A generic algorithm for sequence mining

J. David[a], L. Nourine[b]

[a] *LIPN UMR 7030, Université Paris 13 - CNRS, 99, avenue Jean-Baptiste Clément, 93430 Villetaneuse, France.*
Julien.David@lipn.univ-paris13.fr
[b] *Campus des Cézeaux, LIMOS Université Blaise Pascal, 63173 Aubière CEDEX, France*
UMR 6158 [1]

1. Introduction

The enumeration of all patterns that satisfies a given predicate in a given context is at the core of the data mining field and has numerous applications in various domain such as biology, software engineering or social networks. The fact that the number of solutions of an enumeration problem often being exponential in the size of the input, the complexity of enumeration algorithms is measured according to the size of both input and output. Also, it is essential that the list does not contain duplicates. The mining of frequent subsequences has already been studied and efficient algorithms have been found [1, 2, 6].

In this paper, we develop a generic algorithm for sequence mining that does not appear in the literature, to the best of our knowledge. We consider that a pattern in a sequence is a multiset of words that partitions the sequence. The notion of partition involves the shuffle product operator [10]. Using additional predicates, this notion of pattern can be used for frequent subwords and subsequences mining. Without additional predicates, we obtain a new kind of pattern that might find some use in the future. Given a word w defined over a finite alphabet Σ, we study the generation problem of all the possible ways to partition w into a multiset of subwords. Since a subword appears several time, these partitions are called multisets of words on Σ.

We focus on developing an algorithm to enumerate all frequent multisets that partition a given word. The naive strategy of enumerating all partitions of an input word and check whether the obtained multiset satisfies all the predicates is not a good one: the number of partitions might be exponentially greater than the number of solutions. Also, this strategy does not avoid redundancy, unless the algorithm stores every solution in order to compare them and therefore has an exponential space complexity. Our idea is the following: we define the transition graph T_w whose vertices are multisets and arcs are transitions between

[1] This word was completed with the support of the ANR project DAG number 09-EMER-003-01

frequent multisets. We show that T_w is a directed acyclic graph and moreover we describe its covering tree. We also propose an algorithm that traverse this covering tree and enumerates all solutions without redundancy.

This paper is organized as follows. In Section 2 we give the definitions of some notions that will be used in this paper. In Section 3, we introduce all the different predicates that be will be considered on multisets of words. The central predicate uses the notion of shuffle product to decide if a multiset is a pattern in the input word. The manipulation of the shuffle product is motivated by the NP-completeness results that already exists for this operator, but also the intuition that our definitions and proofs are more elegant using this tool. Using additional predicates, we show that multiset can be used to encode classical pattern such as frequent subwords. In Section 4.3 we describe the transition graph of all the solutions and show it admits a covering tree. Also, we detail how this graph should be manipulated according to different predicates. In Section 5, we describe the algorithm to exhaustively generate all solutions and we give its worst-case complexity.

2. Definitions

An *alphabet* Σ is a finite set of letters. A *word* defined over Σ is a sequence of letters that belongs to Σ. The set of all words over Σ is noted Σ^* and the set of all non-empty words is Σ^+. Let w be a word and $a \in \Sigma$ a letter, $|w|$ denotes the word's length and $\|w\|_a$ is the number of letters a in w. Let w and v be two words over an alphabet Σ. The word v is a *subword* of w if there exists $u \in \Sigma^*$ such that

$$w = u_1 v_1 \dots u_m v_m \text{ with } u = u_1 \dots u_m,\ v = \dots v_m \text{ and } u_i, v_i \in \Sigma^* \text{ for all } i \geq m$$

The word v is a *subsequence* of w if there exists two words $u_1, u_2 \in \Sigma^*$ such that $w = u_1 v u_2$. We recall the notion of *military order* [3] , or *graduated lexicographical order*, on words. Let v and w be two distinct words defined over Σ. We have

$$v <_{mil} w \iff \begin{cases} |v| < |w| \\ \text{or} \\ |v| = |w| \text{ and } v <_{lex} w \end{cases}$$

where $<_{lex}$ is the classical lexicographical order.

A multiset is a set where elements can appear several times. In Singh *et al.* [8], the authors give an overview on multisets and their applications. We recall that a *multiset* M of words is a pair $\langle X, f \rangle$ such that $X \in 2^{\Sigma^+}$ is the underlying set of words of M and $f : \Sigma^+ \mapsto \mathbb{N}$ is a function that associates with each word its multiplicity in M, which means that for all words $w \in \Sigma^+$, we have

$$X = \{w \in \Sigma^+ \mid f(w) > 0\}.$$

A multiset $\langle X, f \rangle$ can also be seen as a set of pairs (v, i) such that $v \in X$ and $i = f(v)$. We note $\mathcal{M}$ the set of all multisets of words over an alphabet Σ.

Definition 1. *Let $M_1 = \langle X, f\rangle$ and $M_2 = \langle Y, g\rangle$ be two multisets. The multiset sum of two multisets M_1 and M_2, noted $M_1 \oplus M_2$, is the multiset $M = \langle Z, h\rangle$ defined as follows:*

- $Z = X \cup Y$,
- *for all $z \in Z$, we have $h(z) = f(z) + g(z)$.*

Definition 2. *Let $M_1 = \langle X, f\rangle$ and $M_2 = \langle Y, g\rangle$ be two multisets. The multiset difference of two multiset M_1 and M_2, noted $M_1 \ominus M_2$, is the multiset $M = \langle X, h\rangle$ defined as follows :*

- *for all $x \in X$, we have $h(z) = \max\{f(z) - g(z), 0\}$.*

3. Constraints on multisets of words

In this section, we present a list of predicates on multisets of words that will be compatible with our generator. We explain how the conjunction of certain predicates allows to obtain a generator for classical sequence patterns, such as frequent subsequences or frequent subwords. We distinguish two kinds of predicates:

- those who involve a test on the input.
- those who can be tested independently from the input word in order to trim the search space.

3.1. The generic input-dependant constraint

In this paper, a pattern in a sequence is a multiset of words that forms a partition of the input word. The notion of partition we introduce here involve the *shuffle product* operator. Partitions of a sequence using multisets of words is a generalization of classical pattern notions, as we will later detail. The *shuffle product* of two sets of words X and Y defined over Σ is defined as follows:

$$\begin{aligned} X \sqcup\!\sqcup Y = \{z \in \Sigma^* \mid \exists n \in \mathbb{N}^*, z = u_1 v_1 \cdots u_n v_n \text{ with} \\ u = u_1 \cdots u_n,\ u \in X, v = v_1 \cdots v_n,\ v \in Y \\ \text{and } u_i, v_i \in \Sigma^*, \text{ for all } 1 \leq i \leq n\}. \end{aligned}$$

One could say that the shuffle product of two sets X and Y is the set of possible ways to sparsely insert a word $u \in X$ in a word $v \in Y$. As stated in the introduction, we use this operator for the purpose of later proofs. To simplify notations, we consider that the shuffle product of two words u and v is equal to the shuffle product of sets $\{u\}$ and $\{v\}$. The shuffle product can be extended in several ways. We have:

$$\sqcup\!\sqcup^{k+1} w = (\sqcup\!\sqcup^{k} w) \sqcup\!\sqcup w \text{ and } \sqcup\!\sqcup^{0} w = \varepsilon$$

and for all sets of m words $X = \{w_1, w_2, \ldots, w_m\}$,

$$\bigsqcup\!\bigsqcup X = w_1 \sqcup\!\sqcup w_2 \sqcup\!\sqcup \cdots \sqcup\!\sqcup w_m$$

Let w and v be two words on an alphabet Σ. The word v is said to be a *subword* of w if and only if there exists a word $u \in \Sigma^*$ such that $w \in v \sqcup\!\sqcup u$.

A set $X = \{w_1, \ldots, w_m\}$ of subwords of a word w is said to be a *partition* of w if and only if $w \in \bigsqcup\!\bigsqcup X$. Clearly partitions of words are different from partitions of multisets.

We now extend the shuffle product to multisets of words. Let $M = \langle X, f \rangle$ be a multiset, $\bigsqcup\!\bigsqcup M$ is the set of words obtained by the following shuffle product:

$$\bigsqcup\!\bigsqcup M = \bigsqcup_{v \in X}\!\!\bigsqcup (\sqcup\!\sqcup^{f(v)}\ v).$$

Definition 3. *A multiset of words M* partitions *a word w if*

$$w \in \bigsqcup\!\bigsqcup M.$$

Example 1. *Let $w = babcbcaab$ be a word over $\{a, b, c\}$. The multiset $\{(a, 3), (bcb, 2)\}$ partitions w, since we have*

$$\sqcup\!\sqcup^3\ a = \{aaa\},\ bbcbcb \in \sqcup\!\sqcup^2\ bcb \text{ and } w \in aaa \sqcup\!\sqcup bbcbcb.$$

Definition 4. *Let P be a set of predicates and $w \in \Sigma^*$ an input word. A multiset $M = \langle X, f \rangle$ is (w, P)-valid if and only if :*

1. *M satisfies all the predicates in P.*
2. *M partitions w.*

We note $\mathcal{M}_{w,P}$ the set of all (w, P)-valid multisets.

Problem: Generic Sequence Mining.

Input :

- an alphabet Σ,
- a word $w \in \Sigma^+$,
- a set of predicates P.

Output : all (w, P)-valid multisets.

To the best of our knowledge, there is currently no algorithm that enumerates such kind of pattern. Though, as we will see, multisets of words are a generalization of common patterns in the sequence mining field.

Parametrization. Here we introduce the first predicate that allows to parametrize the mining problem. Let $\langle X, f\rangle$ be a multiset that partitions a word $w \in \Sigma^*$. Therefore, there exists a partition of the set $\{1, \ldots, |w|\}$, noted $P(M, w)$, such that each part $P_i \in P(M, w)$, with $P_i = \{pos_1, \ldots, pos_{|P_i|}\}$, is associated to the $i-th$ word of M in the military order, noted v_i, such that $v_i = w_{pos_1} w_{pos_2} \ldots w_{pos_{|P_i|}}$.

Example 2. *As we saw in the previous example, the multiset* $\{(a, 3), (bcb, 2)\}$ *partitions* $w = b_1a_2b_3c_4b_5c_6a_7a_8b_9$. *The partition* $\{\{2\}, \{7\}, \{8\}, \{1, 4, 5\}, \{3, 6, 9\}\}$ *indicates the position of each letter of each subword in the multiset of words.*

The distance predicate. Let $\mathfrak{p}_{dist} : \mathcal{M} \times \Sigma^* \times \mathbb{N} \mapsto \{0, 1\}$ be the distance predicate. For a fixed multiset M, a word w and an integer $d \in \mathbb{N}^*$, $P_{dist}(M, w, d)$ is *true* if and only if there exist a partition $P(M, w)$ such that for each part $P \in P(w, M)$ and each $i < |P|$, we have $pos_i + d \geq pos_{i+1}$, with $pos_i, pos_{i+1} \in P$.

In other words, the predicate $\mathfrak{p}_{dist}$ ensures that the distance in the input word between the events of a subwords is bounded by a value d. If d is equal to 1, then $\mathfrak{p}_{dist}$ checks whether w is partitioned by a multiset of subsequences instead of subwords.

3.2. Input-independent constraints

In this section we present a collection of predicates that are can be useful in practical sequence mining problems. We show that, using a conjunction of those predicates, we obtain classical problems such as frequent subword or subsequence mining. The predicates introduced here exclusively deal with the words in the multisets. We consider that words of length 1 in a multiset of words only play a role of regulator, that is to say that they guarantee that the multisets are w-candidates. Thus the predicates we describe only apply on the words of length greater than 1. Though, if required, it is easy to adapt the definitions so the predicates include words of length 1.

The frequency predicate.. The frequency of a pattern is defined as its minimum number of occurrences in a given input. It is one of the most common and most studied predicates [1, 2, 4, 5].

Let Σ be an alphabet and $threshold : \Sigma \mapsto \mathbb{N}$ a function that associates to each letter of the alphabet a threshold that must be satisfied for a multiset of word to be a candidate. Thus, the function *threshold* can be extended to words the following way

$$\forall v \in \Sigma^*,\ threshold(v) = max\{threshold(a) | a \in v\}$$

Knowing that given events are much more frequent than others, to fix a frequency for each letter is a way to reduce the search space.

For a fixed k-letter alphabet, let $\mathfrak{p}_{freq} : \mathcal{M} \times \mathbb{N}^k \mapsto \{0, 1\}$ be the frequency constraint such that for all multiset $\langle X, f\rangle$, we have

$$\mathfrak{p}_{freq}(\langle X, f\rangle, threshold) = \begin{cases} 1, \text{ if for all word } v \in X \setminus \Sigma,\ f(v) \geq threshold(v) \\ 0, \text{ otherwise} \end{cases}$$

The predicate on number of words.. Let $\mathfrak{p}_{nb} : \mathcal{M} \times \{\leq, \geq, =, <, >\} \times \mathbb{N} \mapsto \{0, 1\}$ be the number constraint such that for a multiset $\langle X, f\rangle$, a relation $R \in \{\leq, \geq, =, <, >\}$ and an integer *bound* we have:

$$\mathfrak{p}_{nb}(\langle X, f\rangle, R, bound) = \begin{cases} 1, \text{ if } \left(\sum_{v \in X \setminus \Sigma} f(v)\right) \ R \ bound \\ 0, \text{ otherwise} \end{cases}$$

The length predicate.. It is possible to output only multisets of words whose length have a minimum or a maximum value.

Let $\mathfrak{p}_{length} : \mathcal{M} \times \{\leq, \geq, =, <, >\} \times \mathbb{N} \mapsto \{0, 1\}$ be the length constraint such that for a multiset $\langle X, f\rangle$, a relation $R \in \{\leq, \geq, =, <, >\}$ and an integer *bound*, we have:

$$\mathfrak{p}_{length}(\langle X, f\rangle, R, bound) = \begin{cases} 1, \text{ if for all } v \in X \setminus \Sigma, |v| \ R \ bound \\ 0, \text{ otherwise} \end{cases}$$

Classical sequence mining.. Most mining algorithms on sequence enumerates frequent subsequences or subwords in an input sequence.

The problem of generating frequent subwords in a word $w \in \Sigma^*$ that satisfies a quorum q is equivalent to the generation of (w, P)-valid multisets of words M such that P contains the following predicates:

- $\mathfrak{p}_{nb}(M, =, 1)$,
- $\mathfrak{p}_{freq}(M, threshold)$ with $threshold(a) = q$ for all $a \in \Sigma$.

Example 3. *For the given word $w = babcbcab$ and a quorum $q = 2$, the set of frequent subwords is: $\{a, b, c, ab, ba, bb, bc, cb, bab, bcb\}$. This result can be encoded with multisets. Thus, we obtain:*

$$\begin{aligned} \{\ &\{(\mathbf{a}, 2), (\mathbf{b}, 4), (\mathbf{c}, 2)\},\ \{(b, 2), (c, 2), (\mathbf{ab}, 2)\}, \{(b, 2), (c, 2), (\mathbf{ba}, 2)\}, \\ &\{(a, 2), (c, 2), (\mathbf{bb}, 2)\}, \{(a, 2), (b, 2), (\mathbf{bc}, 2)\},\ \{(a, 2), (b, 2), (\mathbf{cb}, 2)\}, \\ &\{(c, 2), (\mathbf{bab}, 2)\},\ \{(a, 2), (\mathbf{bcb}, 2)\}\ \} \end{aligned}$$

The problem of generating frequent subsequences requires to add the predicate $\mathfrak{p}_{dist}(M, w, 1)$. The generation of frequent subsequence with wildcards can be dealt using the predicate $\mathfrak{p}_{dist}$ with distance greater than 1. The distance is then equal to the maximum number of wildcards between each letter of a subsequence.

Restriction to a regular language.. In [9], the authors study the mining of frequent sequence satisfying a regular expression.

Let $\mathcal{A}$ be the set of finite automata and $A \in \mathcal{A}$ be an automaton recognizing a regular language $\mathcal{L}_A$. One could consider only multisets of words $\langle X, f\rangle$ such that $X \subseteq \mathcal{L}_A$.

Let $\mathfrak{p}_{lang} : \mathcal{M} \times \mathcal{A} \mapsto \{0, 1\}$ be the language predicate such that for a fixed automaton $A \in \mathcal{A}$ and a multiset $\langle X, f\rangle$, we have:

$$\mathfrak{p}_{lang}(\langle X, f\rangle, A) = \begin{cases} 1, \text{ if } X \setminus \Sigma \subseteq \mathcal{L}_A \\ 0, \text{ otherwise.} \end{cases}$$

Automata can be used to describe rules such as: in each word of length greater than 1, an event b has to be preceded by a sequence of event a^*c.

In the following section, we define a transition graph T_w whose vertices are multisets and arcs are transitions between multisets. We show that T_w is a directed acyclic graph and moreover we describe its covering tree. We also propose an algorithm that traverse this covering tree and enumerates all solutions without redundancy.

4. The transition Graph

Definition 5. *Let $M = \langle X, f\rangle$ be a multiset, $w \in \Sigma^+$ be a word and P a set of predicates. We say that M is a* w-candidate *if and only if*

$$\text{for all } a \in \Sigma, \ \sum_{v \in X} (f(v) \times \|v\|_a) = \|w\|_a.$$

Note that w-candidates are exactly the partitions of the word w when words are considered as a bag of letters. Also if a multiset satisfies the Property 2 of Definition 4 for a fixed word w, then this multiset is also a w-candidate.

We note $\mathfrak{M}_w$ the set of all w-candidates.

In this remainder of this paper we define a graph whose vertices are w-candidates and derive an algorithm to search this graph.

We describe an operation on multisets, in order to define a transition relation between w-candidates.

Definition 6. *Let $\mathcal{M} = \langle X, f\rangle$ be a multiset of words on an alphabet Σ, $v \in X$ be a non-empty word, $a \in X \cap \Sigma$ be a letter and $i \in \mathbb{N}^*$. We define the multiset $\mathfrak{extend}(\mathcal{M}, v, a, i)$ obtained by applying the following operations:*

$$\mathfrak{extend}(\mathcal{M}, v, a, i) = \mathcal{M} \oplus \{(va, i)\} \ominus \{(v, i)\} \ominus \{(a, i)\}$$

In other words, to extend a multiset of words consist in suppressing i occurrences of the words a and v and adding i occurrences of the word va. Note that, given a w-candidate $M = \langle X, f\rangle$, if i is less than $f(a)$ and $f(v)$, then the multiset $\mathfrak{extend}(M, v, a, i)$ is also a w-candidate.

For a fixed word w, we define the transition graph $T_w = (\mathfrak{M}_w, E_w)$ is the directed graph defined as follows:

- its set of vertices $\mathfrak{M}_w$ is the set of all w-candidates,
- there is an edge $(M_1, M_2) \in E_w$ if there exists an integer i, a word v and a letter a that belongs to the underlying set of words of M_1, such that $M_2 = \mathfrak{extend}(M_1, v, a, i)$.

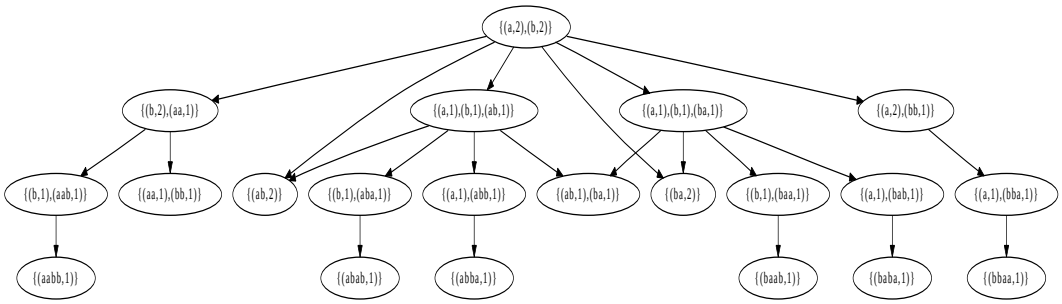

Figure 1: The transition graph associated to the word $w = abab$.

In the following section, we show the following properties of T_w:

- the transition graph is acyclic and its diameter is bounded by $|w|$.
- we describe a covering tree for the transition graph.
- the set of multisets M such that $w \in \bigsqcup\!\bigsqcup M$ forms a subtree of the covering tree.
- we detail how the covering tree can be trimmed according to additional predicates.

Lemma 1. *The graph T_w is acyclic and its diameter is bounded by $|w|$.*

Proof. Recall that there exists an edge $(M_1, M_2) \in E_w$ in the transition graph if and only if there exists a word w, a letter a and an integer i such that $M_2 = \mathfrak{extend}(M_1, v, a, i)$. The number of occurrences of the letter a in M_2 is strictly less that the number of occurrences of the letter a in M_1. Therefore if there exists a path from a multiset $\langle X, f\rangle$ to a multiset $\langle Y, g\rangle$ in the transition graph, then there exists a letter $a \in X \cap \Sigma$ such that $f(a) > g(a)$. Therefore $\langle X, f\rangle \neq \langle Y, g\rangle$ and the transition graph is acyclic. Also, since the sum of the number of occurrences of each letter in a w-candidate is bounded by $|w|$, the longest path in the transition graph is bounded by $|w|$. This concludes the proof. □ □

4.1. Covering Tree

In the following, we characterize a particular multiset named M_w^0 and a subset of transitions $F_w \subseteq E_w$. We prove that F_w forms a covering tree for T_w, whose root is M_w^0.

Definition 7. *Let $w \in \Sigma^+$ be a word, $q \in \mathbb{N}^*$ a quorum, $M = \langle X, f\rangle$ a w-candidate, $v \in X$ a word of M, $a \in X \cap \Sigma$ be a letter. We define the set of integers $\mathcal{I}_w(M, v, a)$ as:*

$$\mathcal{I}_w(M, v, a) = \{i \mid \mathfrak{extend}(M, v, a, i) \in \mathfrak{M}_w\}$$

Definition 8. *Let $F_w \subseteq E_w$ be a set of edges such that for all multisets $\langle X, f\rangle, \langle Y, g\rangle \in \mathcal{M}$, the edge $(\langle X, f\rangle, \langle Y, g\rangle)$ is in F_w if there exist a word $v \in X$, a letter $a \in X$ and an integer i such that:*

- $\langle Y, g\rangle = \mathfrak{extend}(\langle X, f\rangle, v, a, i)$,
- $va = \max_{mil}(Y)$,
- $i = \min(\mathcal{I}_w(\langle X, f\rangle, v, a))$.

Note that using the proper data structures, it can be determined in constant time whether va is maximal in the military order and $\min(\mathcal{I}(\langle X, f\rangle, v, a, P))$ can be computed in constant time.
We define the multiset M_w^0 as follow:

- its underlying set of words is Σ on which w is defined,
- for each letter $a \in \Sigma$, the multiplicity of a in M_w^0 is equal to $\|w\|_a$.

M_w^0 is the only w-candidate whose underlying set of words contains only letters. Indeed, given a multiset $\langle X, f\rangle$ different from M_w^0, if $X \subsetneq \Sigma$, then it is not a w-candidate. Also, if $X = \Sigma$, then there exists a letter $a \in X$ such that $f(a) \neq \|w\|_a$ and then the multiset is not a w-candidate either.

Lemma 2. *For all multiset $M_2 = \langle X, f\rangle$ such that $M_2 \in \mathfrak{M}_w \setminus M_w^0$, there exists a unique multiset $M_1 \in \mathfrak{M}_w$ such that $(M_1, M_2) \in F_w$.*

Proof. First of all, the operation $\mathfrak{extend}$ returns a multiset that contains at least a word of length 2, therefore M_w^0 cannot be obtained from another multiset.

Then for all multisets $M = \langle X, f\rangle \in \mathfrak{M}_w \setminus M_w^0$, there exists a word va of length at least 2 such that $va = \max_{mil}(Y)$ and there is at least one w-candidate $M' \in \mathfrak{M}_w$ and such that $M = \mathfrak{extend}(M', v, a, f(va))$. Note that v and a are uniquely defined.

Let $\langle Y, g\rangle, \langle Z, h\rangle \in \mathfrak{M}_w$ be two multisets such that $(\langle Y, g\rangle, M) \in F_w$ and $(\langle Z, h\rangle, M) \in F_w$. Let's now suppose that $\langle Y, g\rangle \neq \langle Z, h\rangle$. For all word $u \in X \setminus \{v, a, va\}$, we have $g(u) = h(u)$, since the operator $\mathfrak{extend}$ only modifies the number of occurrences of words a, v and va. Therefore if $\langle Y, g\rangle \neq \langle Z, h\rangle$, then we have $g(va) \neq h(va)$, or $g(v) \neq h(v)$, or $g(a) \neq h(a)$.

If $g(va) = h(va)$, then either $g(a) \neq h(a)$ or $g(v) \neq h(v)$. We have $i = \min(\mathcal{I}_w(\langle Y, g\rangle, v, a)) = \min(\mathcal{I}_w(\langle Z, h\rangle, v, a))$ since $g(va)+i = h(va)+i = f(va)$. Though, either $g(a)-i \neq h(a)-i$ or $g(v)-i \neq h(v)-i$, which is a contradiction.

If $g(va) < h(va) < f(va)$, then $g(a) > h(a) > f(a)$ and $g(v) > h(v) > f(a)$. Therefore the integer i such that $M = \mathfrak{extend}(\langle Y, g\rangle, v, a, i)$ is not the minimal value in $\mathcal{I}_w(\langle Y, g\rangle, v, a)$. This is also a contradiction.

This concludes the proof since we cannot have $\langle Y, g\rangle \neq \langle Z, h\rangle$. □ □

Corollary 1. *For a fixed word w and a quorum q, the subset F_w forms a covering tree for the transition graph T_w. Its root is M_w^0 and its height is at most $|w|$.*

Proof. Recall that according to Lemma 1, T_w is acyclic and its longest path is of length less than $|w|$. Also according to Lemma 2, for all multisets $M \in \mathfrak{M}_w \setminus M_w^0$, there exists a unique multiset M' and unique words v and a such that $(M', M) \in F_w$. Moreover, there is no multiset $M \in \mathfrak{M}_w$ such that $(M, M_w^0) \in F_w$. Therefore, there exists a unique path from M_w^0 to any multiset M and this path is of length at most $|w|$. This concludes the proof. □ □

4.2. An antimonotonic predicate

In the following lemma, we show that the set of multisets M such that $w \in \bigsqcup\!\bigsqcup M$ forms a subtree of the covering tree.

Lemma 3. *Let w be a non-empty word. For all multisets $M_1, M_2 \in \mathfrak{M}_w$ such that $(M_1, M_2) \in E_w$, if $w \in \bigsqcup\!\bigsqcup M_2$ is then $w \in \bigsqcup\!\bigsqcup M_1$.*

Proof. Given a multiset M, a word v and a letter a, we have

$$va \in v \sqcup\!\sqcup a$$
$$\implies \left(\bigsqcup\!\bigsqcup M \sqcup\!\sqcup va\right) \subset \left(\bigsqcup\!\bigsqcup M \sqcup\!\sqcup v \sqcup\!\sqcup a\right).$$

By induction, for all integers $i \in \mathcal{I}(M, q, v, a)$,

$$\left(\bigsqcup\!\bigsqcup M \sqcup\!\sqcup^i va\right) \subset \left(\bigsqcup\!\bigsqcup M \sqcup\!\sqcup^i v \sqcup\!\sqcup^i a\right),$$

which can be rewritten as follow:

$$\left(\bigsqcup\!\bigsqcup \mathcal{M} \oplus \{(va, i)\} \ominus \{(v, i)\} \ominus \{(a, i)\}\right) \subset \bigsqcup\!\bigsqcup \mathcal{M}$$

$$w \in \bigsqcup\!\bigsqcup \mathfrak{extend}(M, v, a, i) \implies w \in \bigsqcup\!\bigsqcup M$$

□ □

This result is useful in two ways. First of all, if a multiset M does not partition the word w, then there is no need to check all multisets that can be reached by M in the covering tree. A multiset M is more specific than a multiset M' if M' can be reached from M in the covering tree. A multiset that partitions w is said to be maximal if and only if there is no more specific multiset that partitions w. The set of maximal multisets contains as much information as the set of all multisets that partitions w and the first one can be used to retrieve the second one. Therefore, one could be interested in listing only maximal multisets in order to reduce the quantity of information to analyse afterwards.

Figure 4.2 shows an example of the covering tree and maximal multisets for $w = abab$.

4.3. Compute the set of edges

In the following, we study characterize candidate multisets that are obtained using a transition from a given w-candidate.

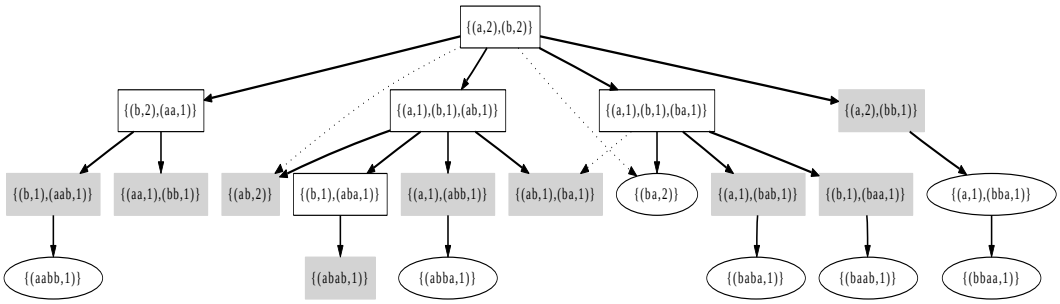

Figure 2: The transition Graph T_{abab}. The edges of the covering tree are bold are the others are dotted. Square boxes contain valid multisets and boxes filled with grey are maximal valid multisets.

The frequency predicate. For a fixed multiset $M = \langle X, f\rangle$, a frequency function $threshold : \Sigma^* \mapsto \mathbb{N}$, two words $v, a \in X$, the set $\mathcal{I}(M, v, a, threshold)$ is the set of integers such that the multiset obtained by the operation $\mathfrak{extend}(M, v, a, i)$ satisfies the predicate $\mathfrak{p}_{freq}$. Since the operation $\mathfrak{extend}$ modifies the number of occurrences of words v, a and va, $\mathcal{I}(M, v, a, threshold)$ is the set of integers i such that, the values $f(v) - i$, $f(a) - i$ and $f(va) + i$ are either equal to 0, or greater than the value obtained with the frequency function $freq$.

Example 4. *Given a multiset $M = \{(a, 3), (b, 2), (bc, 5)\}$, and a frequency function such that $freq(a) = 2$ for all $a \in \Sigma$, the set $\mathcal{I}(M, bc, a, freq)$ is equal to $\{3\}$. Indeed, one needs to add at least 2 occurrences of the words bca or the obtained multiset will not satisfy $\mathfrak{p}_{freq}$. If $i = 2$, we have $\mathfrak{extend}(M, bc, a, 2) = \{(a, 1), (b, 2), (bc, 3), (bca, 2)\}$. This multiset does not satisfy $\mathfrak{p}_{freq}$ because there is only one occurrence of the word a.*
If $i = 3$, we have $\mathfrak{extend}(M, v, a, i) = \{(b, 2), (bc, 2), (bca, 3)\}$, which satisfies $\mathfrak{p}_{freq}$

In Lemma 4, we give a characterization of $\mathcal{I}(M, v, a, threshold)$ for $v \neq a$ and in Lemma 5, we do the same for $v = a$.

Lemma 4. *Let $w \in \Sigma^+$ be a word, $threshold : \Sigma^* \mapsto \mathbb{N}$ a frequency function, $M = \langle X, f\rangle$ a w-candidate satisfying $\mathfrak{p}_{freq}$, $v \in X$ a word of M, $a \in X \cap \Sigma$ be a letter, such that $v \neq a$. The set $\mathcal{I}(M, v, a, threshold)$ is equal to[2]:*

1. $[\max\{threshold(va) - f(va), 1\}, \ldots, \min\{f(v) - threshold(v), f(a) - threshold(a)\}] \cup \min\{f(v), f(a)\}$, *if either $f(v) = f(a)$ or $f(v) - f(a) \geq threshold(v)$ or $f(a) - f(v) \geq threshold(a)$*
2. $[\max\{threshold(va) - f(va), 1\}, \ldots, \min\{f(v) - threshold(v), f(a) - threshold(a)\}]$, *otherwise.*

[2]Here, we consider that if the maximum value of an interval is less than its minimum value, then the interval is empty

Proof. For all word $u \in X$, we have $f(u) \geq q$, since M satisfies $\mathfrak{p}_{freq}$. Let $i \in \mathbb{N}^*$ and $M' = \mathfrak{extend}(M, v, a, i)$. We show that M' satisfies $\mathfrak{p}_{freq}$ iff i belongs to the intervals described in the lemma. We partition $\mathbb{N}^*$ into five different intervals in order to prove the result:

1. if $i < \max\{threshold(va) - f(va), 1\}$, then M' does not satisfy $\mathfrak{p}_{freq}$. Indeed, if $\max\{threshold(va) - f(va), 1\} = 1$, then $i \notin \mathbb{N}^*$ which is a contradiction. Also, if $\max\{threshold(va) - f(va), 1\} = threshold(va) - f(va)$, then $f(va) = 0$ (since $f(va)$ is either equal to 0 or greater than $threshold(va)$) and $i < threshold(va)$, which means that $0 < f'(va) < threshold(va)$ and the multiset M' does not satisfy $\mathfrak{p}_{freq}$.
2. if $i \in [\max\{threshold(va) - f(va), 1\}, \ldots, \min\{f(v) - threshold(v), f(a) - threshold(a)\}]$, then M' is a w-candidate. Indeed we have,
   - either $f(va) = 0$ or $f(va) \geq threshold(va)$. If $f(va) = 0$, then $i \geq threshold(va)$. If $f(va) \geq threshold(va)$, then $i \geq 1$. In both cases, $f'(va) \geq threshold(va)$.
   - $f'(v) \geq threshold(v)$ and $f'(a) \geq threshold(a)$ since $i \leq \min\{f(v) - threshold(v), f(a) - threshold(a)\}$ and $f(v) - \min\{f(v), f(a)\} + q \geq q$.
3. if $\min\{f(v) - threshold(v), f(a) - threshold(a)\} < i < \min\{f(v), f(a)\}$, then either $0 < f'(v) < threshold(v)$ or $0 < f'(a) < threshold(a)$ and M' is not a w-candidate,
4. if $i = \min\{f(v), f(a)\}$, then M' satisfies $\mathfrak{p}_{freq}$ if and only if
   - either $f'(v) = 0$ and $f'(a) \geq threshold(a)$, or $f'(v) \geq threshold(v)$ and $f'(a) = 0$.
   - either $f'(v) = 0$ and $f'(a) = 0$, meaning that $f(v) = f(a) = \min\{f(v), f(a)\}$.
5. if $i > \min\{f(v), f(a)\}$, then either $i > f(v)$ or $i > f(a)$ and the sum of the occurrences of each letter is not the same for M and M', which means that M' is not a a w-candidate.

□ □

Lemma 5. *Let $w \in \Sigma^+$ be a word, $q \in \mathbb{N}^*$ a quorum, $M = \langle X, f \rangle$ a w-candidate and $a \in X \cap \Sigma$ be a letter. The set $\mathcal{I}(M, a, a, threshold)$ is equal to:*

1. $[\max\{threshold(va) - f(va), 1\}, \ldots, \lfloor \frac{f(a) - threshold(a)}{2} \rfloor] \cup \frac{f(a)}{2}$, *if $f(a)$ is even and $f(va) + \frac{f(a)}{2} \geq threshold(va)$.*
2. $[\max\{threshold(va) - f(va), 1\}, \ldots, \lfloor \frac{f(a) - threshold(a)}{2} \rfloor]$, *otherwise.*

Proof. The proof of this lemma is similar to the proof of Lemma 4. □

Other predicates. The predicates $\mathfrak{p}_{nb}$ and $\mathfrak{p}_{length}$ are antimonotonic when used with relation $\leq$ or $<$ and monotonic with relation $\geq$ or $>$. The test of those predicates can be made locally, when the algorithms add a word va. Using proper data structure, to check whether the constraint is satisfied can be made in constant time.

The monotonicity of the predicate $\mathfrak{p}_{lang}$ depends on the language recognized by the finite state machine. If for all word $u \in \mathcal{L}$, all prefixes of u are also in $\mathcal{L}$, then the predicate is antimonotonic. If for all word $u \in \mathcal{L}$, all suffixes of u are also in $\mathcal{L}$, then the predicate is monotonic. If the predicate is not antimonotonic, it is needed to check, each time a word va is added, whether all the words of the multiset are in $\mathcal{L}$.

5. Algorithm and Complexity Analysis

5.1. Presentation of the algorithm

Our algorithm searches the covering tree $(\mathfrak{M}_w, F_w)$ in a depth first manner. This guarantees that a frequent multiset is generated at most once. Also, Lemma 3 guarantees that all frequent multisets are generated.

Algorithm 1: Generator(M, P, w)

```
   Data: A multiset M = ⟨X, f⟩, a set of predicates P, a word w.
 1 begin
 2   forall v ∈ X do
 3     forall a ∈ X ∩ Σ do
 4       if va = max_mil{X ∪ va} then
 5         if I(M, v, a) ≠ ∅ then
 6           i = min(I(M, v, a))
 7           M' = extend(M, v, a, i)
 8           if M is (w, P)-valid then
 9             Generator(M', P, w)
10           end
11         end
12       end
13     end
14   end
15 end
```

The algorithm *Generator* starts with the multiset M_w^0.

Theorem 1. *Let w be a word over a finite alphabet Σ and a set of predicates P. The algorithm Generator generates all (w, P)-valid multisets without redundancy.*

5.2. Complexity of the algorithm

The general case.. The complexity of the algorithm *Generator* mostly depends on the complexity of the problem of checking whether a multiset is (w, P)-valid, for a fixed set of predicates P. This problem is known to be NP-complete in the general case, according to the following theorem.

Theorem 2. *[10] Let X be a set of words and w a word. The problem of deciding if $w \in \bigsqcup_{v \in X} v$ is NP-complete.*

Since a set of words is a particular case of a multiset of words, Theorem 2 also holds for multiset of words. Using a result of [7], we can also show that checking whether a multiset M partitions a word w remains an NP-complete problem even if we use the restriction $\mathfrak{p}_{dist}$ with a distance greater or equal to 1.

In the following, we prove that unless $P = NP$, there is no polynomial time algorithm to check if a multiset partitions a word, even knowing that its predecessors in the transition graph do.

Problem :. Incremental Shuffle Product Checking

Input :. an alphabet Σ, a word $w \in \Sigma^+$, a multiset M that partitions w and a multiset M' such that $(M, M') \in F_w$.

Question :. Does M' partitions w?

Theorem 3. *Unless $P = NP$, there is no polynomial algorithm to solve Incremental Shuffle Product Checking.*

Proof. Suppose there exists a polynomial algorithm to solve Incremental Shuffle Product Checking. According to Corollary 1, for all w-candidate multiset M, there exist a path of length less than $|w|$ from the multiset M_w^0 to the multiset M. Such a path can be computed in polynomial time and stored in polynomial space, starting from M and computing each predecessor $\langle X, f\rangle$ of a multiset $\langle Y, g\rangle$ in the covering tree, that is to say

$$\langle X, f\rangle = \langle Y, g\rangle \ominus \{(va, i\} \oplus \{(v, i)\} \oplus \{(a, i)\},$$

where $va = \max_{mil}(Y)$ and i is the smallest integer such that $\langle X, f\rangle \in \mathfrak{M}_w$. Recall that by definition, M_w^0 partitions the word w. If there is a polynomial algorithm to solve Incremental Shuffle Product Checking, then there is a polynomial algorithm to decide whether M partitions w. Though, according to Theorem 2 such a decision problem is NP-complete. Unless $P \neq NP$, there is a contradiction. □ □

Corollary 2. *The algorithm Generator has an exponential time complexity and a polynomial space complexity in the general case.*

Theorem 4. *For a fixed set of predicates P, $P' \subseteq P$ a subset of antimonotonic predicates and a fixed word w, let $\mathcal{M}_{w,P'}$ be the set of (w, P')-valid multisets. Suppose the oracle that check whether a multiset M is (w, P)-valid has a time complexity $t(|M|, |w|, P)$, then the algorithm Generator has*

$$\mathcal{O}(t(|M|, |w|, P) \times |w|^2 \times |\Sigma| \times |\mathcal{M}_{w,P'}|)$$

time complexity and polynomial space complexity.

Proof. Each call costs $|w|^2 \times |\Sigma|$ since to check whether va is maximal in the military order can be done in $\mathcal{O}(|w|)$. Indeed the words of M are totally ordered and so it suffices to compare va to the maximal one. The smallest $i \in \mathcal{I}_w(M, v, a)$ can be computed in constant time and the update of M can be done in linear time. From Lemma 1, it is known that the depth of recursive calls is bounded by $\mathcal{O}(|w|)$. The result follows. □ □

Particular cases.. For a fixed $k \in \mathbb{N}^*$, if a multiset M satisfies $\mathfrak{p}_{nb}(M, \leq, k)$, then it can be checked in $\mathcal{O}(|w|^k)$ time whether M partitions a word w. Indeed, an automaton with at most $\mathcal{O}(|w|^k)$ states, recognizing $\bigsqcup M$, can be built. This includes classical sequence mining.

[1] R. Agrawal and R. Srikant. Mining sequential patterns. In *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, pages 3–14. IEEE, 1995.

[2] Hiroki Arimura and Takeaki Uno. Polynomial-delay and polynomial-space algorithms for mining closed sequences, graphs, and pictures in accessible set systems. In *SDM*, pages 1087–1098. SIAM, 2009.

[3] Julien David. The average complexity of moore's state minimization algorithm is O(n log log n). In *MFCS*, pages 318–329, 2010.

[4] Jun Huan, Wei Wang, and Jan Prins. Efficient Mining of Frequent Subgraphs in the Presence of Isomorphism. In *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM '03, pages 549–553, Washington, DC, USA, 2003. IEEE Computer Society.

[5] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Min. Knowl. Discov.*, 1:259–289, January 1997.

[6] Nadia Pisanti, Maxime Crochemore, Roberto Grossi, and Marie-France Sagot. Bases of motifs for generating repeated patterns with wild cards. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 2:40–50, January 2005.

[7] Romain Rivière, Dominique Barth, Johanne Cohen, and Alain Denise. Shuffling biological sequences with motif constraints. *J. of Discrete Algorithms*, 6:192–204, June 2008.

[8] D. Singh, A. M. Ibrahim, T. Yohanna, and J. N. Singh. An overview of the applications of multisets. *Novi Sad J. Math*, 37(2):73–92, 2007.

[9] Roberto Trasarti, Francesco Bonchi, and Bart Goethals. Sequence mining automata: A new technique for mining frequent sequences under regular expressions. In *ICDM*, pages 1061–1066. IEEE Computer Society, 2008.

[10] Manfred K. Warmuth and David Haussler. On the complexity of iterated shuffle. *Journal of Computer and System Sciences*, 28(3):345–358, 1984.