A Unifying Polynomial Model for Efficient Discovery of Frequent Itemsets [*]

Slimane Oulad-Naoui[1], Hadda Cherroun[2] and Djelloul Ziadi[3]

[1]*Département des Mathématiques et d'Informatique, Université de Ghardaia, Ghardaia, Algeria*

[2]*Laboratoire d'Informatique et de Mathématiques, Université Amar Telidji, Laghouat, Algeria*

[3]*Laboratoire LITIS - EA 4108, Normandie Université, Rouen, France*

Keywords: Data Mining, Frequent Itemsets, Formal Series, Weighted Automata, Algorithms, Unification.

Abstract: It is well-known that developing a unifying theory is one of the most important issues in Data Mining research. In the last two decades, a great deal has been devoted to the algorithmic aspects of the Frequent Itemset (FI) Mining problem. We are motivated by the need of formal modeling in the field. Thus, we introduce and analyze, in this theoretical study, a new model for the FI mining task. Indeed, we encode the itemsets as words over an ordered alphabet, and state this problem by a formal series over the counting semiring $(\mathbb{N},+,\times,0,1)$, whose the range constitutes the itemsets and the coefficients their supports. This formalism offers many advantages in both fundamental and practical aspects: The introduction of a clear and unified theoretical framework through which we can express the main FI-approaches, the possibility of their generalization to mine other more complex objects, and their incrementalization and/or parallelization; in practice, we explain how this problem can be seen as that of word recognition by an automaton, allowing an efficient implementation in $O(|Q|)$ space and $O(|\mathcal{F}_L||Q|])$ time, where Q is the set of states of the automaton used for representing the data, and $\mathcal{F}_L$ the set of prefixial maximal FI.

1 INTRODUCTION

Mining Frequent Itemsets (FI) is an important problem in Data Mining (DM). Although primitive, it constitutes one of the most challenging and over-two-decade-well-studied subject in the field. Since the introduction of the Apriori algorithm by Agrawal (Agrawal and Srikant, 1994), several algorithms have been proposed to solve it. Without claim of exhaustiveness, we can categorize these works into three main classes, for more detail see (Hipp et al., 2000; Goethals and Zaki, 2003; Han et al., 2007): (*i*) Enumeration of all FI, (*ii*) Discovery of closed/maximal FI, and (*iii*) Incremental algorithms.

The first class of algorithms aims to extract the whole set of FI. The problem space exploration approaches used can be distinguished by the traversal and the support calculation methods (Hipp et al., 2000). In level-wise techniques, a breadth-first traversal is adopted, where a k-itemset is derived by extending a frequent one of length $k-1$ (Agrawal and Srikant, 1994). The calculation of the support of an itemset is performed by database scans. In these techniques, the most remarkable is, undeniably, the A-priori heuristic used to prune the problem space, and widely used in all algorithms later. Unfortunately, these techniques suffer from two major drawbacks: Generating a huge number of candidates, and an excessive I/O cost needed for support counting.

In (Zaki, 2000), the author considered the data from a vertical angle of view, where he associated with each itemset X its list of transactions (tidlist) and uses set intersection for support calculation, which has proven to be a more effective trick. However, and despite the common-prefix equivalence relation proposed to decompose the problem, this approach requires, in dense datasets particularly, a large time and intermediate memory to perform intersections.

In order to reduce the size of the dataset and avoid multiple scans of it, Han et al. introduced the FPGrowth algorithm (Han et al., 2004) that uses a compact structure called Frequent-Pattern Tree (FP-Tree) enhanced with the itemset supports. This algorithm generates recursively a grown-pattern conditional database projections for which a correspond-

[*] This work is supported by the MESRS-Algeria under the project number 8/U03/7015

Oulad-Naoui S., Cherroun H. and Ziadi D.
A Unifying Polynomial Model for Efficient Discovery of Frequent Itemsets.
DOI: 10.5220/0005516200490059
In *Proceedings of 4th International Conference on Data Management Technologies and Applications* (DATA-2015), pages 49-59
ISBN: 978-989-758-103-8

ing FPTrees are also constructed. Though, the performance gain shown, the original version of FPGrowth induces, sadly, an abundant memory and time overhead due to repetitive sorting and reconstructions.

The algorithms of the second class focus on the minimum set of FI, called the cover, which allows to generate all the rest (Pasquier et al., 1999). Thereby, the closed and maximal FI notions have been introduced. These approaches use Formal Concept Analysis (FCA) (Wille, 1982) to extract the set of frequent concepts, that constitutes a condensed representation of the entire set of FI.

The concern of the algorithms of the third class was the incrementality. That is, how to generate the set of FI, and to maintain it in the case of dynamic datasets (Valtchev et al., 2008). Here, the same philosophies were adopted in either algorithmic fashion or using FCA.

Summing up, after more than two decades of active research on the subject, with countless techniques including various efficient algorithms and judicious data structures each with its benefits and drawbacks, we believe that it will be convenient to go back and ask a key question: Besides the existing ones (Godin et al., 1995; Zaki and Ogihara, 1998), are there other formalisms for this basic problem ? In other words, we aim to develop a general unifying model able to express the works done so far in the main state of the art approaches. We wish, moreover, that the proposed formalism should enjoy some capitals characteristics such as : the completeness while remaining simple and intuitive, extensibility, and efficiency. That is, provide, why not, an implementation having a better performances, if not stand at least comparable to those of the existing techniques.

The elaboration of unifying models is a well-established issue in DM (Yang and Wu, 2006). We postulate that the unification can be facilitated if we focus on a particular DM-task. In this paper, we address this question for the FI-mining problem. Indeed, we introduce a new model for enumerating all FI based on formal series, which meets the above proprieties. First, it defines a unified theoretical framework, which leads to see the equivalence of the algorithms as stipulated in (Goethals and Zaki, 2003) and confirmed in one of the early comparative studies (Hipp et al., 2000). Second, it allows their generalization for mining more complex objects. We prove also a natural decomposition scheme, often required in many aspects of the problem such that incrementality or parallelization. Moreover, we explain how this problem can be transposed to that of the realization of a formal series by a weighted automaton (Salomaa et al., 1978), and consequently, to that of word recognition, which is a largely invested topic with a very mature algorithmic. Finally, we propose an efficient algorithm to enumerate all FI, which runs in place without extra memory.

The remaining of this paper is organized as follows. We begin, in Section 2, by some preliminaries on the basic concepts and notions to be used throughout this article. In Section 3, we recall the FI mining problem, and introduce our model. Section 4 is devoted to the definition, the proofs, the construction of the proposed automaton, and the analysis of the mining algorithm. In Section 5, we discuss our model against the existing techniques and show how these can be derived from it, and conclude in Section 6 with some extensions.

2 PRELIMINARIES

A set $\mathbb{M}$ with an associative binary internal operation $*$ admitting a unique $e \in \mathbb{M}$ as an identity element forms a structure of *monoid*, which we denote $(\mathbb{M}, *, e)$. When the operation $*$ is also commutative then the monoid is commutative. The popular example is the free monoid A^* of the set of words over an alphabet A equipped with the concatenation of two words, and having the empty word ε as an identity element.

A word u is a *prefix* (respectively a *suffix*) of a word w if there exists a word v such that $w = uv$ (respectively $w = vu$). The set of the prefixes of a word u will be denoted $\mathrm{Pref}(u)$. This concept can be extended to a set of words by performing the union of the prefixes of its elements. A word $u = u_1 \dots u_k$ is a *subsequence* of a word $w = w_1 \dots w_l$ $(k \leq l)$ if there exist words $v_1, \dots, v_{k+1}$, such that $w = v_1 u_1 v_2 u_2 \dots v_k u_k v_{k+1}$. We write then $u \preccurlyeq w$.

A *semiring* is a tuple $(\mathbb{K}, +, \times, 0, 1)$ such that: $(\mathbb{K}, +, 0)$ is a commutative monoid, $(\mathbb{K}, \times, 1)$ is a monoid, $\times$ distributes on both sides over $+$, and 0 is an absorbing element with respect to $\times$. Examples of semirings are $(\mathbb{N}, +, \times, 0, 1)$ of positive integers, $(\mathbb{B}, \vee, \wedge, \perp, \top)$ of booleans, and the tropical semiring $(\mathbb{N} \cup \{\infty\}, min, +, \infty, 0)$.

Over the monoid A^*, we define a *formal series* $\mathbb{S}$ with coefficients in a semiring $\mathbb{K}$ as a mapping $\mathbb{S} : A^* \rightarrow \mathbb{K}$, which associates with each w its *coefficient* $\langle \mathbb{S}, w \rangle$. The series $\mathbb{S}$ itself will be written as a sum:

$$\mathbb{S} = \sum_{w \in A^*} \langle \mathbb{S}, w \rangle w \qquad (2.1)$$

The set $\mathrm{range}(\mathbb{S}) = \{w \in A^* | \ \langle \mathbb{S}, w \rangle \neq 0\}$ of words with non-null coefficients is called the *range* of the series $\mathbb{S}$ (also called its support, but we prefer range to avoid the confusion with the support of an itemset). The set of formal series over A with coefficients on $\mathbb{K}$ is denoted $\mathbb{K}\langle\langle A\rangle\rangle$. A structure of a semiring is defined on $\mathbb{K}\langle\langle A\rangle\rangle$ as follows, $\mathbb{S}$ and $\mathbb{T}$ are two formal series on A with coefficients in $\mathbb{K}$:

$$\langle \mathbb{S}+\mathbb{T}, w\rangle = \langle \mathbb{S}, w\rangle + \langle \mathbb{T}, w\rangle \tag{2.2}$$

$$\langle \mathbb{S}\mathbb{T}, w\rangle = \sum_{uv=w} \langle \mathbb{S}, u\rangle\langle \mathbb{T}, v\rangle \tag{2.3}$$

The subset of the series of $\mathbb{K}\langle\langle A\rangle\rangle$ with finite range are called *polynomials* and denoted by $\mathbb{K}\langle A\rangle$. Thereafter, in the case of the monoid A^*, and for its identity element ε, and for all $k \in \mathbb{K}$ we write $k\varepsilon$ (or simply k) the term having k as coefficient of ε. In the same way, for a word w on A^*, we denote kw (respectively w) the term whose coefficient for w is k (respectively 1).

A *weighted automaton* $\mathcal{A}$ over an alphabet A with coefficients in a semiring $\mathbb{K}$ is a tuple $\mathcal{A} = (Q, A, \mu, \lambda, \gamma)$, where Q is the finite set of states, μ the function from Q to $\mathbb{K}$ of the initial (input) weights, λ the function from $Q \times A \times Q$ to $\mathbb{K}$ of the transition weights, and γ the mapping from Q to $\mathbb{K}$ of the final (output) weights. A path c in $\mathcal{A}$ is a succession of transitions: $(q_0, a_1, q_1)\ldots(q_{n-1}, a_n, q_n)$ labeled by the word $a_1 \ldots a_n$ obtained by the concatenation of the symbols of its edges. Its weight is the product of the weights of its transitions:

$$\omega(c) = \mu(q_0)\lambda(q_0, a_1, q_1)\ldots\lambda(q_{n-1}, a_n, q_n)\gamma(q_n) \tag{2.4}$$

If we denote by $\mathcal{C}(u)$ the set of all paths labeled u. The weight of a word u in the automaton $\mathcal{A}$, denoted $\mathcal{A}(u)$, is the sum of the weights of the elements of $\mathcal{C}(u)$:

$$\mathcal{A}(u) = \sum_{c \in \mathcal{C}(u)} \omega(c) \tag{2.5}$$

The size of an automaton is the number of its transitions.

3 PROBLEM STATEMENT AND NOTATIONS

First, let us recall the basic concepts of the FI mining problem, and introduce the definitions and notations used throughout this paper.

Let $A = \{a_1, a_2, \ldots, a_m\}$ be an alphabet of m symbols called *items*. Those can designate, according to the application domain, a products purchased from a supermarket, a visited Web pages, a collection of attributes…etc. An *itemset* is a subset of A, if k is its cardinal it is called a k-itemset. A *transaction* t_i is a nonempty set of items identified by its unique identifier i. A *dataset D* is a set of n transactions, which we denote as a multi-set: $D = \{t_1, t_2, \ldots, t_n\}$. In a dataset D, the *support* of an itemset x, denoted $\mathrm{sprt}(x, D)$, is the number of transactions containing x, i.e :

$$\mathrm{sprt}(x, D) = |\{t_k \in D \mid x \subseteq t_k\}| \tag{3.1}$$

An itemset x is *frequent* if its support exceeds a specified minimum support-threshold s. That is, x is frequent in D if and only if $\mathrm{sprt}(x, D) \geq s$. A frequent itemset is *maximal* if an only if there is no superset of it which is frequent. The problem of mining FI consists to discover the set $\mathcal{F}$ of all itemsets whose support is greater than the given minimum support-threshold s.

3.1 The Polynomial Model

Now, we show how to translate the FI mining problem to the formal series model. Taking into account the finiteness of the modeled data (itemsets and datasets), we adopt thus a modeling based on polynomials.

The main idea in this modeling is to encode an itemset by a word, and all its subsets by a polynomial. After defining the polynomial of a dataset, the question is then to extract from this polynomial all the terms where the support-criterion holds.

First, let us assume, without loss of generality, that the alphabet A is sorted according to an arbitrary total order, where we can write:
$A = \{a_1, a_2 \ldots a_m\}$, with: $\varepsilon < a_1 < a_2 < \ldots < a_m$. We represent a k-itemset $x = \{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\}$ by the word $w(x)$ of length k, built by the concatenation of its items according to the predefined order. We will write:

$$w(x) = a_{i_1}a_{i_2}\ldots a_{i_k}, \text{ such that } a_{i_1} < \ldots < a_{i_k} \tag{3.2}$$

In what follows, we confuse an itemset x and its word representation $w(x)$. That is, instead of $x = \{a, b, c\}$, we write simply $x = abc$. Note that the empty itemset $\emptyset$ is represented by the empty word ε of length zero ($|\varepsilon| = 0$).

Definition 3.1 (Itemset Subsequence Polynomial)**.** *Let $x = a_{i_1}a_{i_2}\ldots a_{i_k}$ be a k-itemset, The subsequence polynomial $\mathbb{S}_x$ associated with x is defined as follows:*

$$\mathbb{S}_x = (a_{i_1}+1)(a_{i_2}+1)\ldots(a_{i_k}+1), \textit{ with: } \mathbb{S}_\varepsilon = 1 \tag{3.3}$$

Hereafter, we denote, for each $a \in A$, by $\overline{a}$ the polynomial $(a+1)$. So, the polynomial $\mathbb{S}_x$ associated with a k-itemset x will be denoted: $\mathbb{S}_x = \overline{a_{i_1}a_{i_2}\ldots a_{i_k}}$. So, $\mathbb{S}_x$ is the polynomial that represents all the subsets of x. For example, we associate with the itemset $x =$

abc the polynomial $\mathbb{S}_x = \overline{abc} = (a+1)(b+1)(c+1)$, that gives us the polynomial: $1+a+b+c+ab+ac+bc+abc$.

From the itemset subsequence polynomial, we can derive the subsequence polynomial associated with a dataset D.

Definition 3.2 (Dataset Subsequence Polynomial)**.** *Let $D = \{t_1, \ldots, t_n\}$ be a dataset. The subsequence polynomial $\mathbb{S}_D$ associated with D is the sum of the n subsequence polynomials of its transactions:*

$$\mathbb{S}_D = \sum_{i=1}^{n} \mathbb{S}_{t_i} \qquad (3.4)$$

It is obvious to see that the terms of the polynomial $\mathbb{S}_D$ have the form $\langle \mathbb{S}_D, w \rangle w$, where w is an itemset and $\langle \mathbb{S}_D, w \rangle$ a coefficient in $\mathbb{N}$ representing its support in the database. Indeed, an itemset have 1 as coefficient in the polynomial of the transaction t_i where it appears and, consequently, its coefficient in the database is then the number of the transactions where it occurs. To illustrate this concept let us consider a running example taken from (Zaki and Wagner Meira, 2014). Table 1, shows a database of six transactions, where the third column gives the subsequence polynomial of each transaction. We have calculated also, in the last line, the subsequence polynomial of the whole database. We can easily observe, in the example, that the itemsets: ε, e, bc, $acde$ have the supports : $6, 5, 4$, and 1 respectively.

3.2 General Algorithm

Now, we are given a polynomial $\mathbb{S}$ over an alphabet A with coefficients on a semiring $\mathbb{K}$, and a user specified minimum support-threshold s. We aim to extract the polynomial $\mathbb{F}$ from $\mathbb{S}$ defined as follows:

$$\langle \mathbb{F}, w \rangle = \begin{cases} \langle \mathbb{S}, w \rangle & \text{if } \langle \mathbb{S}, w \rangle \geq s, \\ 0 & \text{otherwise.} \end{cases} \qquad (3.5)$$

So, we look for all words from the range of the polynomial $\mathbb{S}$ having coefficients greater than s. The exploration of the problem space, exponential in nature, is performed by the generic Algorithm 1, which list the searched set of words by invoking DISCOVER-FI($\mathbb{S}, s, \varepsilon, \mathcal{F}$). Thanks to the Apriori property in Proposition 3.3, the problem space can be pruned. Note that since the frequentness is a relative notion, we keep in this work, in a similar way as many works (Cheung and Zaïane, 2003; Goethals, 2004) all the items, regardless of their initial frequencies. This make the model more flexible specially in dynamic datasets.

Algorithm 1: DISCOVER-FI($\mathbb{S}, s, w, \mathcal{F}$).

```
Require: The polynomial $\mathbb{S}$, the min. support-threshold $s$, and an itemset $w = w_1 w_2 \ldots w_{|w|}$
Ensure: The set of all frequent itemsets
  $\mathcal{F} \leftarrow \emptyset$                              ▷ Initialization
  for all $a > w_{|w|}$ do
      if $\langle \mathbb{S}, wa \rangle \geq s$ then
          $\mathcal{F} \leftarrow \mathcal{F} \cup \{(wa, \langle \mathbb{S}, wa \rangle)\}$
          DISCOVER-FI($\mathbb{S}, s, wa, \mathcal{F}$)
      end if
  end for
  return $\mathcal{F}$
```

Proposition 3.3 (A-priori (Agrawal and Srikant, 1994))**.** *Let D be a dataset and w_1, w_2 two itemsets. If $w_1 \preccurlyeq w_2$, then* $\mathrm{sprt}(w_1, D) \geq \mathrm{sprt}(w_2, D)$.

It is clear that the complexity of Algorithm 1 depends on the number of FI as well as the cost of the test of frequentness, which depends in turn on the itemset length and the calculation of its coefficient $\langle \mathbb{S}, w \rangle$ in the chosen data structure. In order to give efficient implementation of Algorithm 1, it is necessary to use an optimal data structure, which must have a reduced size and provides a minimal cost of coefficient calculation. In this work, we claim that the FI mining problem can be formulated using formal series which we realize by means of weighted automata (Salomaa et al., 1978).

4 FREQUENT ITEMSET WEIGHTED AUTOMATON

Let $\mathcal{S}_D$ be a weighted automaton recognizing the subsequence polynomial $\mathbb{S}_D$ associated with a dataset D as defined above. Calculate the coefficient $\langle \mathbb{S}_D, w \rangle$ of an itemset w in this polynomial is equivalent to determine its weight in the automaton $\mathcal{S}_D$. Consequently, the complexity of this calculation relies on the type of the automaton (deterministic, non-deterministic, asynchronous...etc) and its size. Hereafter, we propose a particular and reduced automaton w.r.t the size of the dataset D which realizes the polynomial $\mathbb{S}_D$.

For the purpose of the construction of the automaton $\mathcal{S}_D$, which we refer as FIWA for Frequent Itemset Weighted Automaton, and since the idea of overlapping common prefixes (prefix tree, trie, prefix relation or equivalence class, FPTree) has proven to be very effective in this problem (Zaki, 2000; Cheung and Zaïane, 2003; Han et al., 2004; Valtchev et al., 2008; Totad et al., 2012), we shall go through another type of automaton, which will help us to define our

Table 1: Transaction database and the associated polynomials.

i	t_i	$\mathbb{S}_{t_i}$
1	abde	$1+a+b+d+e+ab+ad+ae+bd+be+de+abd+ade+abe+bde+abde$
2	bce	$1+b+c+e+bc+be+ce+bce$
3	abde	$1+a+b+d+e+ab+ad+ae+bd+be+de+abd+ade+abe+bde+abde$
4	abce	$1+a+b+c+e+ab+ac+ae+bc+be+ce+abc+ace+abe+bce+abce$
5	bcd	$1+b+c+d+bc+bd+cd+bcd$
6	abcde	$1+a+b+c+d+e+ab+ac+ad+ae+bc+bd+be+cd+ce+de+abc+abd$ $+abe+acd+ace+ade+bcd+bce+bde+cde+abcd+abce+abde+bcde+acde+abcde$
		$\mathbb{S}_D = 6+4a+6b+4c+4d+5e+4ab+2ac+3ad+4ae+4bc+4bd+5be+2cd+3ce+3de$ $+2abc+3abd+4abe+acd+2ace+3ade+2bcd+3bce+3bde+cde$ $+abcd+2abce+3abde+bcde+acde+abcde$

intended automaton $\mathcal{S}_D$. This intermediate automaton is the prefixial weighted automaton $\mathcal{P}_D$ defined hereafter. But let us define, first, the prefixial polynomial.

Definition 4.1 (Itemset Prefixial Polynomial)**.** *Let $x = a_{i_1}a_{i_2}\dots a_{i_k}$ be a k-itemset, the prefixial polynomial associated with x is $\mathbb{P}_x$ defined as follows:*

$$\mathbb{P}_x = \sum_{u\in \mathrm{Pref}(x)} u \qquad (4.1)$$

That is, the prefixial polynomial is the sum of all the prefixes of the considered itemset. For example, the prefixial polynomial of the itemset abc is $\mathbb{P}_{abc} = 1+a+ab+abc$.

Definition 4.2 (Dataset Prefixial Polynomial)**.** *Let $D = \{t_1,\dots,t_n\}$ be a dataset. The prefixial polynomial $\mathbb{P}_D$ associated with D is the sum of the n prefixial polynomials of its transactions:*

$$\mathbb{P}_D = \sum_{i=1}^{n} \mathbb{P}_{t_i} \qquad (4.2)$$

Notice that the last definition induces that $\mathrm{range}(\mathbb{P}_D) = \mathrm{Pref}(D)$. In other words, the range of the prefixial polynomial of a dataset D is the set of the prefixes of its transactions. Below is the prefixial polynomial of the dataset of our running example, after some development: $\mathbb{P}_D = 6+4a+2b+4ab+2bc+2abc+2abd+bcd+bce+abcd+abce+2abde+abcde$.

4.1 Prefixial Weighted Automaton

At this level, we claim that the construction of a weighted automaton for the dataset subsequence polynomial $\mathbb{S}_D$ go through the construction of a weighted automaton for $\mathbb{P}_D$ the prefixial one. There exist many weighted automata that realize these polynomials. We give here, a particular deterministic weighted automaton which realizes the prefixial polynomial $\mathbb{P}_D$, then introduce a little change on it to get an automaton that realizes our initial dataset subsequence polynomial $\mathbb{S}_D$.

Definition 4.3 (Prefixial Weighted Automaton (PWA))**.** *Let $\mathbb{P}_D$ be the prefixial polynomial of a dataset D. The related prefixial weighted automaton $\mathcal{P}_D = (Q, A, \mu, \lambda, \gamma)$ is defined as follows:*

- *$Q = \mathrm{range}(\mathbb{P}_D)$,*
- *$\mu(u) = 1$, for $u = \varepsilon$, and 0 otherwise, for $u \in Q$,*
- *$\lambda(u, a, ua) = 1$, for u and $ua \in Q$, and $a \in A$,*
- *$\gamma(u) = \langle \mathbb{P}_D, u\rangle$, for $u \in Q$.*

Note that the weight of any path labeled u in a prefixial weighted automaton $\mathcal{P}_D$ is equal to $\gamma(u)$, since $\mu(\varepsilon) = 1$, and $\lambda(v, a, va) = 1$ for all $v, va \in Q$. In in order to alleviate the reading, an automaton $\mathcal{A}$ that realizes $\mathbb{P}_D$ is said, next, to be PWA if and only if it is isomorphic to $\mathcal{P}_D$, i.e: $(\mathcal{A} \cong \mathcal{P}_D)$. An automaton isomorphic to the prefixial weighted automaton associated with the dataset of our running example is displayed in Figure 1.

Lemma 4.4. *For a dataset D, the automaton $\mathcal{P}_D$ realizes the polynomial $\mathbb{P}_D$.*

Proof. By construction. It is not hard to notice that the boolean automaton derived from $\mathcal{P}_D$ (the later deprived from its weights) recognizes the range of the prefixial polynomial $\mathbb{P}_D$. Indeed, $\mathcal{P}_D$ have only one initial state ε and all the states are final and associated with words in the range of $\mathbb{P}_D$. Moreover, a transition, if it exist, from a state u is made by items of A leading to ua, which yet remains a word in the range of $\mathbb{P}_D$. Furthermore, The weight in the automaton of each word in the range of $\mathbb{P}_D$ is exactly its corresponding coefficient, since $\gamma(u) = \langle \mathbb{P}_D, u\rangle$.

Definition 4.3, introduces the prefixial weighted automaton of a dataset from its associated prefixial polynomial. In what follows, we give a construction procedure of this automaton, which can be done in batch or step by step either taking into account one transaction or a set of them. This process is a general incremental algorithm for the construction of a PWA associated with a dataset D.

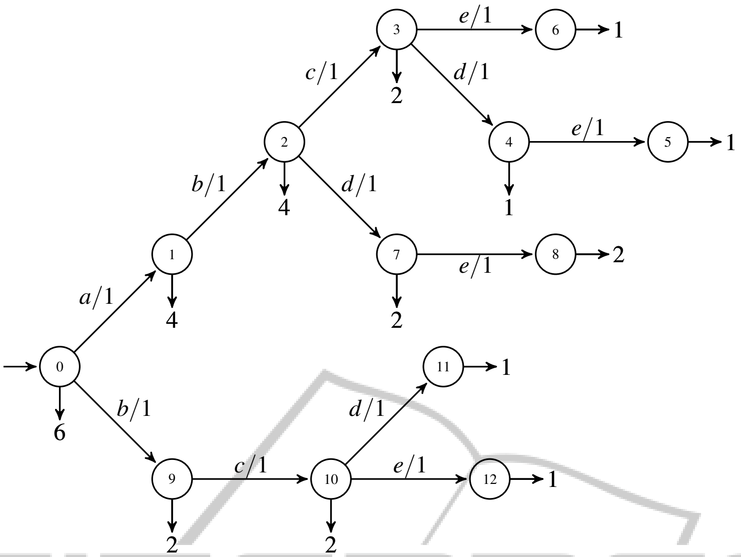


Figure 1: A PWA associated with our running example dataset.

Proposition 4.5. *Let $\mathcal{A}$ and $\mathcal{B}$ be two PWAs associated respectively with datasets X and Y. There exists a PWA $\mathcal{C}$ for the dataset $X \cup Y$ derived from $\mathcal{A}$ and $\mathcal{B}$.*

Proof. The idea is to construct the automaton $\mathcal{C}$ by determinizing both automata $\mathcal{A}$ and $\mathcal{B}$ using the accessible subset-construction procedure. We give below the definition of: R the set of states of the automaton $\mathcal{C}$, and γ the function of final weights (the functions μ and λ are obvious, and remain unchanged as seen in Definition 4.3 and depicted in Figure 2), and a mapping h from R to the set of states of $\mathcal{P}_{X\cup Y}$, which is the range of the polynomial $\mathbb{P}_{X\cup Y}$.
Let $\mathcal{A} = (P,A_1,\mu_1,\lambda_1,\gamma_1)$ be a PWA isomorphic, via h_1, to the automaton $\mathcal{P}_X$, and $\mathcal{B} = (Q,A_2,\mu_2,\lambda_2,\gamma_2)$ the one isomorphic, via h_2, to the automaton $\mathcal{P}_Y$. We define the PWA $\mathcal{C} = (R,A_1 \cup A_2,\mu,\lambda,\gamma)$ as follows ($p \in P$ and $q \in Q$):

- Set of states : $R = \{\{p,q\} \mid h_1(p) = h_2(q)\} \cup \{\{p\} \mid h_1(p) \in h_1(P) \setminus h_2(Q)\} \cup \{\{q\} \mid h_2(q) \in h_2(Q) \setminus h_1(P)\}$,
- Final weights : $\gamma(\{p,q\}) = \gamma_1(p) + \gamma_2(q); \gamma(\{p\}) = \gamma_1(p); \gamma(\{q\}) = \gamma_2(q)$.

The mapping h from R to range$(\mathbb{P}_{X\cup Y})$ as follows :
$h(\{p,q\}) = h_1(p)$; $h(\{p\}) = h_1(p)$; $h(\{q\}) = h_2(q)$.
There is no difficulty to verify that the mapping h, as defined above, is a weighted automata isomorphism which is omitted here for space limitation.

In Figure 2, we illustrate the above construction by an example of merging and determinizing of two simple PWA associated with the following two datasets $X = \{ab,ac\}, Y = \{ac,ad,e\}$

4.2 Analysis of the Prefixial Weighted Automata Merging Construction

The previous procedure, in Proposition 4.5, introduces a construction method of a PWA associated with a dataset. More interesting, it makes no assumptions about the fragments X and Y, and therefore, it provides a flexible construction algorithm of the union of two or more PWAs, either in batch or incremental way.
Moreover, this construction offers some complexity-related remarkable properties. Here, we mention some of them.
The following lemma is induced from the inclusion-exclusion principle.

Lemma 4.6. *Let X and Y be two datasets. Then:*

$$|\mathcal{P}_{X\cup Y}| \leq |\mathcal{P}_X| + |\mathcal{P}_Y| \tag{4.3}$$

Consequently, we obtain this two corollaries about the size of a PWA and the complexity of its construction.

Corollary 4.7. *Let D be a dataset. Then:*

$$|\mathcal{P}_D| \leq |D| \tag{4.4}$$

Likewise, and as the subset construction of a PWA automaton associated with the dataset $X \cup Y$ derived from the PWAs associated with X and Y is guided by the transitions of the smallest automaton, we can state the following lemma.

Lemma 4.8. *A PWA associated with the dataset $X \cup Y$ can be constructed from the PWAs associated with X and Y in $O(\min(|\mathcal{P}_X|, |\mathcal{P}_Y|))$ time.*

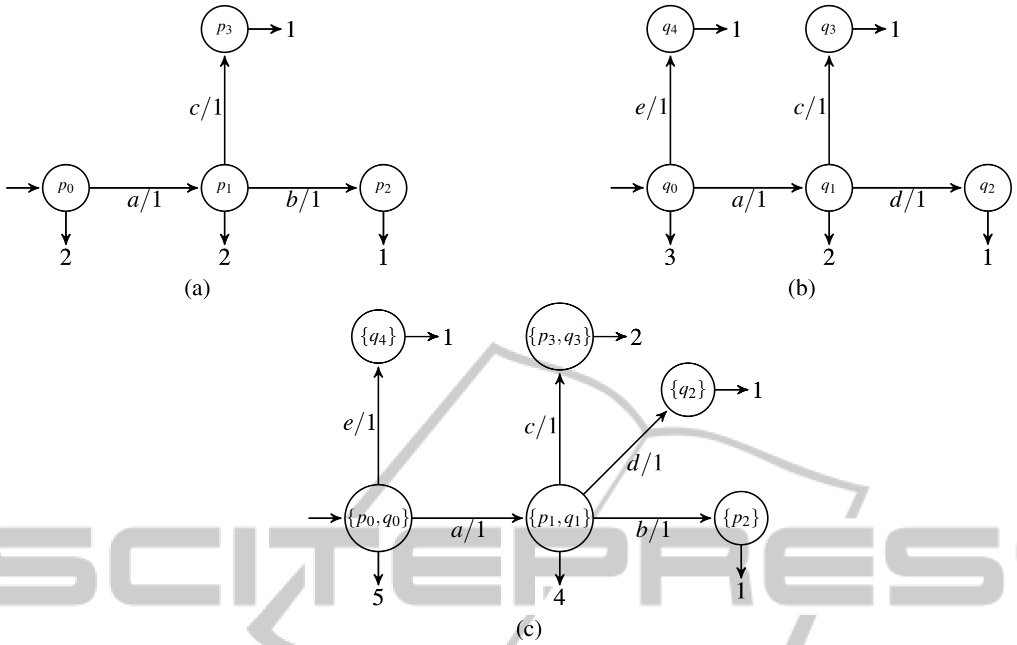


Figure 2: two automata (a) and (b) and the merging one (c) by determinization.

Consequently, we can deduce the following Proposition.

Proposition 4.9. *Let D be a dataset. A PWA of D can be constructed in $O(|D|)$ time and space complexity.*

Further, we can naturally generalize these results to k datasets. This constitutes an important criterion, that provides a fluid tuning and a flexible data partitioning scheme, which is very useful in many aspects of the problem. Indeed, it allows to deal with the memory requirements, parallelization and/or incrementality constraints, since, it does not matter, here, the granularity of this partitioning: by transaction as in (Cheung and Zaïane, 2003), or by batch as in (Totad et al., 2012), taking two or more data fragments. The following corollary gives this extension.

Corollary 4.10. *Let $\mathcal{A}_1,\ldots,\mathcal{A}_k$ be k PWA associated respectively with the datasets $X_1,\ldots,X_k$. One can construct a PWA associated with the union $X_1 \cup \ldots \cup X_k$ in $O(|X_1|+\ldots+|X_k|)$ time and space complexity.*

4.3 Toward the Itemset Weighted Automaton

The work done, so far, is a significant step toward our objective. Recall that our goal is to construct a weighted automaton that realizes the subsequence polynomial $\mathbb{S}_D$ associated with a dataset D. Let us define here another polynomial, which we refer to as the prefixial-bar polynomial. The later serves as an intermediate one, that guides us to obtain the targeted one $\mathbb{S}_D$.

Definition 4.11. *Let D be a dataset, and $\mathbb{P}_D$ the associated prefixial polynomial. The prefixial-bar polynomial $\overline{\mathbb{P}_D}$ is:*

$$\overline{\mathbb{P}_D} = \langle \mathbb{P}_D, \varepsilon \rangle + \sum_{\substack{u \in A^* \\ a \in A}} \langle \mathbb{P}_D, ua \rangle \overline{u} a \qquad (4.5)$$

Obviously, the prefixial-bar polynomial of the dataset D depends on the prefixial one. We give the following proposition.

Proposition 4.12. *Let $D = \{t_1,\ldots,t_n\}$ be a dataset. Let $\overline{\mathbb{P}_D}$ and $\mathbb{S}_D$ respectively the associated prefixial-bar and the subsequence polynomials. Then:*

$$\overline{\mathbb{P}_D} = \mathbb{S}_D \qquad (4.6)$$

Proof. Let us start by checking that the Proposition 4.12 is true for one transaction t_i taken from the dataset D of n transactions.
So, let $t_i = a_{i_1} a_{i_2} \ldots a_{i_k}$ be a k-itemset. According to the definitions in Sections 3 and 4, and the convention $\overline{a_i} = a_i + 1$, we have:

$$\begin{array}{rcl}
\mathbb{P}_{t_i} & = & 1 + a_{i_1} + a_{i_1} a_{i_2} + \ldots + a_{i_1} a_{i_2} a_{i_3} \ldots a_{i_k} \\
\text{So, } \overline{\mathbb{P}_{t_i}} & = & 1 + a_{i_1} + \overline{a_{i_1}} a_{i_2} + \ldots + \overline{a_{i_1} a_{i_2} \ldots a_{i_{k-1}}} a_{i_k} \\
 & = & \overline{a_{i_1}} + \overline{a_{i_1}} a_{i_2} + \ldots + \overline{a_{i_1} a_{i_2} a_{i_3} \ldots a_{i_{k-1}}} a_{i_k} \\
 & = & \overline{a_{i_1} a_{i_2}} + \ldots + \overline{a_{i_1} a_{i_2} a_{i_3} \ldots a_{i_{k-1}}} a_{i_k} \\
 & \ldots & \\
 & = & \overline{a_{i_1} a_{i_2} a_{i_3} \ldots a_{i_{k-1}} a_{i_k}} \\
 & = & \mathbb{S}_{t_i}
\end{array}$$

Now let us verify also the equality between the sum of the prefixial-bar polynomials and the prefixial-bar polynomial of the whole dataset D.

$$\begin{aligned}
\overline{\mathbb{P}_{t_i}} &= \langle \mathbb{P}_{t_i}, \varepsilon\rangle + \sum_{\substack{u\in A^* \\ a\in A}} \langle \mathbb{P}_{t_i}, ua\rangle \overline{u}a \\
\sum_{i=1}^{n} \overline{\mathbb{P}_{t_i}} &= \sum_{i=1}^{n} \langle \mathbb{P}_{t_i}, \varepsilon\rangle + \sum_{i=1}^{n} \sum_{\substack{u\in A^* \\ a\in A}} \langle \mathbb{P}_{t_i}, ua\rangle \overline{u}a \\
&= \sum_{i=1}^{n} \langle \mathbb{P}_{t_i}, \varepsilon\rangle + \sum_{\substack{u\in A^* \\ a\in A}} \sum_{i=1}^{n} \langle \mathbb{P}_{t_i}, ua\rangle \overline{u}a \\
&= \langle \mathbb{P}_{D}, \varepsilon\rangle + \sum_{\substack{u\in A^* \\ a\in A}} \langle \mathbb{P}_{D}, ua\rangle \overline{u}a \\
&= \overline{\mathbb{P}_{D}}
\end{aligned}$$

We've found that: $\overline{\mathbb{P}_{t_i}} = \mathbb{S}_{t_i}$, so $\sum_{i=1}^{n} \overline{\mathbb{P}_{t_i}} = \sum_{i=1}^{n} \mathbb{S}_{t_i}$ which leads to $\overline{\mathbb{P}_{D}} = \mathbb{S}_{D}$.

The construction of an automaton that compute the dataset subsequence polynomial $\mathbb{S}_D$ become now easier. Note that the polynomial $\overline{\mathbb{P}_D}$ can be rewritten to show the link with the polynomial $\mathbb{P}_D$ by adding null terms:

$$\overline{\mathbb{P}_D} = \langle \mathbb{P}_D, \varepsilon\rangle + \sum_{u\in \mathrm{range}(\mathbb{P}_D)} 0\times\overline{u} + \sum_{\substack{ua\in \mathrm{range}(\mathbb{P}_D) \\ a\in A}} \langle \mathbb{P}_D, ua\rangle \overline{u}a$$

By bringing together the expressions of $\mathbb{P}_D$ and that of $\overline{\mathbb{P}_D}$, we can note the bijection between each u in $\mathbb{P}_D$ and $\overline{u}$ in $\overline{\mathbb{P}_D}$. Consequently, since $\overline{u}$ encodes the subsequences of u (see Definition 3.1), it suffices, thus, to add ε-transitions in paths labeled u in our automaton $\mathcal{P}_D$; However, we must be scrupulous about coefficients, because adding ε-transitions may multiply the recognition paths of an itemset. This can be fixed by state duplication. That is, for each state $u \neq \varepsilon$, we create a second one ($\overline{u}a$) with the right coefficient ($\langle \mathbb{P}_D, ua\rangle$), the original becomes a non-accepting state with null weight ($0\times\overline{u}$). Notice that this state/transition duplication is, here, artificial and will be simulated as shown in Algorithm 3. This trick also insures the values of the other terms in the rest of the polynomial $\overline{\mathbb{P}_D}$. We illustrate this idea by a simple example of a dataset D containing only two transactions $D = \{abc, ab\}$. In Figure 3, we give the two automata: a PWA of D, and the extended one.

4.4 The Mining Algorithm

Once the PWA associated with D has been built using one of the processes introduced by the Proposition 4.5 or Corollary 4.10, it serves as a structure for the problem space exploration. In our mining phase, we explore the automaton using a depth-first traversal as exhibited in Algorithm 2. The main strength of our algorithm is that it doesn't require any additional memory other than that needed for the WPA as opposed to the previous approaches (see (Goethals, 2004)).

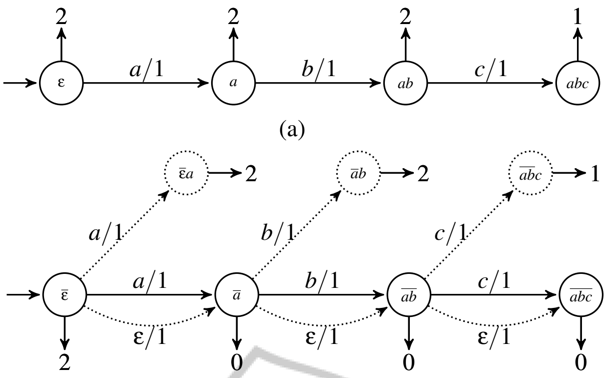
Algorithm 2: DISCOVER-FI$(\mathbb{S}, s, w, \mathcal{F})$.

```
Require: a PWA of $D$, the support-threshold $s$, a set
  of states $Q_w$, and an itemset $w$
Ensure: The set of all FI
  $\mathcal{F} \leftarrow \emptyset$                              ▷ Initialization
  for all $a > w_{|w|}$ do
      $(Q_{wa}, \langle \mathbb{S}_D, wa\rangle) \leftarrow$ EXTEND$(Q_w, w, a)$
      if $\langle \mathbb{S}_D, wa\rangle \geq s$ then
          $\mathcal{F} \leftarrow \mathcal{F} \cup \{(wa, \langle \mathbb{S}, wa\rangle)\}$
          DISCOVER-FI$(\mathbb{S}, s, wa, \mathcal{F})$
      end if
  end for
  return $\mathcal{F}$
```

Algorithm 3: EXTEND(Q_w, w, a).

```
Require: set of states $Q_w$, an itemset $w$, an item $a$
Ensure: The extended set of states $Q_{wa}$
  $P \leftarrow Q_w$
  $R \leftarrow \emptyset$
  while $P \neq \emptyset$ do
      $q \leftarrow$ pick a state from $P$
      if $i(q) = a$ then
          $R \leftarrow R \cup \{q\}$
      else if $i(q) < a$ then
          $P \leftarrow P \cup \delta^+(q)$
      end if
  end while
  return $(R, \gamma(R))$
```

The exploration begins with the invocation DISCOVER-FI$(s, \{q_0\}, \varepsilon, \mathcal{F})$, where q_0 is the initial state of the automaton. At each step, and starting from the set of states Q_w, an itemset w is extended by concatenation with its successors by calling the function EXTEND(Q_w, w, a). This call returns the set of states Q_{wa} of all paths labeled wa with their coefficients. The support of the concerned itemset wa is then the sum of the coefficients of the elements in the returned set Q_{wa}, since $\gamma(R) = \sum_{r\in R} \gamma(r)$. If this extension succeeds with a frequent itemset, the process will continue taking into account the last reached set of states Q_{wa}; Otherwise the returned couple is $(\emptyset, 0)$. Note that $i(q)$ stands for the item-label of the transition leading to the state q, with $i(q_0) = \varepsilon$, and $\delta^+(q)$ for the successor states of the state q.



(a)

(b)

Figure 3: a PWA (a) and its extended automaton (b).

Proposition 4.13. *Algorithm 2 can be done in $O(\sum_{\substack{w\in\mathcal{F}\\ a>w_{|w|}}} C_{wa})$ time and $O(|Q|)$ space, where Q is the set of states of the PWA, and $\mathcal{F}$ is the set of FI in the dataset, and C_{wa} is the time required to compute the set Q_{wa} by extending the set Q_w.*

Proof. Our automaton is acyclic over a sorted alphabet. So, the length of any path is upperbounded by $|Q|$. C_{wa} is the time needed to the call of the function EXTEND, which computes the set Q_{wa} taking into account the last obtained set of states Q_w. Let, without loss of generality, $C_{wa} = |Q_{wa}|$, hence for each itemset $w = w_1 \dots w_k$ in $\mathcal{F}$, we have: $C_{w_1} + C_{w_1 w_2} + \dots + C_{w_1 w_2 \dots w_k} < |Q|$. Consequently, if $\mathcal{F}_M$ denote the set of maximal frequent itemsets, and $\mathcal{F}_L$ the set of maximal frequent itemsets w.r.t to the prefixial relation ($\mathcal{F}_M \subseteq \mathcal{F}_L \subseteq \mathcal{F}$), we obtain the inequality : $|\mathcal{F}_M||Q| \leq \sum_{w\in\mathcal{F}} C_{wa} \leq |\mathcal{F}_L||Q| \leq |\mathcal{F}||Q| \leq |\mathcal{F}||D|$. Further, the memory requirement of the recursive exploration is also upperbounded by $|Q|$; it does not matter the length of the itemset to be recognized or the current level of the exploration, since the returned sets, during the traversal, are pairwise disjoint and their union is Q in the worst case.

5 COMPARISON AND UNIFICATION

A theoretical framework based on formal concept analysis and lattice theory is presented early in (Godin et al., 1995; Zaki and Ogihara, 1998). Recently, in (Pijls and Kosters, 2010) attempt is made to unify the common FI-algorithms w.r.t the traversal paradigms well-known in the operations research community.

Our model uses formal series, which are mappings between a monoid $\mathbb{M}$ and a semiring $\mathbb{K}$. The appropriate choice of $\mathbb{M}$ and $\mathbb{K}$, and the automaton characteristics which realizes it is driven by the targeted application and needed performances. For the basic version of the FI-mining problem, that is mining itemsets, we opted for the counting semiring $(\mathbb{N}, +, \times, 0, 1)$ because it offers an intuitive and easy implementation.

We are convinced that this framework can be generalized for mining other elaborated items such as sequences, trees or graphs, provided that much more work must be carried out to define monoids of these elements with the appropriate operations and the corresponding implementations by means of specific automata.

In what follows, we compare our model against the main state of the art techniques, and explain how these ones can be derived from it.

Level-wise Approaches. An Apriori-like algorithm (Agrawal and Srikant, 1994) proceeds level by level. First, it computes the frequent singletons and then forms from these a set of candidate doublets. After determining the frequent doublets, it continues to generate the set of frequent triplets and so on, until no new frequent itemsets can be generated. Despite its limits: generating a huge number of candidates and repetitive database scans, this algorithm stay one of the top cited algorithms in the DM community (Wu et al., 2008). Our model can be modified to fit a similar principle if we use an adapted deterministic version of the defined automaton, and perform a simple linear traversal of it in a stepwise fashion. Notice that this adaptation to Apriori allows to devise a more efficient algorithm, since in one hand any itemset have only a unique acceptation path, and in the other hand we do not make use of candidate generation neither database scans for support computation.

Vertical Approaches. The main benefit of the vertical approach (Zaki, 2000) against the level-wise approaches is speedy in the support calculation via set intersections. However, the drawback as mentioned in the introduction and by the author itself in an improved version is when the intermediate results become too big. Our method, in contrast, is based on a simple output weight read or their summation without need of any additional memory.

The vertical approach can be seen as a formal series on the powerset semiring of the set T of transactions $(2^T, \cup, \cap, \emptyset, T)$, with the min operation computed by set intersection, and the sum by the union. The weight of a transition represents the cardinality of the tidlist of the itemset formed by the path from the root $\emptyset$ to the considered node.

FPGrowth Approaches. It seems to the first glance that our defined automaton is an FPTree (Han et al., 2000) by an other way. We must emphasize at the outset that the similarity to FPTree or other concepts in any of the previous work should be seen as a positive point and not the inverse, since our purpose is the definition of a unifying model. We claim, in the other hand, that this is not correct enough. First of all, our automaton is not a data structure but rather a computational model, which can be implemented in different ways. Secondly, The mining algorithms are significantly different. While FPGrowth use a heavily intermediate memory, and also time overhead, for conditional databases and conditional FPTrees construction, our model do not require any additional memory other that necessary for the automaton. Further, and unlike FPGrowth, the open ordering adopted in our model leads to significant time improvement both in the construction phase (only one scan is required), and the mining one, since there is no need to repetitive resorting, neither database projections. Additionally, we argue that our approach outperforms also extension of FPTrees like CATSTree (Cheung and Zaïane, 2003), which the building may require many node swaps to maintain its integrity (the support of a parent must be greater than the sum of its children's supports), and incurs consequently some overhead. To the end of unification, we can view these approaches as a sequence of right derivations by the set of items A of our dataset subsequence polynomial $\mathbb{S}_D$, or like the exploration of the mirror of the automaton. Indeed, the right derivative of the polynomial $\mathbb{S}_D$ w.r.t an item a produces the polynomial representation of the a-conditional database in FPGrowth.

Tropical Semiring. An equivalent modeling to our approach can be obtained by using the tropical semiring, computed by a different weighted automaton, where the transitions carry the output weights. In this case, the output weights of all states are ∞. The weight of a path is the minimum of the weights of its transitions. It is obvious to note that this model, although equivalent, is expensive compared to which we have adopted, that consists to a simple read of the state output weight.

6 CONCLUSION

We have proposed a new model for mining FI. This model is based on formal series over the semiring $(\mathbb{N}, +, \times, 0, 1)$, whose the range constitutes the itemsets and the coefficients their supports. We argue that the strength of the introduced formalism are numerous. First, while remaining simple and intuitive, it is complete to model the basic problem. Secondly, it allows the decomposition of the problem to deal, eventually, with the constraints of time or space or both, into independent sub-problems which leads to parallelization and/or incrementatlization processes. Furthermore, the proposed model can be generalized to handle more complex items such as sequences, trees...etc. On the practical side, the model provides an implementation whose performance are proved to be competitive.

We have also, reduced this problem, in its basic version, to that of word recognition, allowing an implementation without extra memory in $O(|\mathcal{F}_L||Q|)$ time and $O(|Q|)$ space.

In future work, we can improve the mining algorithm by avoiding to recompute the extensions for itemsets u and v having the same returned set of states after the call to the function EXTEND ($Q_u = Q_v$). This can be done by working on the deterministic automaton equivalent to $\mathcal{S}_D$. We will show, in a subsequent work, that this optimization gives also a new time upperbound, which is the number of states of this deterministic automaton, which we conjecture will not be exponential. Furthermore, despite that it is not trivial, it would be very interesting to study other properties of the defined automaton such as its minimization.

We also plan to extend this approach to mine, first, the set of frequent maximal and closed itemsets, and then sequences and trees. Finally, the algebraic aspects of formal series deserves more investigation, and might lead to other theoretical or practical results.

REFERENCES

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499.

Cheung, W. and Zaïane, O. R. (2003). Incremental mining of frequent patterns without candidate generation or support constraint. In *7th International Database Engineering and Applications Symposium (IDEAS 2003), 16-18 July 2003, Hong Kong, China*, pages 111–116.

Godin, R., Missaoui, R., and Alaoui, H. (1995). Incremental concept formation algorithms based on galois (concept) lattices. *Computational Intelligence*, 11:246–267.

Goethals, B. (2004). Memory issues in frequent itemset mining. In *Proceedings of the 2004 ACM Symposium on Applied Computing (SAC), Nicosia, Cyprus, March 14-17, 2004*, pages 530–534.

Goethals, B. and Zaki, M. J., editors (2003). *FIMI '03, Frequent Itemset Mining Implementations, Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations, 19 December 2003, Melbourne, Florida, USA*, volume 90 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Han, J., Cheng, H., Xin, D., and Yan, X. (2007). Frequent pattern mining: Current status and future directions. *Data Min. Knowl. Discov.*, 15(1):55–86.

Han, J., Pei, J., and Yin, Y. (2000). Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA.*, pages 1–12.

Han, J., Pei, J., Yin, Y., and Mao, R. (2004). Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Min. Knowl. Discov.*, 8(1):53–87.

Hipp, J., Güntzer, U., and Nakhaeizadeh, G. (2000). Algorithms for association rule mining - A general survey and comparison. *SIGKDD Explorations*, 2(1):58–64.

Pasquier, N., Bastide, Y., Taouil, R., and Lakhal, L. (1999). Discovering frequent closed itemsets for association rules. In *Proceedings of the 7th International Conference on Database Theory*, ICDT '99, pages 398–416, London, UK, UK. Springer-Verlag.

Pijls, W. and Kosters, W. A. (2010). Mining frequent itemsets: a perspective from operations research. *Statistica Neerlandica*, 64(4):367–387.

Salomaa, A., Soittola, M., Bauer, F., and Gries, D. (1978). *Automata-theoretic aspects of formal power series*. Texts and monographs in computer science. Springer-Verlag.

Totad, S. G., Geeta, R. B., and Reddy, P. V. G. D. P. (2012). Batch incremental processing for fp-tree construction using fp-growth algorithm. *Knowl. Inf. Syst.*, 33(2):475–490.

Valtchev, P., Missaoui, R., and Godin, R. (2008). A framework for incremental generation of closed itemsets. *Discrete Applied Mathematics*, 156(6):924–949.

Wille, R. (1982). Restructuring lattice theory: An approach based on hierarchies of concepts. In Rival, I., editor, *Ordered Sets*, volume 83 of *NATO Advanced Study Institutes Series*, pages 445–470. Springer Netherlands.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A. F. M., Liu, B., Yu, P. S., Zhou, Z., Steinbach, M., Hand, D. J., and Steinberg, D. (2008). Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, 14(1):1–37.

Yang, Q. and Wu, X. (2006). 10 challenging problems in data mining research. *International Journal of Information Technology and Decision Making*, 5(4):597–604.

Zaki, M. (2000). Scalable algorithms for association mining. *Knowledge and Data Engineering, IEEE Transactions on*, 12(3):372–390.

Zaki, M. J. and Ogihara, M. (1998). Theoretical foundations of association rules. In *3rd ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*.

Zaki, M. J. and Wagner Meira, J. (2014). *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press.