstract. Finding a pattern, or a set of patterns that best characterizes a set of strings is considered important in the context of Knowledge Discovery as applied in Molecular Biology. Our main objective is to address the problem of "over-generalization", which is the phenomenon that a characterization is so general that it potentially includes many incorrect examples. To overcome this we formally define a criteria for a most fitting language for a set of strings, via a natural notion of *density*. We show how the problem can be solved by solving the membership problem and counting problem, and we study the runtime complexities of the problem with respect to three solution spaces derived from unions of the languages generated from fixed-length substring patterns. Two of these we show to be solvable in time polynomial to the input size. In the third case, however, the problem turns out to be NP-complete.

* This work was partially supported by the Scientific Grant in Aid of the Ministry of Education, Science, Sports, Culture and Technology of Japan.

** The second author is supported by the the Japanese Government Scholarship of the Ministry of Education, Science, Sports, Culture and Technology of Japan.

1 Introduction

Extracting a common description for a set of strings is a central objective in many fields of research such as Knowledge Discovery and Artificial Intelligence. This description is sometimes expressed in a form of languages generated by some patterns, such as the one introduced by Angluin [1]. In the context of such patterns, the problem has received very intensive treatment in the field of Molecular Biology, where the patterns discovered are commonly called *consensus patterns* [4, 6].

A common problem in finding a language that best characterizes a set of examples is that of avoiding "over-generalization". A language that includes all the correct examples is called a *generalization* of the examples. It is called an *over-generalization* when it is overly general so that many potentially incorrect examples are also included. It is in general difficult to avoid over-generalization in the absence of incorrect examples. To overcome this Angluin [1] and Shinohara [11] considered finding languages which are in a sense "minimal". More precisely, they considered finding a pattern language L covering a set of strings, such that no other pattern language that covers the strings is a proper subset of L. The solution space was extended to a subclass of the (bounded) unions of pattern languages by Arimura *et al.* [3, 10, 12], using a polynomial time algorithm called k-MMG. However, no algorithm has yet been proposed for the general case, and the output of k-MMG is dependent on the search path traversed [2]. Also, we note that there is another notion of "minimality" not guaranteed using their approach. We illustrate this with an example: let $S = \{$"abc", "bca"$\}$, and consider the language generated by the pattern "$*a*$" ("$*$" can be replaced with any string). While it does not contain any smaller proper subset which also contain S, it is arguably an inferior characterization compared to "$*bc*$".

We suggest a strategy to overcome this shortcoming. Our idea is to exclude as many as possible of all the strings which has not appeared in the input strings — up to a certain length. More precisely, given any input set S of strings, we consider patterns where their languages include S, while excluding as many as possible of the strings outside of S that are not longer than the longest string in S. In this sense the solution we aim at is "minimal" with respect to the set of all the strings of length up to that of the longest string seen. We define a notion of "density" which naturally expresses this idea, and ask the following questions: (1) with respect to a language and a set of examples, how efficiently can this density measure be computed, and (2) with respect to a set of examples, are there efficient computations for finding a language with the highest density? In this paper we study this measure as applied to a subclass of the pattern languages, namely the *substring patterns*.

This paper is structured as follows. We first introduce our notion of "density" and formally define the problem of finding the most fitting pattern languages for a set of strings. We then show how such densities can be computed in polynomial-time for the languages generated by the substring patterns. Finally, we study the hardness of finding a language of substring patterns that maximizes this density.

2 Preliminaries

Let N and N^+ denote the set of natural numbers and the set of positive integers, respectively. Cardinality of a set S is denoted by $|S|$. max(.), min(.) denote maximum and minimum of a set, where by convention $\max(\emptyset) = 0$ and $\min(\emptyset) = \infty$. A *word* over a non-empty alphabet A is a finite string of symbols taken from A. The *empty word* is a null string. The symbols A^+, $A^{=m}$ and $A^{\leqq m}$ denote the sets of non-empty words, words of length m, and words of length m or less, respectively. Let $A \setminus B$ denote the set $\{x \mid x \in A \text{ and } x \notin B\}$.

Let Σ be a fixed finite alphabet with at least 2 elements, and $V = \{x_1, x_2\}$ be an alphabet disjoint from Σ. Elements in Σ are called *constants* and elements in V are called *variables*. Each constant and variable has unit length. A non-empty word over Σ is called a *constant word*, or simply a *string*. A non-empty word over $V \cup \Pi$ is called a *pattern*. The concatenation of two patterns p_1 and p_2, with p_2 as suffix, is written p_1p_2. An arbitrary string of length m is written "$?^m$", and "$?^1$" is written simply "?". The length of a pattern p is denoted by $|p|$. For a set of patterns P, the length of the longest word in it is denoted by $\lceil P \rceil$, the length of the shortest word is denote by $\lfloor P \rfloor$, while the sum of all the lengths is denoted by $\parallel P \parallel$.

Let $\boldsymbol{P}$ denote the set of all patterns. A pattern is *regular* if each variable in it appears at most once. Let $\boldsymbol{RP}$ denote the set of all regular patterns. A pattern is a *substring pattern* just in case it is of the form "x_1sx_2" and s is a non-empty word over Σ. Let $\boldsymbol{Psub}$ denote the set of all substring patterns. We also denote the set of all substring patterns "x_1sx_2" where $|s|$ is of length l by $\boldsymbol{Psub}_l$. For each $l \in N^+$, we let $\boldsymbol{Psub}_l^* = \{P \mid P \subseteq \boldsymbol{Psub}_l\}$. For a set T of strings, $P(T)$ denotes a set of substring patterns constructed from T, i.e., $P(T) = \{x_1sx_2 \mid s \in T\}$. Two strings t_1 and t_2 are said to *overlap* if either some suffix of t_1 matches some prefix of t_2, or some suffix of t_2 matches some prefix of t_1. For a set T of strings, we say T is *non-overlapping* if no string in T overlaps with itself (except when fully overlapping), and any two strings in T are non-overlapping.

A *substitution* is a homomorphism from pattern to pattern that maps every constant symbol to itself. For a pattern p and a substitution θ, we denote by $p\theta$ the image of p by θ. The language of a pattern p, written as $L(p)$, is the set $\{w \in \Sigma^+|$ exists substitution θ such that $w = p\theta\}$. For a set P of patterns, $L(P)$ denotes the language $\{w \in L(p) \mid p \in P\}$ and $\mathcal{L}(P)$ denotes the class of languages $\{L(p) \mid p \in P\}$. A set of patterns P is said to *cover* a set S of strings if $S \subseteq L(P)$.

3 Maximum Density Pattern for a Set of Strings

We consider the problem of finding a set of patterns such that their languages best represent a given set of strings. Since we consider unions of sets of pattern languages, this problem can be compared to those studied in [3, 5]. We introduce a measure for how well a set of patterns cover a given set of strings. For a set of patterns P and a set of strings S, we define the *density of* P *with respect to* S,

$$d(P,S) = \frac{|L(P) \cap S|}{|L(P) \cap \Sigma^{\leqq \lceil S \rceil}|}.$$

For a set of strings S and two sets of patterns P and Q, both which covers S individually, we argue that P generalizes S better than Q if $d(P,S) > d(Q,S)$, since Q includes more examples not in S within the size of the longest example that is known to exist. For example, in the case where the input strings S is $\{$"abc'', "$bca''$$\}$, "$x_1bcx_2$" would be chosen over "$x_1ax_2$", since $d(L($"x_1bcx_2"$), S) > d(L($"x_1ax_2"$), S)$. Based on this, we introduce the following problem:

Problem Maximum Density Patterns (MDP)
Input: A set S of strings over $\Sigma^{\leqq \lceil S \rceil}$, and a set of sets of patterns $\boldsymbol{\mathcal{P}}$.
Output: A set of patterns $P \in \boldsymbol{\mathcal{P}}$ covering S and maximizing $d(P,S)$.

The choice of the solution space $\boldsymbol{\mathcal{P}}$ fundamentally affects the problem. To ensure that MDP output a target generalization $P \in \boldsymbol{\mathcal{P}}$ on input S, no other generalization $P' \in \boldsymbol{\mathcal{P}}$ covering S should have $d(P', S) \geq d(P, S)$. For example consider the case where $\boldsymbol{\mathcal{P}}$ is the set of all finite sets of patterns from $\boldsymbol{P}$ or $\boldsymbol{RP}$, for which given any input S, an optimal solution is simply S itself. Such an output requires virtually no computation, and probably makes very poor generalization of the input. Ideally, the solution space $\boldsymbol{\mathcal{P}}$ should be such that any $P \in \boldsymbol{\mathcal{P}}$ maximizing $d(P,S)$ is an interesting generalization.

3.1 Computing density

Consider the computation for an instance of the MDP problem. Since we require that $S \subseteq L(P)$, we have $|L(P) \cap S| = |S|$. To solve MDP with input $\boldsymbol{\mathcal{P}}$ it suffices that we find among all sets of patterns $P \in \boldsymbol{\mathcal{P}}$ for one where: (1) $S \subseteq L(P)$, and (2) $|L(P) \cap \Sigma^{\leqq \lceil S \rceil}|$ is minimized. Condition (1) cannot be computed efficiently if $\boldsymbol{\mathcal{P}} = \{\{p\} \mid p \in \mathcal{L}(\boldsymbol{P})\}$ (refer Section 2 for pattern class $\boldsymbol{P}$), since it is known that the membership problem is NP-complete for the class $\mathcal{L}(\boldsymbol{P})$ [1]. However, for any set of strings S and any $P \subseteq \boldsymbol{RP}$, whether $S \subseteq L(P)$ can be computed in time $O(|P| \cdot \parallel S \parallel + |S| \cdot \parallel P \parallel)$ [11].

Condition (2) can be solved by counting $|L(P) \cap \Sigma^{\leqq \lceil S \rceil}|$. For regular patterns, counting the number of strings generated by a set of patterns can be done efficiently by constructing a DFA that accepts strings in its languages, and then counting the number of distinct paths of up to a given length it accepts. This gives us the following.

Theorem 1 ([9, 7]). Given a set P of regular patterns and $m \in N^+$, there exists an algorithm which computes the total number of strings in $L(P)$ of length up to m in time $O(m \cdot \parallel P \parallel \cdot \min(\{|\Sigma|, \parallel P \parallel\}))$. □

3.2 Comparing densities

Another way to solve Condition (2) is by comparing the number of strings generated. In this section we show how this can be done very efficiently for any two substring patterns of the same length.

For substring patterns, there is a recurrence relation which allows us to count the strings using a dynamic programming approach. The following algorithm (*CountStrings*) computes, given as input a set of strings T and $m \in N^+$, the total number of unique strings of length up to m within $L(P(T))$. We assume that no string in T is a substring of some other string in T. Otherwise, the strings that are substring of some other string in T can be removed in time $O(|T|^2 \lceil T \rceil)$, and their removal will not affect the computation of $|L(P(T)) \cap \Sigma^{\leqq \lceil S \rceil}|$.

$CountStrings(T, m)$
For each r from 1 to m,
  For each string $t \in T$ where $|t| \leq r$,
    For each r' from $r - |t| + 1$ to $r - 1$, calculate $S(T, t, r, r')$.
  Calculate $\Phi(T, r)$.
Output $\sum_{r=1}^{m} \Phi(T, r)$.

For each $r \in N^+$, $\Phi(T, r)$ is the total number of strings of length r in $L(P(T))$. For $t \in T$, r, $r' \in N^+$ where $r > r'$, $S(T, t, r, r')$ is the number of strings in $L(\text{“}x_1 t\text{”})$ of length r which does not begin with some string of length r' in $L(P(T))$. It is easy to see that $S(T, t, r + |t|, r) = |\Sigma|^{r-|t|} - \Phi(T, r)$. The following is the computation of $S(T, t, r, r')$. For a string t and $n \in N$, prefix(t, n) denotes the prefix of t of length n, while suffix(t, n) denotes the suffix of t of length $|t| - n$.

$S(T, t, r, r')$
If $|t| > r$, output 0.
Else if $|t| = r$, output 1.
Otherwise, for each $t' \in T$, we try to find all $0 < n < |t'|$ such that
  (a) suffix(t', n) matches some prefix of t,
  (b) $n + |t| \leq r$,
  (c) $|t| - (|t'| - n) \geq r - r'$,
  and collect all such prefix(t', n) into a set $A_{t,r,r'}$.
We then remove from $A_{t,r,r'}$ all the strings for which some of its suffixes
  are also in $A_{t,r,r'}$. (Hence $A_{t,r,r'}$ is a set of shortest prefix strings of T
  where the suffix part is consistent with t.)
Let $B_{t,r,r'} = \{(t', \text{suffix}(t', n)) \mid \text{prefix}(t', n) \in A_{t,r,r'}\}$.
Output $|\Sigma|^{r-|t|} - \Phi(T, r - |t|) - \sum_{(t', t'') \in B_{t,r,r'}} S(T, t', r - |t| + |t''|, r - |t|)$.

For each r, the following is the computation of $\Phi(T, r)$.

$\Phi(T, r)$
Initialize ϕ to $|\Sigma| \cdot \Phi(T, r - 1)$ if $r \geq 1$, 0 otherwise.
  For each $t \in T$ where $|t| \geq r$, add $S(T, t, r, r - 1)$ to ϕ.
Output ϕ.

Theorem 2. Given a set $P(T)$ of substring patterns and $m \in N^+$, $m \gg \lceil T \rceil$, *CountStrings* computes the total number of strings in $L(P(T))$ of length up to m in time $O(m \cdot \parallel T \parallel^2)$. □

Note that its runtime is comparable to the algorithm in Theorem 1 for the cases where $|\Sigma|$ is large. A straightforward analysis of the recurrence relation in *CountStrings* gives us the following.

Lemma 1. *Let* $|\Sigma| \geq 3$*,* $x \in N$ *and* $d \in N^+$*. For all* t*,* $t' \in \Sigma^*$ *where* $|t| = |t'| > d$*,* $S(\{t\}, t, x+d, x) > \sum_{0<i<d} S(\{t'\}, t', x+i, x)$*.*

Lemma 2. *Let* $|\Sigma| \geq 4$*,* $x \in N$ *and* $d \in N^+$*. For all* t*,* t'*,* $t'' \in \Sigma^*$ *where* $|t| = |t'| = |t''| > d$*,* $|S(\{t\}, t, x+d, x) - S(\{t'\}, t', x+d, x)| \leq S(\{t''\}, t'', x+1, x)$*.*

Since $|T| = 1$, for all t, r and r', for all (t, t_1), $(t, t_2) \in B_{t,r,r'}$, $|t_1| = |t_2|$ implies $t_1 = t_2$. That is, only the length of the overlap is needed. Let $\mathbf{B}(t, r) = \{|t''| \mid (t, t'') \in B_{t,r,r-1}\}$. It is clear that $\max(\mathbf{B}(t, r)) < |t|$. Note that for any string t and $r \geq 2|t| - 1$, $\mathbf{B}(t, r + 1) = \mathbf{B}(t, r)$. For this reason we also let $\mathbf{B}(t)$ denote $\mathbf{B}(t, 2|t| - 1)$.

Theorem 3. *Let* $|\Sigma| \geq 4$*. For all* t*,* $t' \in \Sigma^*$ *where* $|t| = |t'|$*, if* $\mathbf{B}(t) = \mathbf{B}(t')$*, then for all* $r \in N$*,* $\Phi(\{t\}, r) = \Phi(\{t'\}, r)$*. Otherwise without loss of generality let* $\max(\mathbf{B}(t) \setminus \mathbf{B}(t')) < \max(\mathbf{B}(t') \setminus \mathbf{B}(t))$*, in which case*
(1) $\Phi(\{t\}, r) = \Phi(\{t'\}, r)$ *for* $r < 2|t| - \max(\mathbf{B}(t') \setminus \mathbf{B}(t))$*, and*
(2) $\Phi(\{t\}, r) > \Phi(\{t'\}, r)$ *for* $r \geq 2|t| - \max(\mathbf{B}(t') \setminus \mathbf{B}(t))$*.*

From Theorem 3, it is clear that for any strings t, t' of the same length, it suffices that we compute the sets $\mathbf{B}(t)$ and $\mathbf{B}(t')$ to decide if $CountStrings(\{t\}, m)$ $> CountStrings(\{t'\}, m)$ for any $m \in N^+$.

4 Time Complexities of the MDP problem

In this section we study the runtime complexities for MDP for three related cases. We examine the classes derived from unions of languages in $\mathcal{L}(Psub_l)$ for some $l \in N^+$. That is, the languages generated from substring patterns of a fixed length l. We are interested in both the case where the number of languages allowed in a union is bounded, and the case where it is unbounded.

We first look at the case where no union is allowed.

4.1 The case of $\boldsymbol{\mathcal{P}} = \{\{p\} \mid p \in \boldsymbol{Psub_l}\}$

Theorem 4. *Given* $l \in N^+$ *and* Σ *where* $|\Sigma| \geq 4$*. Let* $\boldsymbol{\mathcal{P}} = \{\{p\} \mid p \in \boldsymbol{Psub_l}\}$ *and* $S \subseteq \Sigma^*$*. There exists an algorithm that solves the* MDP *problem in time* $O(\min(\{\parallel S \parallel, |\Sigma|^l\}) \cdot (l^2 + l|S| + \parallel S \parallel))$*.*

Proof. We first consider the total possible candidates of length l. Since ideally we want each pattern in the solution to cover some strings in S, we choose only substrings of strings in S for candidates. There are possibly $\parallel S \parallel$ such substrings, of which at most $|\Sigma|^l$ are unique. This gives us a total of $\min(\{\parallel S \parallel, |\Sigma|^l\})$ candidates. We denote this candidate set by C. It is easy to verify, using Theorem 3, that the following algorithm (*SolveMDP*) find a set from among C which fulfills the requirement of the MDP problem.

SolveMDP(C, S)
1. Let $maxSet = \{-1\}$.
2. For each candidate t in C where $S \subseteq L(P(\{t\}))$
3. Compute $\mathbf{B}(t)$. If $\max(\mathbf{B}(t) \setminus maxSet) > \max(maxSet \setminus \mathbf{B}(t))$ and $\lceil S \rceil \geq 2|t| - \max(\mathbf{B}(t) \setminus maxSet)$, set $maxSet$ to $\mathbf{B}(t)$ and let $p = \{t\}$.
4. If no such t exists, no solution exists. Otherwise output p.

As discussed in Section 3.1, each check at line-2 takes $O(\| S \| + l|S|)$ time and each computation of line-3 takes $O(l^2)$ time, that is, time needed for finding $\mathbf{B}(t)$. Each of these is performed at most once for each candidate in C, thus giving us the runtime as stated. □

It is worth noting that in this case, a solution for MDP can be obtained in time that is not directly dependent on the maximum length of the samples.

Corollary 1. *Given Σ where $|\Sigma| \geq 4$. Let $\boldsymbol{\mathcal{P}} = \{\{p\} \mid p \in \boldsymbol{Psub}\}$ and $S \subseteq \Sigma^*$. There exists an algorithm that solves the* MDP *problem in time $O(\| S \| \cdot (\lfloor S \rfloor^3 + \lfloor S \rfloor^2 |S| + \lfloor S \rfloor \| S \|))$.*

Proof. To find a solution in this case it suffices that we compute MDP for $\boldsymbol{\mathcal{P}} = \{\{p\} \mid p \in \boldsymbol{Psub}_l\}$ for each $l \leq \lfloor S \rfloor$. □

We next look at the case where the unions can involve at most k languages, for some $k \in N^+$, $k \geq 2$.

4.2 The case of $\boldsymbol{\mathcal{P}} = \{P \subseteq \boldsymbol{Psub}_l \mid |P| \leq k\}$

Theorem 5. Given l, $k \in N^+$, $k \geq 2$ and Σ where $|\Sigma| \geq 2$. Let $\boldsymbol{\mathcal{P}} = \{P \subseteq \boldsymbol{Psub}_l \mid |P| \leq k\}$ and $S \subseteq \Sigma^*$. There exists an algorithm that solves the MDP problem in time $O(\min(\{\| S \|, |\Sigma|^l\})^k \cdot (\lceil S \rceil \cdot lk \cdot \min(\{|\Sigma|, lk\}) + lk|S| + k \| S \|)$.

Proof. The proof is the same as that for Theorem 4 except that the number of candidates in this case is $\min(\{\| S \|, |\Sigma|^l\})^k$, and the check at line-3 is not valid. Hence in this case, we use the string counting algorithm as in Theorem 1 for the comparison at line-3. Each computation is done in time $\lceil S \rceil \cdot lk \cdot \min(\{|\Sigma|, lk\})$, giving us the runtime as stated. □

We next consider the computational complexity of MDP with respect to unbounded unions of the languages in $\mathcal{L}(\boldsymbol{Psub}_l)$. Regrettably, in this case, the MDP problem turns out to be NP-complete.

4.3 The case of $\boldsymbol{\mathcal{P}} = \boldsymbol{Psub}_l^*$

We first consider the case where l is very small. In this case, the problem can be solved in time polynomial in the size of the input as follows. Let $T = \{s \mid |s| = l$ and s is a substring of some $w \in S\}$. Hence $|T| \leq |\Sigma|^l$. We then check if $S \subseteq L(P)$ for each $P \subseteq P(T)$, compute its density, and find such P with the highest density. Each computation for T is done in $O(\lceil S \rceil l |T| \min(\{|\Sigma|, l|T|\}))$ by Theorem 1, and the number of candidates T is bounded by $2^{|\Sigma|^l}$. In total, all the computation can be done in $O(2^{|\Sigma|^l} \lceil S \rceil\ l^2\ |\Sigma|^{l+1})$ time.

Theorem 6. The problem of MDP for $\boldsymbol{Psub}_l^*$ can be solved in polynomial time, if l is $O(poly(\log\log(\max\{|S|, \lceil S \rceil\}))$. □

Unfortunately, this problem is NP-hard in general. More precisely, we can show the NP-completeness of the decision version of MDP for $\boldsymbol{Psub}_l^*$; for input parameter D, does there exist a set P of patterns in $\boldsymbol{\mathcal{P}}$ such that $S \subseteq L(P)$ and $d(P, S) \geq D$?

Theorem 7. MDP for $\boldsymbol{Psub}_l^*$ (decision version) is NP-complete.

To prove this, we infer on the number of strings of length m in $|L(P)|$ where $P \in \boldsymbol{Psub}_l^*$. We obtain the following bounds on this number for any P consisting solely of non-overlapping strings from the inclusion-exclusion principle.

Lemma 3. Let T and T' be sets of non-overlapping strings of length l. If $m \leq l^2$, $|T'| = |T| + 1 \leq l$, $|\Sigma| \geq 3$ and $l \geq 14$, then $|L(P(T')) \cap \Sigma^{=m}| > (m - l + 1)|\Sigma|^{m-l} > |L(P(T)) \cap \Sigma^{=m}|$ holds. □

The proof of Theorem 7. Obviously, the decision version of MDP for $\boldsymbol{Psub}_l^*$ belongs to NP by Theorem 1.

We then show the completeness part. We reduce a well-known NP-complete problem, Minimum 3-Set Cover (for short, M3SC) [8] to our MDP.

Problem Minimum 3-Set Cover	
Input:	A set $U = \{1, \ldots, n\}$, its subsets $V_1, \ldots, V_k \subseteq U$, where $\lvert V_i\rvert = 3$ for $i = 1, \ldots, k$, and an integer $K > 0$.
Question:	Is there a family $\mathcal{V} \subseteq \{V_1, \ldots, V_m\}$ of subsets, such that $\bigcup_{V_i \in \mathcal{V}} V_i = U$ and $\lvert\mathcal{V}\rvert \leq K$?

For the M3SC instance, we construct a set of strings $S \subseteq \Sigma^{=m}$, where $\Sigma = \{a, b, c\}$, $m = 3k(r + 2)$ with $r = \max\{n, k\}$, and let $l = 2(r + 2)$. In the construction, we compose $3k$ component (sub)strings with length $l = r + 2$ into strings with length $m = 3k(r + 2)$. For this purpose, we first define component strings. For each subset V_i, we define string $v^{(i)} \in \Sigma^{=r+2}$ as follows:

$$v^{(i)} = \overbrace{c\,\underbrace{bb\cdots b}_{i}\,a\,\underbrace{bb\cdots b}_{i}\,ab\cdots ba\,\underbrace{bb\cdots b}_{p}}^{r+2},$$

where $p \equiv r + 1 \pmod{i + 1}$. For example, $v^{(1)} = cbaba\cdots$, $v^{(2)} = cbbabba\cdots$. Also, for each element $i \in U$ we define three strings $u^{(i,\alpha,\beta)}$:

$$u^{(i,\alpha,\beta)} = \overbrace{\alpha\,\underbrace{a\cdots a}_{p}\,b\cdots ab\,\underbrace{aa\cdots a}_{i}\,b\,\underbrace{aa\cdots a}_{i}\,b\beta}^{r+2},$$

where α, $\beta \in \{a, b, c\}$ and $p \equiv r \pmod{i + 1}$. For example, $u^{(3,a,c)} = aa\cdots abaaabaaac$. By using these component strings in $R = \{v^{(i)},\ \text{for } i = 1, \ldots, k\} \cup \{u^{(i,\alpha,\beta)},\ \text{for } i = 1, \ldots, n, \text{ and } \alpha, \beta \in \{a, b, c\}\}$, we now construct a set S of strings corresponding to the base set U of a given M3SC problem instance. Each string $w \in S$ consists of $3k$ substrings of length $(r + 2)$, each of which is a string in R. Let us denote the j-th substring of string w by w_j. For an element $i \in U$, the corresponding string $w^{(i)}$ is defined as follows:

$$w_h^{(i)} = \begin{cases} u^{(i,\alpha,\beta)} & \text{if } h \equiv 0 \pmod 3 \\ v^{(j)} & \text{if } h \not\equiv 0 \pmod 3 \text{ and } i \in V_j, \text{ where } j = \lceil h/3 \rceil \\ u^{(i,c,c)} & \text{otherwise.} \end{cases}$$

where

$$\alpha = \begin{cases} a & \text{if } i \in V_j \text{ and } |\{1,2,\ldots,i-1\} \cap V_j| = 0, \\ b & \text{if } i \in V_j \text{ and } |\{1,2,\ldots,i-1\} \cap V_j| = 1, \\ c & \text{otherwise,} \end{cases}$$

$$\beta = \begin{cases} a & \text{if } i \in V_{j+1} \text{ and } |\{1,2,\ldots,i-1\} \cap V_{j+1}| = 0, \\ b & \text{if } i \in V_{j+1} \text{ and } |\{1,2,\ldots,i-1\} \cap V_{j+1}| = 1, \\ c & \text{otherwise.} \end{cases}$$

For example, if in the original M3SC instance the sets to which element 4 belongs are only $V_1 = \{4,5,6\}$ and $V_3 = \{1,4,6\}$, the corresponding string is

$$w^{(4)} = \underbrace{v^{(1)}v^{(1)}u^{(4,a,c)}}_{4 \in V_1}\underbrace{u^{(4,c,c)}u^{(4,c,c)}u^{(4,c,b)}}_{4 \notin V_2}\underbrace{v^{(3)}v^{(3)}u^{(4,b,c)}}_{4 \in V_3}u^{(4,c,c)} \cdots u^{(4,c,c)} \cdots.$$

Note that each segment $u^{(i,\alpha,\beta)}$ is unique to $w^{(i)}$ and only $v^{(j)}$s may appear in more than one string in S. Now we claim that for this string set $S = \{w^{(i)}, i \in U\}$, the maximum density of $P \in \boldsymbol{Psub}_l^*$ is not smaller than $n/(K \cdot \sum_{j=l}^{m}((j-l+1)3^{j-l}))$, if and only if the M3SC's answer is yes.

We prove the only-if part. Let $P(T') \in \boldsymbol{Psub}_l^*$ be a cover for S with the highest density. By the above argument, every $t \in T'$ must be a substring of some $s \in S$. Here we have the following three types of substrings t with length $l = 2(r+2)$:

1) $t = v^{(i)}v^{(i)}$, where $i \in \{1,2,\ldots,m\}$,
2) t contains $u^{(j,\alpha,\beta)}$ as its substring where $j \in \{1,\ldots,n\}$,
3) $t = \overbrace{a \cdots abaa \cdots \beta}^{q} v^{(i)} \overbrace{cbb \cdots babb \cdots}^{r+2-q}$, or $\overbrace{b \cdots babb \cdots}^{q} v^{(i)} \overbrace{\alpha aa \cdots abaa \cdots}^{r+2-q}$, where $\alpha, \beta \in \{a,b,c\}$ and $i \in \{1,2,\ldots,m\}$.

Type 1) and 2) of substring pattern covers three strings and one string in S, respectively. As for Type 3), t covers just one string in S, since all the three strings which contain $v^{(i)}$ have different α or β. From these, it follows that when we maximize the density of patterns to cover S, only Type 1) strings are in T'. For solution P with only Type 1) strings, its density is $d(P,S) = \frac{|L(P) \cap S|}{|L(P) \cap \Sigma^{\leq m}|} = \frac{|S|}{\sum_{i=k}^{m} |L(P) \cap \Sigma^{=i}|}$, where $m = 3k(r+2)$ and $l = 2(r+2)$. By the construction of $v^{(i)}$, P with only Type 1) strings are non-overlapping, $m = 3k(r+2) \leq l^2$ and $|P| \leq k \leq l$. By these properties and Lemma 3, solutions P with $|P| = K$ and

P' with $|P'| = K + 1$ satisfy that

$$d(P,S) = \frac{|S|}{\sum_{i=l}^{m} |L(P) \cap \Sigma^{=i}|} \geq \frac{n}{\sum_{j=l}^{m}(K(j-l+1)\cdot 3^{j-l})} \tag{1}$$

$$\geq \frac{|S|}{\sum_{i=l}^{m} |L(P') \cap \Sigma^{=i}|} = d(P', S). \tag{2}$$

Inequality (1) implies that, if the optimal solution is not smaller than $n/(K \cdot (\sum_{j=l}^{m}((j-l+1)\cdot 3^{j-l}))$, $K \geq |P|$ holds. Recall that each member of P is Type 1) and forms $x_1 v^{(i)} v^{(i)} x_2$, which corresponds to set V_i. Thus, by solving MDP for $\boldsymbol{Psub}_l^*$, we can obtain the solution of M3SC, which proves the only-if part. The opposite direction can also be shown by a similar argument and (2). □

5 Conclusions and Future Work

In this paper, we considered the problem of finding patterns that covers a given set of strings. We have argued in this paper our dissatisfactions with current strategies to avoid over-generalizations in the solution pattern languages. We proposed a natural measure, "density", to evaluate the degree of over-generalization of a given solution, and considered the problem of finding among all solutions for one which maximizes such density.

We showed that the resultant problem is closely related to the counting problem and the membership problem. As such, we listed methods to perform these efficiently for a few cases where such computations are possible. We showed that for substring patterns of the same length, to compare between the number of strings up to a certain length generated by any two patterns does not require an exhautive counting of the strings. It remains to be seen if similar methods exist for other subclasses of the pattern languages.

Finally, we studied the runtime complexities for solving the MDP problem on three related classes of languages derived from unions of the substring pattern languages. One of these turned out to be very efficient, the other less so, but still within runtime polynomial to the input size. Regrettably, for the third class of languages we considered, the set of unbounded unions of fixed-length substring pattern languages, the problem of finding a solution which maximizes the density can be related to the Minimum Set Cover Problem, and is shown to be NP-complete.

Acknowledgement.

The authors would like to thank Prof. Takeshi Shinohara for discussions.

References

1. D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.
2. H. Arimura, R. Fujino, T. Shinohara, and S. Arikawa. Protein motif discovery from positive examples by Minimal Multiple Generalization over regular patterns. In *Proceedings of the Genome Informatics Workshop*, pages 39–48, 1994.

3. H. Arimura, T. Shinohara, and S. Otsuki. Finding minimal generalizations for unions of pattern languages and its application to inductive inference from positive data. In *Proc. Annual Symp. on Theoretical Aspects of Computer Sci.*, 1994.
4. A. Brāzma, I. Jonassen, I. Eidhammer, and D. Gilbert. Approaches to the automatic discovery of patterns in biosequences. *J. Comp. Biol.*, 5(2):277–304, 1998.
5. A. Brāzma, E. Ukkonen, and J. Vilo. Discovering unbounded unions of regular pattern languages from positive examples. In *Proceedings of the 7-th International Symposium on Algorithms and Computation (ISAAC'96)*, 1996.
6. B. Brejova, T. Vinar, and M. Li. *Pattern Discovery: Methods and Software*, chapter 29, pages 491–522. Humana Press, 2003.
7. Chee-Yong Chan, Minos Garofalakis, and Rajeev Rastogi. RE-tree: an efficient index structure for regular expressions. *The VLDB Journal*, 12(2):102–119, 2003.
8. M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., New York, 1979.
9. Sampath Kannan, Z. Sweedyk, and Steve Mahaney. Counting and random generation of strings in regular languages. In *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 551–557. Society for Industrial and Applied Mathematics, 1995.
10. M. Sato, Y. Mukouchi, and D. Zheng. Characteristic sets for unions of regular pattern languages and compactness. In *Algorithmic Learning Theory: Ninth International Conference (ALT' 98)*, volume 1501 of *Lecture Notes in Computer Science*, pages 220–233. Springer-Verlag, 1998.
11. T. Shinohara. Polynomial time inference of extended regular pattern languages. In *RIMS Symposia on Software Science and Engineering, Kyoto, Japan*, volume 147 of *Lecture Notes in Computer Science*, pages 115–127. Springer-Verlag, 1982.
12. J. Uemura and M. Sato. Compactness and learning of classes of unions of erasing regular pattern languages. In *Algorithmic Learning Theory: Thirteenth International Conference (ALT' 02)*, volume 2533, pages 293–307. Springer-Verlag, 2002.

Appendix

This appendix provides the proofs of several results in the paper which have been omitted due to space reasons. They may be read to the discretion of the program committee.

A Proof of Theorem 2: The correctness and runtime analysis of *CountStrings*

To see the correctness of *CountStrings*, first consider any string s of length m in $L(``x_1tx_2")$ for some string t shorter than m. Now s must be of one of the following forms: (F.0) "$t?^{m-|t|}$", or (F.1) "$?t?^{m-|t|-1}$", or …, (F.$m-|t|$) "$?^{m-|t|}t$". Now for each n, $0 \leq n \leq m-|t|$, if we know the total number of strings, $g(n)$ say, of the form (F.n) but not of the forms (F.0), (F.1), …, (F.$n-1$), the sum of $g(0)$, $g(1)$, …, $g(m-|t|)$ would then give us the number of strings of length m in $L(``x_1tx_2")$. The same method is used in *CountStrings*, where for $t \in T$ and $r \in N$, $\lfloor T \rfloor \leq r \leq m$, we compute the total number, $f(t,r)$ say, of strings of length r of the form "w_1t" (that is, ending in t), but not of any of the form "$w_2t'w_3$" for some $t' \in T$, $w_1, w_2 \in \Sigma^*$, $w_3 \in \Sigma^+$. (Here, the case where $|w_3| = 0$ is not considered since no string in T is a substring of another.) It is easy to verify that the total number of strings of length m in $L(P(T))$, $\Phi(T, m) = \sum_{r=\lfloor T \rfloor}^{m} \sum_{t \in T \wedge |t| \geq r} |\Sigma|^{m-r} f(t, r) = |\Sigma| \cdot \Phi(T, m-1) + \sum_{t \in T \wedge |t| \geq m} f(t, m)$.

To calculate $f(t, r)$, we first see that there are $|\Sigma|^{r-|t|}$ strings of length r of the form "$x_1 t$", and $f(t, r)$ is given by this number deducted by the number of strings of length r that are simultaneously of the forms "$w_1 t$" and "$w_2 t' w_3$" for some $t' \in T$ and $w_1, w_2 \in \Sigma^*$, $w_3 \in \Sigma^+$. The strings of length r of the form "$wt'w't$" where w, $w' \in \Sigma^*$ trivially matches these two forms, and it is easy to see that their number is given by $\Phi(T, r - |t|)$.

The remaining strings simultaneously having the forms "$w_1 t$" and "$w_2 t' w_3$" must have: (C1) $\max(0, |t| - |t'|) < |w_3| < |t|$, (C2) some suffix of t' matching some prefix of t, and (C3) not of the form "$wt''w't$" for some $t'' \in T$, w, $w' \in \Sigma^*$. As an example, string (A) below fulfills (C1), (C2) but fails (C3); while string (B) fulfills all three conditions for $T = \{$"bab"$\}$.

$$
\begin{array}{ll}
 & b\ a\ b\ ?\ ?\ \overbrace{b\ a\ b}^{t} \\
 & ?\ b\ a\ b\ ?\ b\ a\ b \\
 & ?\ ?\ b\ a\ b\ b\ a\ b \\
\text{(A)} & a\ b\ a\ \underbrace{b\ a\ b}_{t}\ a\ b \\
\text{(B)} & a\ b\ b\ \overbrace{b\ a\ b}^{t}\ a\ b
\end{array}
$$

In general, for any t, $t' \in T$, $r \in N$ and $w_3 \in \Sigma^+$ where $\max(0, |t| - |t'|) < |w_3| < |t|$, the number of strings of length r ending in "$t'w_3$" which fulfill (C3) is the same as the number of strings of length $r - |w_3|$ ending in t' and with the initial $r - |t|$ long segment not in $L(P(T))$. For this reason we let $S(T, t', r', r'')$ denote the number of strings of length r' that ends in t' and does not begin with a string in $\Sigma^{=r''} \cap L(P(T))$. Hence, it is sufficient to compute $f(t, r)$ with only $\Phi(T, r - |t|)$ and the values $S(T, t', r', r - |t|)$ for all $t' \in T$ and for all r' where $r - |t| < r' < r - \max(0, |t| - |t'|)$.

Noting that $f(t, r)$ is simply $S(T, t, r, r-1)$, it is easy to verify that using the same strategy for computing $f(t, r)$, it is sufficient for us to compute $S(T, t, r, r')$ for any $t \in T$, r, $r' \in N$ with only $\Phi(T, r - |t|)$ and $S(T, t', r'', r - |t|)$ for all $t' \in T$ and all r'' where $r - |t| < r'' < r$. The listing for $S(T, t, r, r')$ in the paper does exactly this. Note that lines-8–10 remove the prefix strings where their suffixes are also included, since they need not be considered because a string which ends in one of these prefix strings also ends in their suffixes.

We now look at the runtime complexity for *CountStrings*. $S(T, t, r, r')$ is called $m \cdot \parallel T \parallel$ times, while $\Phi(T, r)$ is called m times. The computation for $\Phi(T, r)$ takes time $O(|T|)$ for each r, costing $O(m \cdot |T|)$ in total. In $S(T, t, r, r')$, the computation at line-12 takes time $O(\parallel T \parallel)$, i.e. the maximum value of $|B_{t,r,r'}|$; while the computation for $B_{t,r,r'}$ takes time $O(\sum_{t \in T} |t|^2)$. However, we note that for all $r \geq 2\lceil T \rceil - 1$, $B_{t,r,r'} = B_{t,r+1,r'+1}$, and hence some trivial modification to $S(T, t, r, r')$ would make it run in time $O(\parallel T \parallel)$ for $r \geq 2\lceil T \rceil - 1$. Hence the algorithm runs in time $O(m \parallel T \parallel^2)$ for $m \gg \lceil T \rceil$. □

B Proof of Lemma 1

We first note that for all $i > 0$, $S(\{t\}, t, x+i, x) \leq |\Sigma|^{x+i-|t|}$ and $\Phi(\{t\}, x+i) \leq |\Sigma|^{x+i-|t|}$. Now from the recurrence relation,

$$\begin{aligned} S(\{t\}, t, x+d, x) &\geq |\Sigma|^{x+d-|t|} - \Phi(\{t\}, x+d-|t|) \\ &\qquad\qquad - \textstyle\sum_{0\leq i<|t|-d} S(\{t\}, t, x-i, x+d-|t|) \\ &\geq |\Sigma|^{x+d-|t|} - |\Sigma|^{x+d-2|t|} - \textstyle\sum_{0\leq i<|t|-d} |\Sigma|^{x-i-|t|} \end{aligned}$$

Since $d < |t|$, $|\Sigma|^{x-|t|}$ is the single largest negative term, while all other $|\Sigma|^n$ terms for $n < x-|t|$ appear at most once. That $|\Sigma| \geq 3$ then gives us $S(\{t\}, t, x+d, x) \geq \sum_{1\leq i\leq d-1} |\Sigma|^{x-|t|+i} \geq \sum_{1\leq i\leq d-1} S(\{t'\}, t', x+i, x)$.

C Proof of Lemma 2

From the recurrence relation,

$$\begin{aligned} &S(t'', t'', x+1, x) - |S(\{t\}, t, x+d, x) - S(\{t'\}, t', x+d, x)| \\ &\quad \geq |\Sigma|^{x+1-|t|} - \Phi(\{t''\}, x+1-|t|) - \textstyle\sum_{0\leq i<|t|-1} S(\{t''\}, t'', x-i, x+1-|t|) \\ &\qquad\qquad - \max(\{\Phi(\{t'\}, x+d-|t|), \Phi(\{t\}, x+d-|t|)\}) \\ &\qquad\qquad - \max(\{\textstyle\sum_{0\leq i<|t|-d} S(\{t\}, t, x-i, x+d-|t|), \\ &\qquad\qquad\qquad \textstyle\sum_{0\leq i<|t|-d} S(\{t'\}, t', x-i, x+d-|t|)\}) \end{aligned}$$

The analysis is similar to that for the proof of Lemma 1. Each of the four negative terms may introduce at most one $|\Sigma|^n$ for any $n \leq x-|t|$, none of $|\Sigma|^n$ for $n > x-|t|$, and only the last three terms may include $|\Sigma|^{x-|t|}$.

D Proof of Theorem 3

Let Σ, t, t' be as in Claim. From the recurrence relation in *CountStrings* we have the following equation. For convenience we denote the first term of the left hand side of the equation (A), the second term (B), and the two remaining terms (C).

$$\begin{aligned} &\Phi(\{t\}, r) - \Phi(\{t'\}, r) \\ &\quad = |\Sigma| \cdot (\Phi(\{t\}, r-1) - \Phi(\{t'\}, r-1)) && \text{(A)} \\ &\qquad -(\Phi(\{t\}, r-|t|) - \Phi(\{t'\}, r-|t|)) && \text{(B)} \\ &\qquad\quad - (\textstyle\sum_{x\in\mathbf{B}(t,r)} S(\{t\}, t, r-|t|+x, r-1-|t|) \\ &\qquad\qquad - \textstyle\sum_{x'\in\mathbf{B}(t',r)} S(\{t'\}, t', r-|t|+x', r-1-|t|)) && \text{(C)} \end{aligned}$$

From the recurrence relation we further know that for any t and for any $x \in \mathbf{B}(t)$, $x \in \mathbf{B}(t, r)$ if and only if $r \geq 2|t| - x$. Intuitively, if we consider each $x \in \mathbf{B}(t)$ as being added to $\mathbf{B}(t, r)$ as r increases, then the value x is added when $r = 2|t| - x$.

Hence if $\mathbf{B}(t) = \mathbf{B}(t')$, then for all $r \in N$, $\Phi(\{t\}, r) = \Phi(\{t'\}, r)$. We now consider the case where $\mathbf{B}(t) \neq \mathbf{B}(t')$. As stated in the claim, let $\max(\mathbf{B}(t) \setminus$

$\mathbf{B}(t')) < \max(\mathbf{B}(t') \setminus \mathbf{B}(t))$. For $r < 2|t| - \max(\mathbf{B}(t') \setminus \mathbf{B}(t))$, since $\mathbf{B}(t, r) = \mathbf{B}(t', r)$ for $\Phi(\{t\}, r) = \Phi(\{t'\}, r)$. This concludes Part (1) of the Theorem.

For all $r \geq 2|t| - \max(\mathbf{B}(t') \setminus \mathbf{B}(t))$, since $\max(\mathbf{B}(t', r)) > \max(\mathbf{B}(t, r))$, by Lemma 1 and 2, the following holds.

$$\sum_{x \in \mathbf{B}(t,r)} S(\{t\}, t, ., .) < \sum_{x' \in \mathbf{B}(t',r)} S(\{t'\}, t', ., .).$$

That is, part (C) is non-zero and negative, and since negated, it takes non-zero positive values.

Since for all $r \leq 2|t| - \max(\mathbf{B}(t') \setminus \mathbf{B}(t))$, $\Phi(\{t\}, r) = \Phi(\{t'\}, r)$, (A) and (B) are both zero for these values of r. (C) part then gives us that

$$\Phi(\{t\}, 2|t| - \max(\mathbf{B}(t') \setminus \mathbf{B}(t))) > \Phi(\{t'\}, 2|t| - \max(\mathbf{B}(t') \setminus \mathbf{B}(t))) \tag{3}$$

We claim that for all r,

$$\Phi(\{t\}, r) - \Phi(\{t'\}, r) \geq \Phi(\{t\}, r-1) - \Phi(\{t'\}, r-1). \tag{4}$$

We have seen that this is true for all r where $|t| \leq r \leq 2|t| - \max(\mathbf{B}(t') \setminus \mathbf{B}(t))$. We now consider $r = 2|t| - \max(\mathbf{B}(t') \setminus \mathbf{B}(t)) + 1$. Considering only (A) and (B) part of $\Phi(\{t\}, r) - \Phi(\{t'\}, r)$, since $|\Sigma| \geq 2$,

$$\begin{aligned} &|\Sigma| \cdot (\Phi(\{t\}, r-1) - \Phi(\{t'\}, r-1)) - (\Phi(\{t\}, r-|t|) - \Phi(\{t'\}, r-|t|)) \\ &\geq (\Phi(\{t\}, r-1) - \Phi(\{t'\}, r-1)) + && \text{(AB1)} \\ &\quad (\Phi(\{t\}, r-1) - \Phi(\{t'\}, r-1)) - (\Phi(\{t\}, r-|t|) - \Phi(\{t'\}, r-|t|))) && \text{(AB2)} \end{aligned}$$

Since (4) holds for all $r' \leq \max(\mathbf{B}(t') \setminus \mathbf{B}(t)) + 1$, (AB2) is positive. Hence $\Phi(\{t\}, r) - \Phi(\{t'\}, r)$ is of at least (AB1)$= \Phi(\{t\}, r-1) - \Phi(\{t'\}, r-1)$ plus part (C), which is positive. Hence (4) holds for $r = 2|t| - \max(\mathbf{B}(t') \setminus \mathbf{B}(t)) + 1$. The same argument applies to all subsequent values of r. (3) and (4) then concludes Part (2) of the Theorem.

E $|L(P(T))| \cap \Sigma^{=m}$ for a set T of non-overlapping strings of length l and the proof of Lemma 3

We give here bounds of the number of strings with length m that are covered by a set of patterns $P(T) \in \boldsymbol{Psub}_l^*$ where T is non-overlapping.

Here, given a set T of non-overlapping strings with length l, we consider the cardinality of $L(P(T)) \cap \Sigma^{=m}$, and its upper bound and lower bound. By the inclusion and exclusion relationship, we have

$$|L(P(T)) \cap \Sigma^{=m}| = \sum_{i=1}^{|T|} (-1)^{i-1} \left(\sum_{\substack{P' \subseteq P(T), \\ |P'|=i}} \left| \left(\bigcap_{\text{“}x_1 t x_2\text{”} \in P'} L(\text{“}x_1 t x_2\text{”}) \right) \cap \Sigma^{=m} \right| \right).$$

Namely, we have the following two inequalities:

$$|L(P(T)) \cap \Sigma^{=m}| \le \sum_{t \in T} |L(\text{“}x_1 t x_2\text{”}) \cap \Sigma^{=m}|\,, \tag{5}$$

$$|L(P(T)) \cap \Sigma^{=m}| \ge \sum_{t \in T} |L(\text{“}x_1 t x_2\text{”}) \cap \Sigma^{=m}| - \sum_{t_1, t_2 \in T} |(\bigcap_{i=1,2} L(\text{“}x_1 t_i x_2\text{”})) \cap \Sigma^{=m}|. \tag{6}$$

First, we consider (5), the upper bound on $|L(P(T)) \cap \Sigma^{=m}|$. Each $t \in T$, we have the similar equation,

$$|L(\text{“}x_1 t x_2\text{”}) \cap \Sigma^{=m}| = \sum_{i=1}^{m-l+1} (-1)^{i-1} \left(\sum_{\substack{P' \subseteq P_t,\\ |P'|=i}} \left| (\bigcap_{\text{“}?^j t x_2\text{”} \in P'} L(\text{“}?^j t x_2\text{”})) \cap \Sigma^{=m} \right| \right), \tag{7}$$

where P_t denotes $\{\text{“}?^j t x_2\text{”} \mid j = 0, \ldots, m-l\}$, so the upper bound on $|L(\text{“}x_1 t x_2\text{”}) \cap \Sigma^{=m}|$ is $\sum_{j=0}^{m-l} \left| L(\text{“}?^j t x_2\text{”}) \cap \Sigma^{=m} \right| = (m-l+1)|\Sigma|^{m-l}$. Therefore, the upper bound (5) is bounded by $|T|(m-l+1)|\Sigma|^{m-l}$.

Next, we consider (6), the lower bound on $|L(P(T)) \cap \Sigma^{=m}|$. By equation (7) and T's non-overlappingness, we have the lower bound on $|L(\text{“}x_1 t x_2\text{”}) \cap \Sigma^{=m}|$,

$$\begin{aligned} |L(\text{“}x_1 t x_2\text{”}) \cap \Sigma^{=m}| &\ge \sum_{j=0}^{m-l} \left| L(\text{“}?^j t x_2\text{”}) \cap \Sigma^{=m} \right| - \sum_{j_1, j_2} \left| (\bigcap_{j=j_1,j_2} L(\text{“}?^j t x_2\text{”})) \cap \Sigma^{=m} \right| \\ &= (m-l+1)|\Sigma|^{m-l} - \binom{m-2l+2}{2} |\Sigma|^{m-2l}. \end{aligned} \tag{8}$$

Since T is non-overlapping, so for $t_1, t_2 \in T$, $|L(\text{“}x_1 t_1 x_2\text{”}) \cap L(\text{“}x_1 t_2 x_2\text{”}) \cap \Sigma^{=m}|$ is also $\binom{m-2l+2}{2}|\Sigma|^{m-2l}$. From (6) and (8), we have the lower bound on $|L(\text{“}x_1 t x_2\text{”}) \cap \Sigma^{=m}|$,

$$|T| \left((m-l+1)|\Sigma|^{m-l} - \binom{m-2l+2}{2} |\Sigma|^{m-2l} \right) - \binom{|T|}{2} \binom{m-2l+2}{2} |\Sigma|^{m-2l}$$

By these upper and lower bounds, for non-overlapping T' and T with $|T'| = h+1$ and $|T| = h$, we have

$$\begin{aligned} & |L(P(T')) \cap \Sigma^{=m}| - |L(P(T)) \cap \Sigma^{=m}| \\ \ge\ & (m-l+1)|\Sigma|^{m-l} - \binom{h+2}{2} \binom{m-2l+2}{2} |\Sigma|^{m-2l} \\ =\ & |\Sigma|^{m-2l} \left((m-l+1)|\Sigma|^{l} - \binom{h+2}{2} \binom{m-2l+2}{2} \right) \\ \stackrel{\text{def}}{=}\ & |\Sigma|^{m-2l} \cdot f(m, h, l). \end{aligned}$$

That is, if m and h is not too large with respect to l and l is not so small, $|L(P(T')) \cap \Sigma^{=m}| - |L(P(T)) \cap \Sigma^{=m}| = |\Sigma|^{m-2l} \cdot f(m,h,l) > 0$ holds, since the growth of the exponential function $(m-l+1)|\Sigma|^l$ is much larger than the one of the polynomial function $\binom{h+2}{2}\binom{m-2l+2}{2}$.

Proposition 1. If m and l are bounded by some $poly(l)$s, there exists a constant K such that $|L(P(T')) \cap \Sigma^{=m}| > (m-l+1)|\Sigma|^{m-l} > |L(P(T)) \cap \Sigma^{=m}|$ holds for any $l \geq K$. □

For example, if $m \leq l^2$, $h \leq m$ and $|\Sigma| = 3$, we have $\partial f(m,h,l)/\partial m \geq 3^l - (l+2)(l+1)(2l^2-4l+3)/4 \geq 0$ for not small l, such as $l \geq 14$. That is, $|L(P(T')) \cap \Sigma^{=m}| - |L(P(T)) \cap \Sigma^{=m}| > 0$ holds. This calculation leads to a more specific proposition.

Proposition 2. (**Lemma 3**) If $m \leq l^2$, $h \leq l$, $|\Sigma| \geq 3$ and $l \geq 14$, $|L(P(T')) \cap \Sigma^{=m}| > (m-l+1)|\Sigma|^{m-l} > |L(P(T)) \cap \Sigma^{=m}|$ holds. □