

ACADÉMIE DE MONTPELLIER  
UNIVERSITÉ MONTPELLIER II  
— SCIENCES ET TECHNIQUES DU LANGUEDOC —

## Mémoire de Master Recherche

SPÉCIALITÉ : **Recherche en Informatique**  
MENTION : **Informatique, Mathématiques, Statistiques**

effectué au laboratoire LIRMM/INFO

—  
sous la direction de MAGUELONNE TEISSEIRE ET ANNE LAURENT

### Mining Sequential Patterns with Transversal Hypergraph Computation

par

**Haoyuan Li**

Soutenue le 21 juin 2006



# Acknowledgements

I would like to express my gratitude to all those who gave me the possibility to complete this Master thesis. I want to thank Alain Jean-Marie and Roland Ducournau, without them I couldn't be at the Université Montpellier II for my studies of Master 2 Recherche en Informatique. I have furthermore to thank Jocelyne Nanard and Stefano A. Cerri who kindly gave me many advice and encouragement.

I am deeply indebted to my directors of this Master thesis, Maguelonne Teisseire and Anne Laurent, whose help, suggestions and encouragement helped me in all the time of research for and writing of this Master thesis. I want to specially thank Prof. Pascal Poncelet for his kind and his advice in research.

I would like to thank Marc Plantevit, Federico Del Razo, Céline Fiot, Chedy Raïssi and all other colleagues at both the Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier and the Master 2 Recherche en Informatique of Université Montpellier II. I would like to specially thank Marc for helping me work on this Master thesis.

Finally, I would like to give my special thanks to my parents and my girl friend for their patient love, support and encouragement.



# Table of Contents

Acknowledgements	I
List of Figures	IV
Abstract	1
1 Introduction	3
I Data Mining and Hypergraph Transversals	7
2 Association Rules and Sequential Patterns	9
2.1 Abstraction of Database . . . . .	9
2.2 Mining Association Rules . . . . .	10
2.3 Mining Sequential Patterns . . . . .	11
3 Framework of Mannila and Toivonen	14
3.1 The Theory Extraction Formulation . . . . .	14
3.2 The Levelwise Algorithm . . . . .	15
3.3 Borders of Theories . . . . .	17
3.4 Transversal Hypergraph and Borders . . . . .	18
4 The Dualize and Advance Algorithm	21
4.1 The AMSS and All_AMSS Algorithms . . . . .	21
4.1.1 Finding A Most Specific Sentence . . . . .	21
4.1.2 Finding All Most Specific Sentences . . . . .	22
4.2 Computing Transversal Hypergraph . . . . .	23
4.3 The Complexity . . . . .	24
5 Discussion	25

<b>II</b>	<b>Discovering Sequences with Transversal Hypergraph</b>	<b>27</b>
<b>6</b>	<b>Constraints on Representing as Sets</b>	<b>29</b>
6.1	Bijection Constraint . . . . .	29
6.2	Isomorphism Constraint . . . . .	30
6.3	Powerset Constraint . . . . .	31
<b>7</b>	<b>Problems with Representing Sequences as Sets</b>	<b>33</b>
7.1	Bijjective Powerset Mapping . . . . .	33
7.2	Position Numbering . . . . .	34
<b>8</b>	<b>Ordered Patterns and Sequences</b>	<b>36</b>
8.1	Patterns and Data Mining . . . . .	36
8.2	The Ordered Patterns . . . . .	37
8.3	Finding All Interesting Ordered Patterns . . . . .	39
8.4	Representing Sequences . . . . .	41
<b>9</b>	<b>Discovering Ordered Patterns for Sequential Patterns</b>	<b>44</b>
9.1	Overview . . . . .	44
9.2	The Quality Predicate for Mining Sequential Patterns . . . . .	46
9.3	Alias Generation . . . . .	48
9.4	Evaluation Process . . . . .	48
9.5	Slicing Database . . . . .	49
9.6	The Complexity . . . . .	50
9.7	A Detailed Example . . . . .	51
	<b>Conclusions</b>	<b>57</b>
	<b>Bibliography</b>	<b>59</b>

# List of Figures

2.1	(a) A typical customer transaction database. (b) Abstraction of transaction. (c) Binary valued attributes representation. . .	10
2.2	An instance of customer transaction database with 8 transactions over 8 binary valued attributes. . . . .	11
2.3	(a) An abstraction of customer transaction database corresponding to 15 transactions of 4 customers and 6 items. (b) Customer sequences. . . . .	12
3.1	Subset lattice for $R = \{A, B, C, D\}$ . . . . .	15
3.2	Borders of a subset lattice. . . . .	18
3.3	Subset lattice generated by $R = \{A, B, C, D\}$ , $\mathcal{S} = \{ABC, D\}$ . $\mathcal{S}$ is exactly the positive border, and the negative border is the sets $\{AD, BD, CD\}$ . . . . .	20
6.1	(a) The transversal hypergraph computation can be used in a complete subsets lattice. (b) The transversal hypergraph computation cannot be used in an incomplete subsets lattice. . . . .	32
7.1	Bijjective mapping composition. . . . .	34
8.1	Patterns in data mining. . . . .	37
8.2	Sample data set for data mining tasks. . . . .	38
8.3	Transversal hypergraph computation is applicable to a lattice representing ordered patterns. . . . .	41
9.1	(a) A database $\mathcal{D}$ , it consists in 12 transactions of 4 customers and 2 items. (b) 4 slices of customer transactions. . . . .	45
9.2	The bitmap representation for a slice. . . . .	49



## **Abstract**

Most research in data mining has been addressed to the problem of mining association rules. As one of the fundamental theories of data mining, the framework of Mannila and Toivonen introduces the generalization of data mining tasks and proposes the application of transversal hypergraph computation in mining frequent itemsets. This approach has successfully applied to the association rules mining problem. On the other hand, the hypergraph theory has not been applied to the problem of mining sequential patterns because the condition of application of transversal hypergraph computation is restricted. In this Master thesis, we propose a new approach to associate transversal hypergraph computation with the problem of mining sequential patterns within the framework of Mannila and Toivonen.

We first analyze the condition of applying transversal hypergraph computation in data mining, called representing as sets, and propose three principal constraints on this problem. We then propose the ordered pattern model for representing and discovering sequences, with respect to the constraints on representing as sets. We finally show that the problem of mining sequential patterns can be transformed to the problem of discovering frequent ordered patterns, therefore the computation of transversal hypergraph can be used in mining sequential patterns.

## Résumé

Le problème de l'extraction de motifs fréquents est l'un des problèmes des plus étudiés au sein de la communauté de fouille de données. Manilla et Toivonen proposent une généralisation de ce problème en réduisant ce problème au problème de l'énumération de traverses minimales dans un hypergraphe. Même si cette généralisation s'applique avec succès à l'extraction de règles d'association, il n'existe, à notre connaissance, aucune approche utilise la théorie des traverses minimales d'hypergraphe pour l'extraction des motifs séquentiels. Dans ce mémoire de Master Recherche, nous proposons une nouvelle approche d'extraction de motifs séquentiels basée sur la théorie des traverses minimales d'hypergraphe. Nous définissons un cadre formel (représentation des séquences sous forme d'ensembles, contraintes, ...) ainsi que des algorithmes permettant la mise en œuvre de notre approche : extraire les motifs séquentiels en recherchant les traverses minimales de l'hypergraphe associé.

# Chapter 1

## Introduction

We are given a database  $\mathcal{D}$  of customer transactions where each record is a transaction that consists of transaction time, customer identification and items bought in the transaction. The problem of mining association rules [1] is stated to find rules such that “90% of transactions that purchase notebook computer and CD-R discs also purchase USB flash drive”. The antecedent of such rule consists of notebook computer and CD-R disks, and the consequent consists of USB flash drive alone. The number 90% is the *confidence* factor of the rule, and the *support* of the rule is defined to be the fraction of transactions in the database  $\mathcal{D}$  that satisfy the union of items in the antecedent and consequent of the rule.

Therefore, the problem of mining association rules could be decomposed into two subproblems, to find all frequent itemsets that have fractional transaction support above a given threshold *minimal\_support*, and to find all rules generated from these frequent itemsets that have fractional confidence above a given threshold *minimal\_confidence*. It is not difficult to find that against large databases the efficiency of mining association rules depends heavily upon the efficiency of finding all such frequent itemsets, so that most research in mining association rules has concentrated on the theories and algorithms for the discovery of frequent itemsets.

Different from mining association rules, the problem of mining sequential patterns [3] was proposed initially on purpose to discover rules like “60% customers buy notebook computer and CD-R discs, then buy USB flash drive, then buy DVD-RW discs”. The number 60% is the *support* of the rule, which is defined as the fraction of all customers in the database  $\mathcal{D}$ . Thus most algorithms for mining association rules could not be used in mining sequential patterns.

On the other hand, in [12] Mannila and Toivonen proposed a general

framework for data mining. Within this framework data mining can be viewed as following problem: given a database, a language, a quality predicate, find all *interesting* sentences from the language, means that the sentences are true by evaluating with the quality predicate in the database. Typically, this quality predicate is a frequency criterion that states that there are sufficiently many instances in the database satisfying the sentence. This framework requires that all sentences of the language follow a *specialization relation* and the quality predicate is monotone to this relation. Therefore a subset lattice could be imposed on the language by the specialization relation and be used in finding all interesting sentences. Within this framework, the computation task of finding all interesting sentences is handled by the *levelwise* algorithm [2, 12]. A large part of current research in data mining can be addressed by this framework, for example the discovery of frequent itemsets, of association rules, of episodes [14], of sequential patterns, of minimal keys, and learning monotone functions [6].

Several variations of the *levelwise* algorithm have been successfully applied to problems of data mining [2, 3, 10, 9, 13, 12, 14]. The *levelwise* algorithm computes the interesting sentences by walking up in the subset lattice of sentences, one level at a time. First, the simplest, most general, sentences from the language are evaluated against the database, and then the process continues for more specific sentences, one level at a time. This algorithm is optimal when the most specific interesting sentences are low level.

The analysis of the complexity of finding all interesting sentences uses the concept of *border*. In short words, the border is the union of the set of most specific interesting sentences (called *positive border*) and the set of most general non-interesting sentences (called *negative border*). The results in [12] show that the complexity of the *levelwise* algorithm is related to the border.

The *levelwise* algorithm and the notions of border are applicable to all data mining tasks which can be generalized to the framework of Mannila and Toivonen. The problems of mining association rules and sequential patterns can be resolved within this framework by the *levelwise* algorithm, because the monotone specialization relations can be easily defined in these two problems.

The *levelwise* algorithm can be quite slow if there are interesting sentences that are far from the bottom of the specialization hierarchy. To improve the performance [12] proposed a *guess-and-correct* algorithm. This algorithm starts the process from an initial set of sentences of the language, and then correcting the guess by looking at database. Therefore the *ver-*

*ification problem* was addressed by this proposition. If we are given a set of interesting sentences, the verification problem is equivalence to the problem of computing the negative border, since the positive border consists of the most specific sentences which are already given without computing. However it could be difficult to compute the negative border.

[12] shows that in a restricted condition, the problem of computing negative border could be reduced to the problem of computing transversal hypergraph. Given the database  $\mathcal{D}$  of customer transactions, each item is viewed as a binary valued attribute, corresponds to whether an item was bought or not. Let attributes  $R$  denote the set of all items, then each transaction can be viewed as a subset of  $R$ . Therefore, the language for finding frequent sets defines all subsets of  $R$ . If we consider  $\{\}$  in this language, then this language corresponds to the powerset of  $R$ . In this case, given the positive border of a set of interesting sentences, the negative border can be computed by transversal hypergraph computation without looking at the database.

[9] further analyzed the above problem and shows the condition of using transversal hypergraph computation in computing the negative border. This condition is called *representing as sets* and requires that the language can be represented as a powerset. Using the results in [12], [9] proposed the *Dualize and Advance* algorithm for computing all most specific sentences with transversal hypergraph computation, that has worst case running time that is subexponential to the output size. This algorithm is one of the most efficient algorithms for mining frequent itemsets [7], which has been well applied to the problem of mining association rules.

However, the transversal hypergraph computation has not been applied to the problem of mining sequential patterns, because the requirement of representing as sets is quite strong and it is not clear how to represent sequences of attributes as sets, so that the transversal hypergraph computation based algorithms, such as the *Dualize and Advance* algorithm, cannot be used in this case.

**Contributions** In this Master thesis, we first propose three constraints on representing as sets, they are *bijection*, *isomorphism* and *powerset*. We then analyze the problem with representing sequences as sets and we therefore propose the *ordered pattern* model for representing sequences. We further present that the problem of mining sequential patterns can be transformed to the problem of discovering ordered patterns with respect to the constraints on representing as sets. With the above propositions, we show that the computation of transversal hypergraph can be used in mining sequential patterns.

**Organization** The rest of this Master thesis is organized as follows. Chapter 2 introduces the model of customer transaction database we considered, and the problems of mining association rules and sequential patterns on this model. Chapter 3 introduces the framework of Mannila and Toivonen and the application of transversal hypergraph computation in data mining with the condition of representing as sets. In Chapter 4 we present the *Dualize and Advance* algorithms in theory extraction which is based on transversal hypergraph computation. Chapter 5 stands a summary of the previous three chapters and addresses the problems we considered in this Master thesis. In Chapter 6 we analyze the constraints on representing as sets and propose three principal constraints including bijection, isomorphism and powerset. In Chapter 7 we analyze the problems with representing sequences as sets and show that transversal hypergraph computation is not applicable to the language that defines sequences for mining sequential patterns. In Chapter 8 we propose the ordered pattern model which can be used in representing and discovering sequences with respect to the constraints on representing as sets. In Chapter 9 we show a framework for mining sequential patterns by discovering frequent ordered patterns and the transversal hypergraph based *Dualize and Advance* algorithm can be therefore applied. The final chapter is the conclusions and the perspectives.

Part I

Data Mining and  
Hypergraph Transversals



## Chapter 2

# Association Rules and Sequential Patterns

Many data mining problems focus on transaction databases, which can be generalized to a data set over binary-valued attributes. In this chapter, we first present this generalization of database, we then introduce the problems of mining association rules [1] and sequential patterns [3] on such generalized data set.

### 2.1 Abstraction of Database

Let us consider a typical customer transaction database, in which each transaction consists of several rows and each row corresponds to the item bought in the transaction and its quantity. Figure 2.1(a) shows two transactions of different customers in such database, where  $A, B, C$  and etc. represents an item.

We do not consider quantities of items bought in a transaction on mining association rules and sequential patterns, thus each transaction can be abstracted to a set of items. Figure 2.1(b) shows this abstraction. If we use a set  $R$  of binary-valued attributes in describing the set of items, then each transaction stands a set of attributes of  $R$  having the value “1”, which is a subset of  $R$ . Figure 2.1(c) shows this representation.

Therefore, the model of database considered in this Master thesis is a database  $\mathcal{D}$  (or data set  $\mathbf{r}$ , relation  $\mathbf{r}$  in other words) with  $n$  rows over binary valued attributes  $R$ . We denote these attributes as  $A, B, C, \dots$  or  $R_1, R_2, \dots, R_i$ , and in particular we use especially  $X, Y$  in describing subset of  $R$ .

Transaction	Customer	Item	Quantity
20060502	30876	A	2
20060502	30876	B	1
20060502	30876	D	1
20060502	30876	F	3
20060502	30876	G	1
20060502	30876	H	8
20060502	30898	A	1
20060502	30898	C	4
20060502	30898	E	1
20060502	30898	G	1
20060502	30898	H	6

(a)

Transaction	Customer	Items Bought
20060502	30876	A B D F G H
20060502	30898	A C E G H

(b)

Row	A	B	C	D	E	F	G	H
1	1	1	0	1	0	1	1	1
2	1	0	1	0	1	0	1	1

(c)

Figure 2.1: (a) A typical customer transaction database. (b) Abstraction of transaction. (c) Binary valued attributes representation.

## 2.2 Mining Association Rules

We first introduce the task of mining frequent itemsets. Given a relation  $\mathbf{r}$  with  $n$  rows over binary valued attributes  $R$  and a subset  $X \subseteq R$ , define  $t(X) = true$  if and only if row  $t \in \mathbf{r}$  has the value *true* in each column  $R_i \in X$ . The *frequency*  $fr(X)$  of  $X$  is defined as following,

$$fr(X) = \frac{|\{t \in \mathbf{r} \mid t(X) = true\}|}{n}.$$

With a user defined frequency threshold *minimal\_support*, the problem of mining frequent itemsets is to find all attribute set  $X \subseteq R$  such that  $fr(X) \geq minimal\_support$ .

**Example 1.** Given a customer transaction database  $\mathcal{D}$  with 8 transaction over 8 binary valued attributes  $R = \{A, B, C, D, E, F, G, H\}$ , shown in Figure 2.2. We compute the frequency of attribute set  $X = \{G, H\}$ .

$$fr(\{G, H\}) = \frac{|\{G, H\}|}{n} = \frac{5}{8} = 0.625$$

□

There exist various efficient algorithms and implementations for mining frequent itemsets [7].

Now we introduce the problem of mining association rules. Given a relation  $\mathbf{r}$  with  $n$  rows over binary valued attributes  $R$ , an attribute  $A \in R$  and a subset  $X \subseteq R$ , where  $A \notin X$ , an association rule is an expression of the form  $X \Rightarrow A$ . The semantics of such a rule is that if a row has *true* in all

Transaction	Customer	A	B	C	D	E	F	G	H
20060502	30876	1	1	0	1	0	1	1	1
20060502	30898	1	0	1	0	1	0	1	1
20060503	30898	0	1	0	0	1	1	1	1
20060504	41552	1	1	1	1	0	1	0	0
20060504	00677	0	1	1	0	0	1	1	1
20060504	30876	1	1	1	0	0	0	0	0
20060508	30898	1	0	0	0	1	0	0	0
20060509	41552	0	0	1	1	0	1	1	1

Figure 2.2: An instance of customer transaction database with 8 transactions over 8 binary valued attributes.

columns of attribute  $R_i \in X$  then it tends also to have *true* in column of attribute  $A$ .

Typically, association rules are searched by examining whether the attribute set  $X \cup A$  is frequent. The actual frequency is called *support* of the rule, that is,

$$support = fr(X \cup A) = \frac{|\{t \in \mathbf{r} \mid t(X \cup A) = true\}|}{n}.$$

Furthermore, the ratio of rows including *true* in  $X \cup A$  to those including *true* in the set  $X$  is called the *confidence* of the rule, that is,

$$confidence = \frac{fr(X \cup A)}{fr(X)}.$$

**Example 2.** Consider again the customer transaction database of Example 1. We compute the support and confidence of rule  $\gamma = \{G, H\} \Rightarrow F$ .

$$support_\gamma = \frac{|\{F, G, H\}|}{n} = \frac{4}{8} = 0.5$$

$$confidence_\gamma = \frac{fr(\{F, G, H\})}{fr(\{G, H\})} = \frac{4}{5} = 0.8$$

□

It is easy to see that mining frequent itemsets is a major subtask of mining association rules, and therefore the efficiency of mining frequent itemsets is the main factor of the one of mining association rules.

## 2.3 Mining Sequential Patterns

In this section we detail the problem of mining sequential patterns. We first introduce several definitions used in this problem.

TID	CID	Items Bought
776962	1	C, D, E
777873	4	A, D, E, F
778092	3	A, B, C, D, F
7783C	2	A
787055	3	D, E, F
787197	2	B, F
787245	4	A, B, C, D, F
787939	1	A
788146	4	B, C, D
788208	1	F
788247	3	D
788301	1	E, F
788346	2	A, B, D, E
788410	2	F
788449	2	A, F

CID	Customer Sequence
1	$\langle (C D E) (A) (F) (E F) \rangle$
2	$\langle (A) (B F) (A B D E) (F) (A F) \rangle$
3	$\langle (A B C D F) (D E F) (D) \rangle$
4	$\langle (A D E F) (A B C D F) (B C D) \rangle$

(a)
(b)

Figure 2.3: (a) An abstraction of customer transaction database corresponding to 15 transactions of 4 customers and 6 items. (b) Customer sequences.

An *itemset* is a non-empty set of items. A *sequence* is an ordered list of itemsets. We denote an itemset  $i$  by  $(i_1 i_2 \dots i_m)$ , where  $i_j$  is an item. We denote a sequence  $s$  by  $\langle s_1 s_2 \dots s_n \rangle$ , where  $s_j$  is an itemset. The *length* of a sequence is the number of itemsets in the sequence, a sequence of length  $k$  is called a  $k$ -sequence. A sequence  $\langle a_1, a_2, \dots, a_n \rangle$  is *contained in* another sequence  $\langle b_1, b_2, \dots, b_n \rangle$  if there exist integers  $i_1 < i_2 < \dots < i_n$  such that  $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \dots, a_n \subseteq b_{i_n}$ .

**Example 3.** For example,  $(A D E)$  and  $(C D E)$  are two itemsets, where  $A, C, D$  and  $E$  are items. The sequence  $\langle (A)(C D)(E) \rangle$  is contained in  $\langle (A C)(A D E)(A E) \rangle$ , since  $(A) \subseteq (A C)$ ,  $(C D) \subseteq (C D E)$  and  $(E) \subseteq (A E)$ .  $\square$

Figure 2.3(a) shows an abstraction of customer transaction database, where the TID field stands transaction time and the CID field stands customer identification. The field **Items Bought** corresponds to items bought in each transaction. All the transactions of a customer can together be viewed as a sequence, where each transaction corresponds to a list of items, and the list of transactions ordered by increasing transaction time, corresponds to a sequence. We call such a sequence a *customer sequence*. Figure 2.3(b) shows all customer sequences expressed by the database abstraction shown in Figure 2.3(a).

A customer *supports* a sequence  $s$  if  $s$  is contained in the customer sequence for this customer. The *support for a sequence* is defined as the fraction of total customers who support this sequence. The problem of *mining*

*sequential patterns* is therefore defined to find the maximal sequences among all sequences that have a certain user specialized *minimum support*. Each such maximal sequence represents a *sequential patterns*.

We denote  $s \sqsubseteq s_c$  if a sequence  $s$  is maximum contained in a customer sequence  $s_c$ , means that for no  $s_1 \sqsubset s_c$  we have  $s \sqsubset s_1$ . Assume there are totally  $n$  customer sequences exist in the database  $\mathcal{D}$ , the support for mining sequential patterns is then computed by following,

$$support = \frac{|\{s \mid s \sqsubseteq s_c\}|}{n}.$$

**Example 4.** Let us consider the customer sequences shown in Figure 2.3(b). The sequences  $\langle(A)(F)(F)\rangle$  has a support of 75%, since it is supported by customers 1, 2 and 3. Customer 4 does not support this sequence because  $\langle(A)(F)(F)\rangle$  is not contained in sequence  $\langle(A D E F)(A B C D F)\rangle$ . If the minimal support is defined to be 50%, then the sequence  $\langle(A)(F)(F)\rangle$  is one of the sequential patterns with  $support > 50\%$  in our customer transactions database.  $\square$

There exist many variations of mining sequential patterns [4, 15, 16, 17, 18, 19, 21].

## Chapter 3

# Framework of Mannila and Toivonen

In this chapter we introduce the framework of Mannila and Toivonen [12]. We first present the theory extraction formulation that generalizes data mining tasks to the problem of finding sentences of a language as the theory extraction. We then show the *levelwise* algorithm for finding all interesting sentences. With the notion of border, we further show how to use transversal hypergraph computation in mining frequent itemsets.

### 3.1 The Theory Extraction Formulation

The tasks of data mining can be formally described as follows [12]. Given a data set  $\mathbf{r}$ , a language  $\mathcal{L}$  for expressing properties or defining subgroups of the data, and a *quality predicate*  $q$  for evaluating whether a sentence  $\varphi \in \mathcal{L}$  defines a sufficiently large subclass of  $\mathbf{r}$ . The computational task of data mining is to find the theory of  $\mathbf{r}$  with respect to  $\mathcal{L}$  and  $q$ , that is, the set

$$\mathcal{Th}(\mathcal{L}, \mathbf{r}, q) = \{\varphi \in \mathcal{L} \mid q(\mathbf{r}, \varphi) \text{ is true}\}.$$

We use a specialization/generalization relation between sentences of  $\mathcal{L}$  in  $\mathbf{r}$ . A *specialization relation* is a partial order  $\preceq$  on the sentences in  $\mathcal{L}$ . We say that  $\varphi$  is more general than  $\theta$ , if  $\varphi \preceq \theta$ ; we also say that  $\theta$  is more specific than  $\varphi$ .

The relation  $\preceq$  is a *monotone specialization relation* with respect to  $q$  if the quality predicate  $q$  is monotone with respect to  $\preceq$ , that is, for all  $\mathbf{r}$  and  $\varphi$  we have the following: if  $q(\mathbf{r}, \varphi)$  and  $\varphi \preceq \gamma$ , then  $q(\mathbf{r}, \gamma)$ . In other words, if a sentence  $\varphi$  is interesting based on the quality predicate  $q$ , then also all more general sentences  $\gamma \preceq \varphi$  are interesting.

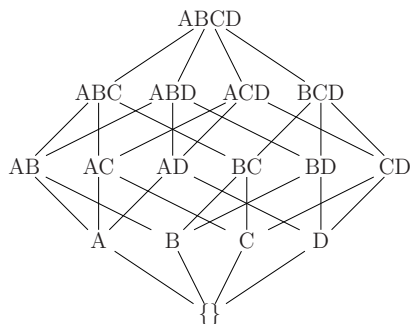


Figure 3.1: Subset lattice for  $R = \{A, B, C, D\}$ .

Obviously, if  $\mathcal{L}$  is infinite and  $q(\mathbf{r}, \varphi)$  is satisfied for infinitely many sentences,  $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$  and  $\mathcal{M}\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$  can not be computed. Thus the language  $\mathcal{L}$  must be defined carefully.

**Example 5.** We consider the problem of mining frequent itemsets. Given data set  $\mathbf{r}$  on attributes  $R$ , the language  $\mathcal{L}$  for mining frequent itemsets defines all subsets of  $R$ . For subsets  $X, Y \subset R$ , where  $X$  is defined by sentence  $\theta$  and  $Y$  is defined by sentence  $\gamma$ ,  $X \subseteq Y$  if and only if  $\theta \preceq \gamma$ .

Assume  $R = \{A, B, C, D\}$ , the partial order relation of specialization can be represented by a subset lattice shown in Figure 3.1. Within this lattice, for example, we can say that the subset  $\{A, B, C\}$  is more specific than the subset  $\{A, B\}$  and the subset  $\{A, B\}$  is more general than the subset  $\{A, B, C\}$ . Let  $\theta = \{A, B\}$  and  $\gamma = \{A, B, C\}$ , we have  $\theta \preceq \gamma$ . If  $q(\mathbf{r}, \gamma)$  is satisfied, that is, the set  $\{A, B, C\}$  is frequent, then  $q(\mathbf{r}, \theta)$  must be satisfied, that is, the set  $\{A, B\}$  must be frequent. Concretely, if  $\{A, B\}$  is not frequent, means that  $q(\mathbf{r}, \theta)$  can not be satisfied, then the set  $\{A, B, C\}$  cannot be frequent, thus  $q(\mathbf{r}, \gamma)$  cannot be satisfied.  $\square$

## 3.2 The Levelwise Algorithm

The *levelwise (a priori)* algorithm [2, 12] computes the theory  $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$  by level.

Algorithm 1 shows the *levelwise* algorithm, it solves the problem of theory extraction by finding all interesting sentences, the whole theory  $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ , going bottom up. It proceeds by first computing the  $\mathcal{T}h_0$  consisting of the most general sentences that are in  $\mathcal{T}h$ . Then, assuming  $\mathcal{T}h_i$  is known, it computes a set of *candidates*: sentences  $\psi$  with level  $i + 1$  such that all  $\theta$  with  $\theta \prec \psi$  are in  $\mathcal{T}h$ . For each one of these candidates  $\psi$ , the

---

**Algorithm 1:** The *levelwise* algorithm for finding all interesting sentences.

---

**Input:** A data set  $\mathbf{r}$ , a language  $\mathcal{L}$  with specialization relation  $\preceq$ .

**Output:** The set  $Th(\mathcal{L}, \mathbf{r}, q)$ .

```

1  $C_1 \leftarrow \{\varphi \in \mathcal{L} \mid \text{there is no } \gamma \in \mathcal{L} \text{ such that } \gamma \prec \varphi\};$ 
2  $i \leftarrow 1;$ 
3 while  $C_i \neq \emptyset$  do
4    $F_i \leftarrow \{\varphi \in C_i \mid q(\mathbf{r}, \varphi)\};$ 
5    $C_{i+1} \leftarrow \{\varphi \in \mathcal{L} \mid \text{for all } \gamma \prec \varphi \text{ we have } \gamma \in \bigcup_{j \leq i} F_j\} \setminus \bigcup_{j \leq i} C_j;$ 
6    $i \leftarrow i + 1;$ 
7 end
8 output  $\bigcup_{j \leq i} F_j;$ 

```

---

algorithm calls the function  $q$  to check whether  $\psi$  really belongs to  $Th$ . This iterative procedure is performed until no more sentences in  $Th$  are found.

**Theorem 1.** *The levelwise algorithm computes the set of interesting sentences correctly, and it evaluates the predicate  $q$*

$$|Th(\mathcal{L}, \mathbf{r}, q) \cup Bd^-(Th(\mathcal{L}, \mathbf{r}, q))|$$

*times.* [12]

**Example 6.** Let us consider again Example 5. Given data set  $\mathbf{r}$  on attributes  $R = \{A, B, C, D\}$ , Figure 3.1 shows a subset lattice of the language  $\mathcal{L}$  defining all subsets on  $R$ . Let us consider the problem of finding frequent sets on  $\mathbf{r}$ . Assume that the collection  $Th$  of  $R$  is

$$Th = \{\{A\}, \{B\}, \{C\}, \{D\}, \{A, B\}, \{A, C\}, \{B, C\}, \{A, B, C\}\}.$$

Algorithm 1 (*levelwise*) starts with  $C_1 = \{\{A\}, \{B\}, \{C\}, \{D\}\}$ . In the first iteration  $C_1$  is evaluated in Step 4 and  $F_1 = \{\{A\}, \{B\}, \{C\}, \{D\}\}$  is found as frequent. In Step 5 the algorithm generates new candidate sentences  $C_2 = \{\{A, B\}, \{A, C\}, \{A, D\}, \{B, C\}, \{B, D\}, \{C, D\}\}$ . In the next iteration new frequent sentences  $F_2 = \{\{A, B\}, \{A, C\}, \{B, C\}\}$  are found in Step 4, and in Step 5 new candidate sentences  $C_3 = \{\{A, B, C\}\}$  is generated. In final iteration the algorithm finds  $F_3 = \{\{A, B, C\}\}$  is frequent. The result is exactly

$$\begin{aligned} Th &= F_1 \cup F_2 \cup F_3 \\ &= \{\{A\}, \{B\}, \{C\}, \{D\}, \{A, B\}, \{A, C\}, \{B, C\}, \{A, B, C\}\}. \end{aligned}$$

□

The *levelwise* algorithm has been used in various forms in finding association rules, episodes, sequential patterns, etc. [2, 3, 9, 10, 12, 13, 14].

### 3.3 Borders of Theories

Consider a set  $\mathcal{S}$  of sentences from  $\mathcal{L}$  such that  $\mathcal{S}$  is closed downwards under the relation  $\preceq$ . That is, if  $\varphi \in \mathcal{S}$  and  $\gamma \preceq \varphi$ , then  $\gamma \in \mathcal{S}$ . The *border*  $\mathcal{Bd}(\mathcal{S})$  of  $\mathcal{S}$  consists of those sentences  $\varphi$  such that all generalization of  $\varphi$  are in  $\mathcal{S}$  and none of the specialization of  $\varphi$  is in  $\mathcal{S}$ . Those sentences  $\varphi$  in  $\mathcal{Bd}(\mathcal{S})$  that are in  $\mathcal{S}$  are called *positive border*  $\mathcal{Bd}^+(\mathcal{S})$ , and those sentences  $\varphi$  in  $\mathcal{Bd}(\mathcal{S})$  that are not in  $\mathcal{S}$  are called *negative border*  $\mathcal{Bd}^-(\mathcal{S})$ . In other words, the positive border consists of the most specific sentences in  $\mathcal{S}$ , the negative border consists of the most general sentences that are not in  $\mathcal{S}$ :

$$\mathcal{Bd}(\mathcal{S}) = \mathcal{Bd}^+(\mathcal{S}) \cup \mathcal{Bd}^-(\mathcal{S}),$$

where

$$\mathcal{Bd}^+(\mathcal{S}) = \{\varphi \in \mathcal{S} \mid \text{for all } \theta \in \mathcal{L} \text{ with } \varphi \prec \theta, \text{ we have } \theta \notin \mathcal{S}\}$$

and

$$\mathcal{Bd}^-(\mathcal{S}) = \{\varphi \in \mathcal{L} \setminus \mathcal{S} \mid \text{for all } \gamma \in \mathcal{L} \text{ with } \gamma \prec \varphi, \text{ we have } \gamma \in \mathcal{S}\}.$$

**Example 7.** Figure 3.2 illustrates the notion of border. Given a language  $\mathcal{L}$  for defining subset lattice, for instance, generated from a set of attributes  $R = \{A, B, C, D, E, F, G, H\}$ , each sentence stands a subset of  $R$ . The positive consists of a set of the most specific interesting sentences and the negative border consists the most general non-interested sentences. If we use a frequency threshold in addressing the interestingness, then all subsets inside the border, from the vision of specialization, can be frequent but the ones outside the border cannot be frequent.  $\square$

The main effort in finding interesting sentences is in the step where the interestingness of subgroups is evaluated against the database. Thus we consider the following model of computation. Assume the only way of getting information from the database is by asking questions of the form

*Is-interesting.* Is the sentence  $\varphi$  interesting, that is, does  $q(\mathbf{r}, \varphi)$  holds?

**Theorem 2.** Any algorithm for computing  $\mathcal{Th}(\mathcal{L}, \mathbf{r}, q)$  must use at least  $|\mathcal{Th}(\mathcal{L}, \mathbf{r}, q) \cup \mathcal{Bd}^-(\mathcal{Th}(\mathcal{L}, \mathbf{r}, q))|$  evaluations of the quality predicate  $q$ . [12]

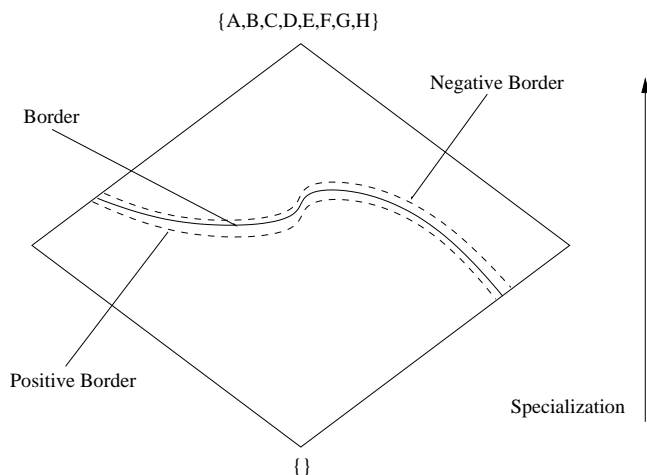


Figure 3.2: Borders of a subset lattice.

Given a specialization relation  $\preceq$ , the set  $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$  can be represented by enumerating only its maximal elements [9], i.e., the set

$$\mathcal{M}Th(\mathcal{L}, \mathbf{r}, q) = \{\varphi \in \mathcal{T}h(\mathcal{L}, \mathbf{r}, q) \mid \text{for no } \theta \in \mathcal{T}h(\mathcal{L}, \mathbf{r}, q), \varphi \prec \theta\},$$

which is equivalence to the positive border of  $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ .

Therefore, data mining tasks can be generalized to the problem of finding  $\mathcal{M}Th(\mathcal{L}, \mathbf{r}, q)$ , that is, the problem of finding the positive border of theory  $\mathcal{B}d^+(\mathcal{T}h(\mathcal{L}, \mathbf{r}, q))$ .

### 3.4 Transversal Hypergraph and Borders

A *hypergraph*  $\mathcal{H} = (V, E)$  consists of a finite collection  $E$  of sets over a finite set  $V$ . The elements of  $E$  are called the *hyperedges*, or simply *edges* of the hypergraph. A *transversal* of  $\mathcal{H}$  is a set  $T \subseteq V$  that intersects all the edges of  $E$ . A transversal is *minimal*, if no  $T' \subset T$  is a transversal. The set  $\mathcal{T}r(\mathcal{H})$  of all minimal transversals of a hypergraph  $\mathcal{H}$  is itself a hypergraph called the *transversal hypergraph*, of  $\mathcal{H}$ .

Generating and computing minimal transversals of a hypergraph (or transversal hypergraph) is an important problem which has many applications in data mining and other AI domains [6, 8, 9, 10, 12, 20].

Given a class of sentences  $\mathcal{S} \subseteq \mathcal{L}$  closed downwards under the specialization relation  $\preceq$ , the problem of computing the negative border  $\mathcal{B}d^-(\mathcal{S})$  can be reduced to transversal hypergraph computation [12], if the language  $\mathcal{L}$  satisfied the following condition of *representing as sets*.

**Definition 1** (Representing as Sets). *Let  $\mathcal{L}$  be the language,  $\preceq$  a specialization relation, and  $\mathcal{R}$  a set; denote by  $\mathcal{P}(\mathcal{R})$  the powerset of  $R$ . A function  $f : \mathcal{L} \rightarrow \mathcal{P}(\mathcal{R})$  is a representation of  $\mathcal{L}$  (and  $\preceq$ ) as sets, if  $f$  is one-to-one and surjective,  $f$  and its inverse are computable, and for all  $\theta$  and  $\varphi$  we have  $\varphi \preceq \theta$  if and only if  $f(\varphi) \subseteq f(\theta)$ . This transformation is called representing as sets. [9]*

The representing as sets requires that the structure imposed on  $\mathcal{L}$  by  $\preceq$  is isomorphic to a finite subset lattice on the relation  $\preceq$ , and its size must be a power of 2. The mapping  $f$  must be invertible and must cover all of  $\mathcal{P}(\mathcal{R})$ .

For the problem of mining frequent itemsets, we have already  $\mathcal{L}$  represented as sets. Given data set  $\mathbf{r}$  on attributes  $R$ , let the language  $\mathcal{L}$  defines all sentences for mining frequent itemsets. The language  $\mathcal{L}$  therefore satisfies the requirement of representing as sets, since  $\mathcal{L}$  defines all subsets of  $R$ , and that is the powerset of  $R$  if we consider  $\{\} \in \mathcal{L}$ . In this case the identity mapping  $f(X) = X$  can be used in representing  $\mathcal{L}$  as a powerset.

We further detail the constraints on representing as sets in Chapter 6, however it is not clear how to represent the language for mining sequential patterns as sets.

We now show how the computation of transversal hypergraph can be used in computing the negative border in mining frequent itemsets.

Given data set  $\mathbf{r}$  on attributes  $R$ , let  $\mathcal{S} \subseteq \mathcal{L}$  be a class of sentences closed downwards under the specialization relation  $\preceq$ , where the language  $\mathcal{L}$  defines all sentences for mining frequent itemsets.

Consider the hypergraph  $\mathcal{H}(\mathcal{S})$  on  $R$ , that contains the complements of sets  $f(\varphi)$  for  $\varphi \in \mathcal{B}d^+(\mathcal{S})$  as edges:

$$\mathcal{H}(\mathcal{S}) = \{R \setminus f(\varphi) \mid \varphi \in \mathcal{B}d^+(\mathcal{S})\}.$$

Then  $Tr(\mathcal{H}(\mathcal{S}))$  is a hypergraph on  $R$ , and hence we can apply  $f^{-1}$  to it:

$$f^{-1}(Tr(\mathcal{H}(\mathcal{S}))) = \{f^{-1}(H) \mid H \in Tr(\mathcal{H}(\mathcal{S}))\}.$$

We have the following.

**Theorem 3.**  $f^{-1}(Tr(\mathcal{H}(\mathcal{S}))) = \mathcal{B}d^-(\mathcal{S})$ . [12]

**Example 8.** We consider again the relation  $R = \{A, B, C, D\}$  illustrated in Figure 3.3. Let  $\mathcal{B}d^+(\mathcal{S}) = \{ABC, D\}$ , where we use a shorthand notation for sets, for example, we represent  $\{A, B, C\}$  by  $ABC$ . Then, the

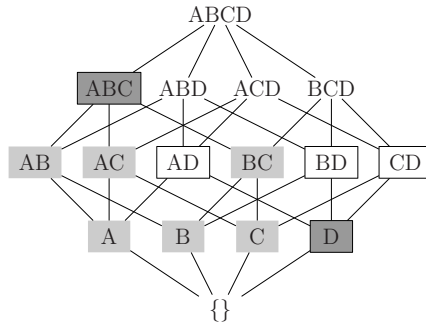


Figure 3.3: Subset lattice generated by  $R = \{A, B, C, D\}$ ,  $\mathcal{S} = \{ABC, D\}$ .  $\mathcal{S}$  is exactly the positive border, and the negative border is the sets  $\{AD, BD, CD\}$ .

downward closure of  $\mathcal{S}$  is equal to  $\{ABC, AB, AC, BC, A, B, C, D\}$ , and  $\mathcal{S}$  includes the maximal elements. It is not difficult to find that the negative border (that can be found by drawing the corresponding lattice) is  $\mathcal{Bd}^-(\mathcal{S}) = \{AD, BD, CD\}$ .

For this problem we have already  $\mathcal{L}$  represented as sets and we use the identity mapping  $f(X) = X$ . We have  $\mathcal{H}(\mathcal{S}) = \{D, ABC\}$ . It is easy to see that  $\mathcal{Tr}(D, ABC) = \{AD, BD, CD\}$ , and thus  $f^{-1}$  is exactly the negative border  $\mathcal{Bd}^-(\mathcal{S})$ .  $\square$

In next chapter we introduce the {Dualize and Advance} algorithm [9] that uses transversal hypergraph computation in computing  $\mathcal{MTh}(\mathcal{L}, \mathbf{r}, q)$ . This algorithm is one of the most efficient algorithms for mining frequent itemsets.

## Chapter 4

# The Dualize and Advance Algorithm

In this chapter we introduce the *Dualize and Advance* algorithm [9] and the algorithm associated for computing transversal hypergraph [9, 11].

### 4.1 The AMSS and All\_AMSS Algorithms

The *Dualize and Advance* algorithm [9] is efficient in finding frequent itemsets by computing transversal hypergraph. The algorithm includes two routines, the *AMSS* (A Most Specific Sentence) algorithm and the *All\_AMSS* (All Most Specific Sentences) algorithm.

#### 4.1.1 Finding A Most Specific Sentence

Given data set  $\mathbf{r}$  on attributes  $R$ , a language  $\mathcal{L}$  for mining frequent itemsets, and an interesting sentence  $\varphi \in \mathcal{L}$ , Algorithm 2 (*AMSS*) finds a maximal interesting sentence  $\theta \in \mathcal{L}$ , such that  $\varphi \preceq \theta$ .

---

**Algorithm 2:** The *AMSS* algorithm.

---

```
1  $R_i \leftarrow \text{permutation}(R, \varphi);$ 
2 while  $R_i \neq \emptyset$  do
3   if  $q(\mathbf{r}, \varphi \cup R_i) = \text{true}$  then
4      $\varphi \leftarrow \varphi \cup R_i;$ 
5   end
6    $R_i \leftarrow \text{permutation}(R, \varphi);$ 
7 end
8 return  $\varphi;$ 
```

---

Once a set of most specific sentences is found the algorithm focuses its search by computing the negative border of the sentences found (using a transversal hypergraph computation), and starting its upward search from this negative border. If progress can be made, it can be made from the negative border and thus the approach is guaranteed to succeed.

The algorithm *AMSS* has to consider all attributes sequentially but the actual order does not matter. The use of permutations is an interesting heuristic that can allow more efficient discovery of new maximal interesting sentences. We use a further extendable function  $permutation(R, \varphi)$  for generating the next attribute  $R_i$ .

#### 4.1.2 Finding All Most Specific Sentences

Given data set  $\mathbf{r}$  on attributes  $R$ , a language  $\mathcal{L}$  for mining frequent itemsets, Algorithm 3 (*AllAMSS*) computes all most specific interesting sentences.

---

**Algorithm 3:** The *AllAMSS* algorithm.

---

```

1  $i \leftarrow 1$ ;
2  $\mathcal{C} \leftarrow \{\}$ ;
3 while true do
4    $\delta \leftarrow \emptyset$ ;
5    $\overline{D}_i \leftarrow \{R \setminus C \mid C \in \mathcal{C}\}$ ;
6    $\mathcal{T}_i \leftarrow \text{minimal\_transversals}(\overline{D}_i)$ ;
7   foreach  $\tau \in \mathcal{T}_i$  do
8     if  $q(\mathbf{r}, \tau) = \text{true}$  then
9        $\delta \leftarrow \tau$ ;
10      break ;
11    end
12  end
13  if  $\delta = \emptyset$  then
14    return  $\mathcal{C}$ ;
15  end
16   $\psi \leftarrow \text{amss}(\mathbf{r}, \delta)$ ;
17   $\mathcal{C} \leftarrow \mathcal{C} \cup \{\psi\}$ ;
18   $i \leftarrow i + 1$ ;
19 end

```

---

It is useful to notice that the transversal hypergraph computation does not look at the data, only at elements of  $\mathcal{L}$ ; if the input data is large, a complicated computation on  $\mathcal{L}$  can still be much cheaper than just reading

the data once.

## 4.2 Computing Transversal Hypergraph

The efficiency of the *Dualize and Advance* algorithm depends on the algorithm for computing the transversal hypergraph. Based on the algorithm introduced in [5], a heuristic algorithm was proposed in [11] for generating all transversals of a hypergraph. In this algorithm, a tree structure is used in traversing the set of transversals of hypergraph. Using this tree structure scheme, [9] presents an algorithm for computing the transversals of the hypergraph incrementally at each step of the *Dualize and Advance* algorithm. We use this algorithm in computing transversal hypergraph of the *ALLAMSS* algorithm.

Algorithm 4 corresponds to the function  $minimal\_transversals(\overline{D_i})$  in the *ALLAMSS* algorithm, it starts with  $i \geq 2$  and returns all minimal transversals of step  $i$ . The case of  $i = 1$  is quite simple, the transversal hypergraph is equivalence to  $\overline{D_1}$ .

---

**Algorithm 4:** Computing the transversal hypergraph incrementally within the *ALLAMSS* algorithm.

---

```

1  $\mathcal{T}_i \leftarrow \{\}$ ;
2 foreach  $\tau \in \mathcal{T}_{i-1}$  do
3    $\mathcal{X}' \leftarrow expand(\tau, \overline{D_i})$ ;
4   foreach  $\pi \in \mathcal{X}'$  do
5     if  $dop\_q(s, \pi) \neq \mathbf{true}$  then
6        $remove\_sentence(\mathcal{X}', \pi)$ ;
7     end
8   end
9    $\mathcal{T}_i \leftarrow \mathcal{T}_i \cup \mathcal{X}'$ ;
10 end
11 return  $\mathcal{T}_i$ ;

```

---

Consider two consecutive steps of the algorithm *ALLAMSS*, the  $(i)$ th and the  $(i + 1)$ th where  $i \geq 2$ . If some new maximal frequent sets of ordered patterns were found during step  $i$ , then  $\overline{D_i} \subset \overline{D_{i+1}}$ . Let  $X$  be a transversal of  $\overline{D_i}$ , then either  $X$  is a transversal of  $\overline{D_{i+1}}$  as well, or it can become a transversal of  $\overline{D_{i+1}}$  if it is expended by a set of items that cover  $\overline{D_{i+1}} \setminus \overline{D_i}$ . Each such transversal can be expended to a set of transversals of the set  $\overline{D_{i+1}}$ .

Therefore, if we have the transversal of  $\overline{D_i}$ , we can incrementally create the transversals of  $\overline{D_{i+1}}$  by expanding each of the original transversals so that they intersect each of the new complements of maximal elements. In addition, if a transversal  $X$  is found to be not frequent at step  $i$ , then we do not expand this transversal in the next step. All the transversals we can generate from it have to be non-frequent as well. This is an important improvement because it allows us to reduce the number of transversals we have to generate at each step.

### 4.3 The Complexity

As shown in [9], the time complexity of the *Dualize and Advance* algorithm depends on the complexity of transversal hypergraph computation.

**Theorem 4.** *If there is an incremental  $T(I, i)$  time algorithm for computing transversal hypergraph then  $\mathcal{MTh} = \mathcal{MTh}(\mathcal{L}, \mathbf{r}, q)$  can be computed in time polynomial in  $|\mathcal{MTh}|$  and  $T(|\mathcal{MTh}|, |\mathcal{Bd}^-(\mathcal{MTh})|)$ , while using at most  $(|\mathcal{Bd}^-(\mathcal{MTh})| + \text{width}(\mathcal{L}, \preceq)|\mathcal{MTh}|)$  queries. [9]*

The time complexity and query complexity of the algorithm are separated. Theorem 4 shows that, while the running time depends on the transversal hypergraph computation and may not be polynomial, only a polynomial number of queries is required.

[8] presented an incremental algorithm for the transversal hypergraph computation with time complexity  $T(I, i) = (I + i)^{O(\log(I+i))}$ , we can therefore conclude that the time complexity of the *Dualize and Advance* algorithm could be

$$\mathcal{O} = (|\mathcal{MTh}| + |\mathcal{Bd}^-(\mathcal{MTh})|)^{\mathcal{O}(\log(|\mathcal{MTh}| + |\mathcal{Bd}^-(\mathcal{MTh})|))}.$$

## Chapter 5

# Discussion

In previous chapters, we compare the mining association rules and mining sequential patterns problems with an abstraction of customer transaction database. With the comparison, it is not difficult to see that the two problems are two different visions to the same database generation. However, more search work has been applied to the problem of mining association rules in both theories and algorithms.

As one of the most important fundamental theories in data mining, the framework of Mannila and Toivonen [12] generalizes data mining tasks to a theory extraction formulation, and with this formulation the *levelwise* is used in computing generalized data mining tasks. This generalization system can be applied to both the problem of mining association rules and the problem of mining sequential patterns, since between elements of the database abstraction of these two problems the specialization relation is respected.

The framework of Mannila and Toivonen also associates the mining frequent itemsets problem with transversal hypergraph computation, and in [9] Gunopulos et al. further proposed the transversal hypergraph computation based *Dualize and Advance* algorithm that finds the most specific elements in polynomial time and query complexity. These approaches have been successfully applied to improve the association rules mining theory.

To the best of our knowledge, the transversal hypergraph computation has not been applied to the problem of mining sequential patterns within the framework of Mannila and Toivonen. The requirement of representing as sets is strong in this application however the representation of sequences defined in mining sequential patterns does not satisfy this requirement.

In next part of this Master thesis, we analyze the problem of representing sequences and propose a model for using transversal hypergraph computa-

tion in mining sequential patterns.

## Part II

# Discovering Sequences with Transversal Hypergraph



## Chapter 6

# Constraints on Representing as Sets

As introduced in [9], the problem of representing as sets restricts the applications of transversal hypergraph computation in data mining tasks, so that such algorithms like *Dualize and Advance* could not be used in mining sequential patterns. However, [9] did not further detail this problem.

In this chapter, we analyze and detail the constraints on representing as sets, our propositions in next chapters are based on this analysis.

### 6.1 Bijection Constraint

The problem of representing as sets is to find an invertible mapping function  $f$ , with which we have the structure imposed on the language  $\mathcal{L}$  by  $\preceq$  being isomorphic to a powerset  $\mathcal{P}(\mathcal{R})$ , where  $\mathcal{R}$  is a finite set. Thus, to represent a language  $\mathcal{L}$  as sets, the mapping function  $f$  must be bijective so that the inverse function  $f^{-1}$  exists and all of  $\mathcal{P}(\mathcal{R})$  are covered.

**Constraint 1** (Bijection). *Mapping function  $f$  must be bijective.*

With the definition of bijection, the Constraint 1 can be formulated as

$$\varphi \in \mathcal{L} \iff f(\varphi) \in \mathcal{P}(\mathcal{R})$$

and

$$X_\varphi \in \mathcal{P}(\mathcal{R}) \iff f^{-1}(X_\varphi) \in \mathcal{L}.$$

**Example 9.** Given attributes  $R$ , the description language  $\mathcal{L}$  for defining all subsets of  $R$  is the powerset of  $R$ , denote by  $\mathcal{P}(R)$ , and we have immediately  $\mathcal{P}(\mathcal{R}) = \mathcal{P}(R)$ . Therefore the mapping function  $f$  for mining frequent sets

such that  $f : \mathcal{L} \rightarrow \mathcal{P}(R)$  can be an identity mapping  $f(X) = X$  and we have  $f^{-1}(X) = X$ .  $\square$

If a mapping function  $f$  does not respect the Constraint 1, i.e., the inverse function  $f^{-1}$  does not exist, after computing the transversal hypergraph, a set  $X_\varphi \in \mathcal{P}(\mathcal{R})$  may not have an inverse mapping to be applied in the transformation from  $X_\varphi \in \mathcal{P}(\mathcal{R})$  to the language  $\varphi \in \mathcal{L}$ .

In particular, for attribute sets we have  $\mathcal{L} \equiv \mathcal{P}(R)$ , thus for all  $f(\varphi) = X_\varphi$  we have  $\varphi = X_\varphi$  and therefore we can simply write the identity mapping as  $f(X) = X$ .

## 6.2 Isomorphism Constraint

The goal of representing as sets is that we can apply the application of transversal hypergraph computation to the powerset  $\mathcal{P}(\mathcal{R})$  and then we can transfer the results from the powerset  $\mathcal{P}(\mathcal{R})$  to the description language  $\mathcal{L}$  via the inverse function  $f^{-1}$ . This transformation is based on the isomorphism on the specialization relation  $\preceq$  over the mapping function  $f$ .

**Constraint 2** (Isomorphism). *Mapping function  $f$  must be isomorphic to the specialization relation  $\preceq$ .*

The mapping function for representing as sets transfers the language  $\mathcal{L}$  to a powerset  $\mathcal{P}(\mathcal{R})$  and then the transversal hypergraph computation on  $\mathcal{P}(\mathcal{R})$  can be used to reduce the complexity of computing the negative border of the theory  $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ . The Constraint 2 requires  $f$  is monotone with respect to the specialization relation  $\preceq$ , that is,

$$\varphi \preceq \theta \iff f(\varphi) \preceq f(\theta).$$

**Example 10.** The identity mapping function  $f(X) = (X)$  is isomorphic on  $\preceq$  to attribute sets. Let us consider two sentences  $\varphi = \{A, C\}$  and  $\theta = \{A, B, C, D\}$  in the description language for given attributes  $R = \{A, B, C, D\}$ . After applying the identity mapping, we have

$$f(\varphi) = \{A, C\}$$

and

$$f(\theta) = \{A, B, C, D\}.$$

Clearly that we have therefore

$$\varphi \preceq \theta \iff f(\varphi) \preceq f(\theta).$$

But the identity mapping is not isomorphic on  $\preceq$  to sequences. Consider two sequences  $\langle\langle AC \rangle\rangle$  and  $\langle\langle AB \rangle\rangle \langle\langle CD \rangle\rangle$ . With the identity mapping we have again

$$f(\langle\langle AC \rangle\rangle) = \{A, C\}$$

and

$$f(\langle\langle AB \rangle\rangle \langle\langle CD \rangle\rangle) = \{A, B, C, D\},$$

but we do not have

$$\langle\langle AC \rangle\rangle \subseteq \langle\langle AB \rangle\rangle \langle\langle CD \rangle\rangle$$

by the inverse mapping on relation

$$\{A, C\} \subseteq \{A, B, C, D\}.$$

□

The Constraint 2 is an important issue for considering the mapping function for representing sequences as sets.

### 6.3 Powerset Constraint

The application of transversal hypergraph computation requires a complementary set of  $\mathcal{R} \setminus f(\varphi)$  to be edges of the hypergraph,  $f(\varphi) \in \mathcal{P}(\mathcal{R})$  therefore satisfies this requirement.

**Constraint 3** (Powerset). *Mapping function  $f$  must return a powerset.*

Recall the hypergraph  $\mathcal{H}(\mathcal{S})$  defined on  $\mathcal{R}$ ,

$$\mathcal{H}(\mathcal{S}) = \{\mathcal{R} \setminus f(\varphi) \mid \varphi \in \mathcal{B}d^+(\mathcal{S})\}.$$

$\mathcal{R} \setminus f(\varphi)$  exists if and only if  $f(\varphi) \in \mathcal{P}(\mathcal{R})$ . The size of  $\mathcal{P}(\mathcal{R})$  is a power of 2, that is

$$|\mathcal{P}(\mathcal{R})| = 2^{|\mathcal{R}|}.$$

**Example 11.** For depicting the requirement of the Constraint 3, we compute the negative border for a given  $\mathcal{S}$  on attributes  $R = \{A, B, C, D\}$  by using the transversal hypergraph computation. Assume

$$\mathcal{S} = \{A, B, C, D, AB, BC, BD, CD, BCD\},$$

without difficulty we have

$$\mathcal{B}d^+(\mathcal{S}) = \{AB, BCD\},$$

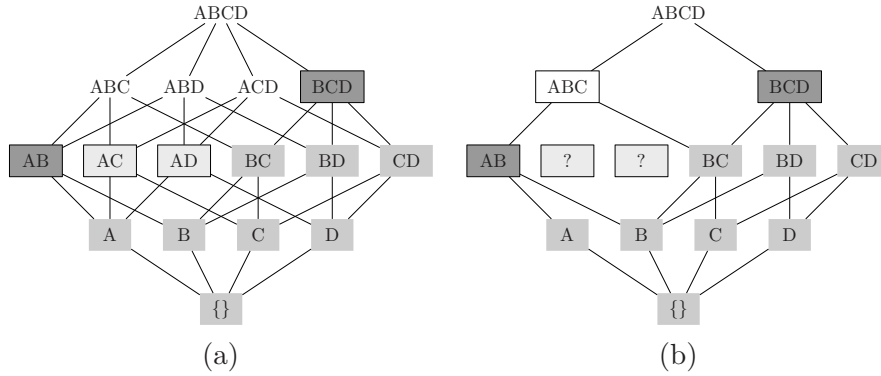


Figure 6.1: (a) The transversal hypergraph computation can be used in a complete subsets lattice. (b) The transversal hypergraph computation cannot be used in an incomplete subsets lattice.

the hypergraph for computing  $\mathcal{B}d^-(\mathcal{S})$  is therefore

$$\mathcal{H}(\mathcal{S}) = \{CD, A\},$$

thus we have the minimal transversals

$$\mathcal{T}r(\mathcal{H}(\mathcal{S})) = \{AC, AD\}.$$

Figure 6.1(a) shows this application of transversal hypergraph computation, where the subsets lattice represents the powerset of  $\mathcal{R}$  and we have  $\mathcal{R} = R$ .

Now let us consider the case of an incomplete subsets lattice, i.e., non powerset. Shown in Figure 6.1(b),  $AC$  and  $AD$  do not exist, clearly that in this case  $\mathcal{B}d^-(\mathcal{S}) \neq \mathcal{T}r(\mathcal{H}(\mathcal{S}))$ . The negative border in this case is in fact

$$\mathcal{B}d^-(\mathcal{S}) = \{ABC\}.$$

□

## Chapter 7

# Problems with Representing Sequences as Sets

In this chapter we analyze the problems with representing sequences as sets and show that sequences cannot be represented as sets, thus the transversal hypergraph computation is not applicable to the language that defines sequences for mining sequential patterns.

### 7.1 Bijective Powerset Mapping

For discovering sequences based patterns, the task of representing as sets is to find such a mapping function that it maps a description language  $\mathcal{L}_{seq}$  for defining sequences to a powerset. With respect to the constraints on representing as sets, the mapping function  $f$  is required to be invertible and to be isomorphic to a powerset.

Let us consider the satisfaction of the Constraint 3 on representing sequences as sets, that corresponds to the problem of finding a mapping from sequences to powerset.

**Problem 1** (Bijective Powerset Mapping). *Given a finite non-empty set  $S$ , find a bijective function  $f$  that maps  $S$  to a non-empty powerset  $\mathcal{P}(\mathcal{R})$ .*

If a bijective mapping function  $f : S \rightarrow \mathcal{P}(\mathcal{R})$  exists, the size of set  $S$  must be a power of 2. For the problem of finding frequent sets, given attributes  $R$ , let  $S$  be the set of all subsets of  $R$  defined by the language  $\mathcal{L}$ , then  $S = \mathcal{P}(\mathcal{R})$  is a powerset and  $|S| = 2^{|R|}$ , thus an identity mapping  $f(S) = S$  can be used in this case.

**Property 1.** *Given a data set on binary valued attributes  $R$ , let  $\mathcal{L}_{seq}$  be a class of sentences which defines all sequences generated from  $R$ , and the*

length of any sequence is no more than a given value. There does not exist bijective mapping functions that map the language  $\mathcal{L}_{seq}$  to a powerset.

*Proof.* The proof is immediate. Since the number of sequences defined by the language  $\mathcal{L}_{seq}$  is not a power of 2, it does not exist bijective functions that maps  $\mathcal{L}_{seq}$  to a powerset.  $\square$

**Property 2.** Given a data set over binary valued attributes  $R$ , let  $\mathcal{L}_{set}$  be the language for describing all subsets of attributes  $R$ . It does not exist bijective functions that map  $\mathcal{L}_{set}$  to a language  $\mathcal{L}_{seq}$  for defining sequences.

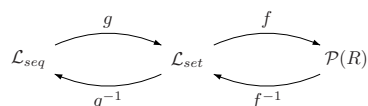


Figure 7.1: Bijective mapping composition.

Shown in Figure 7.1, if there exists a bijective function maps  $\mathcal{L}_{set}$  to  $\mathcal{L}_{seq}$ , for example functions  $g$  and  $g^{-1}$ , then let  $h = g \circ f$  we have that the language  $\mathcal{L}_{seq}$  can be mapped to a powerset by  $h$ . This is concrete to Theorem 1.

Therefore the description language  $\mathcal{L}_{seq}$  for defining sequences does not satisfy the conditions on representing as sets, it does not exist any mapping function maps  $\mathcal{L}_{seq}$  to a powerset.

## 7.2 Position Numbering

Given a data set on binary valued attributes  $R$ , a sequence is an ordered list of subsets of  $R$ . Let us consider a following sequence,

$$s = \langle (AB)(C)(BC)(ABD) \rangle.$$

If we label each subset of  $R$  in the sequence by an integer for its position in a sequence, then each attribute of  $R$  contained in each subset can be distinguished by the same number, thus the above sequence  $s$  can be represented by a set of attributes with position numbers,

$$s = \{A_1, B_1, C_2, B_3, C_3, A_4, B_4, D_4\}.$$

We construct a new set  $R_N$  of attributes with the position numbers from  $R$ , given a limit of length of sequence  $n$ , the size of  $R_N$  is  $|R| \cdot n$ . Let  $\mathcal{L}_N$  be the description language for defining all subsets of  $R_N$ , it is easy to see

that  $\mathcal{L}_N$  respects all constraints on representing as sets, and the hypergraph transversals can be applied to compute the negative border in the discovery of  $L_N$ .

**Example 12.** Consider two subsets of  $s_1, s_2 \in R_N$ , they are

$$\begin{aligned} s_1 &= \{A_1, B_2, C_3\} \\ s_2 &= \{A_2, B_4, C_6\}. \end{aligned}$$

With the position numbering, the sequence  $s = \langle(A)(B)(C)\rangle$  can be represented as  $s_1$ , but  $s_2$  is ambiguous in this case, it does not represent any sequence. For representing sequences, the sentence defining  $s_2$  is a redundant one in  $\mathcal{L}_N$ .  $\square$

The language  $\mathcal{L}_N$  cannot be mapped to  $\mathcal{L}_{seq}$  bijectively, otherwise it is concrete to Theorem 2. It is clear that

$$\mathcal{L}_{seq} \subset \mathcal{L}_N,$$

and the results of discovering  $R_N$  can not be directly applied to find frequent sequences.

## Chapter 8

# Ordered Patterns and Sequences

In this chapter we present a model for representing and sequences with respect to the constraints on representing as sets. We first introduce patterns in data mining, and then we propose a model of *ordered patterns* for representing sequences. We also show how to use transform the problem of discovering sequences to the problem of discovering ordered patterns.

### 8.1 Patterns and Data Mining

Given attributes  $R$ , a *pattern* is an attribute set  $X \subseteq R$ . Let  $\mathcal{L}_R$  denote the language defining of all subsets of  $R$ , a pattern can be defined by a sentence in the language  $\mathcal{L}_R$ . Without losing generality, we denote the pattern as  $I_\varphi$  where  $\varphi \in \mathcal{L}_R$ . That is,

$$\varphi \in \mathcal{L}_R \equiv I_\varphi \subseteq R.$$

From the definition of  $\mathcal{L}_R$ , we have that the description language  $\mathcal{L}_R$  describes the powerset of  $R$  if we consider the empty set as a part of  $\mathcal{L}_R$ . The size of  $\mathcal{L}_R$  is therefore

$$|\mathcal{L}_R| = 2^{|R|}.$$

In our following descriptions, we use  $\varphi$  or  $I_\varphi$  equivalently to describe a pattern without special remarks. And we do not consider any difference between a pattern, an attribute set, or an itemset since they are the same definition.

Typically, the data set given for data mining can be generalized to be a relation  $\mathbf{r}$  with  $n$  rows over binary valued attributes  $R$ . Figure 8.1 shows a representation of such data set, every row with the value *true*, or marked

row	$R_1$	$R_2$	$R_3$	$R_4$	...	$R_m$
1	×		×		...	
2		×	×		...	×
3	×			×	...	
4		×	×		...	
5				×	...	×
6	×	×	×		...	
...	...	...	...	...	...	...
n		×		×	...	

Figure 8.1: Patterns in data mining.

by symbol “×” in the figure, represents a set of patterns. Furthermore, we use a sentence  $\varphi_i \in \mathcal{L}_R$  to describe all attributes valued by *true* in row  $i$ .

**Example 13.** Consider the discovery of frequent itemsets, that is to find all frequent patterns within a given data set such like the one shown in Figure 8.1. Each row in the figure implies a set of patterns, and therefore the task is to compute the frequency of every pattern.  $\square$

**Example 14.** The input data for mining sequential patterns can be viewed as multiple instances of the data set shown in Figure 8.1. For instance, by using SQL queries like “GROUP BY CID ORDER BY TID”, it is easy to convert the relation shown in Figure 2.3(a) to the form shown in Figure 8.1, which can be considered as all transactions of a single customer, or in other words, a customer sequence. The task is therefore to find maximal frequent sequences in all customer sequences.  $\square$

## 8.2 The Ordered Patterns

Given a data set  $\mathbf{r}$  over  $n$  rows of binary valued attributes  $R$ , an *ordered pattern* is a pair  $(I_\varphi, o)$ , where  $I_\varphi \subseteq R$  is a pattern and  $1 \leq o \leq n$  is an integer, the row number of the pattern. We call  $o$  the *order* of an ordered pattern.

Let  $\mathcal{L}_R$  denote the language for defining all subsets of  $R$ , each sentence  $\varphi \in \mathcal{L}$  can be also expressed by a subset  $I_\varphi \subseteq R$ . Without ambiguous, we use  $(\varphi, o)$  or  $(I_\varphi, o)$  in depicting the same ordered pattern.

**Example 15.** Let us consider the data set presented in Figure 8.2. The rows with number 1 and 6 have the same pattern  $\{R_1, R_3, R_5\}$ . They can be distinguished by ordered patterns  $(\{R_1, R_3, R_5\}, 1)$  and  $(\{R_1, R_3, R_5\}, 6)$ , means that they are two different ordered patterns.  $\square$

Any data set with the form shown in Figure 8.2 can be represented by a

row	$R_1$	$R_2$	$R_3$	$R_4$	$R_5$
1	×		×		×
2		×		×	×
3	×		×	×	×
4		×	×		
5				×	×
6	×		×		×
7	×		×	×	
8	×	×	×		×

Figure 8.2: Sample data set for data mining tasks.

set of ordered patterns. In the problem of mining sequential patterns, each customer sequence can be represented by a set of ordered patterns.

An ordered pattern is a set of attributes with an order, it can be rewritten as a follows,

$$(I_\varphi, o) = \{(R_1, o), (R_2, o), \dots, (R_j, o)\},$$

where  $R_1, R_2, \dots, R_j \in R$  and  $|I_\varphi| = j$ . The pair such as  $(R_i, o)$  is an *ordered attribute*, where  $R_i \in R$ . Let  $R_A$  denote the set of ordered attributes on  $R$ , we have

$$R_A = \{(R_i, o) \mid R_i \in R, 1 \leq o \leq n\},$$

and

$$|R_A| = |R| \cdot n.$$

Therefore, there exist two forms for representing ordered patterns. We call the form based on pattern, like  $(I_\varphi, o)$ , the *P-form*, and the form based on attribute, like  $\{(R_1, o), (R_2, o), \dots, (R_j, o)\}$ , the *A-form*.

The following characteristics of ordered patterns are important.

- For patterns  $I_\varphi, I_\theta, I_\gamma$  and an order  $o$ , if  $I_\gamma = I_\varphi \cup I_\theta$ , then we have  $(I_\gamma, o) = (I_\varphi, o) \cup (I_\theta, o)$  for the same order  $o$ . That is,

$$I_\gamma = I_\varphi \cup I_\theta \iff (I_\gamma, o) = (I_\varphi, o) \cup (I_\theta, o).$$

- For patterns  $I_\varphi, I_\theta$  and an order  $o$ , if  $I_\varphi \subseteq I_\theta$ , then we have  $(I_\varphi, o) \subseteq (I_\theta, o)$  for the same order  $o$ . that is,

$$I_\varphi \subseteq I_\theta \iff (I_\varphi, o) \subseteq (I_\theta, o).$$

- Any two ordered pattern with different orders cannot be equivalence. That is,

$$i \neq j \Rightarrow (I_\varphi, i) \neq (I_\varphi, j).$$

Especially in *A-form*, for each two attributes associated with orders, we have always the following,

$$\{(R_i, i), (R_j, j)\} = \{(R_j, j), (R_i, i)\}.$$

### 8.3 Finding All Interesting Ordered Patterns

We first consider the description languages for discovering ordered patterns with the two forms introduced in previous section.

Given data set on attributes  $R$  and order  $n$ , the powerset  $\mathcal{P}(R)$  of  $R$  defines all subsets of  $R$ . Thus, the description language  $\mathcal{L}_P$  based on *P-form* can be defined as following,

$$\mathcal{L}_P = \mathcal{P}(\{(X, o) \mid X \in \mathcal{P}(R), 1 \leq o \leq n\}).$$

The size of  $\mathcal{L}_P$  is therefore

$$|\mathcal{L}_P| = 2^{n \cdot 2^{|R|}}.$$

The description language  $\mathcal{L}_A$  based on *A-form* can be defined as following,

$$\mathcal{L}_A = \mathcal{P}(\{(X, o) \mid X \in R, 1 \leq o \leq n\}).$$

The size of  $\mathcal{L}_A$  is therefore

$$|\mathcal{L}_A| = 2^{n \cdot |R|}.$$

In this Master thesis we consider only the discovery of the ordered patterns represented by the *A-form*, that is, the sentences defined by description language  $\mathcal{L}_A$ .

**Example 16.** Given a data set  $\mathbf{r}$  over 5 rows of binary-valued attributes  $R = \{A, B, C\}$ . We have  $R_O = \{(A, 1), (B, 1), (C, 1), \dots, (A, 5), (B, 5), (C, 5)\}$ . Here we use the shorthand notations for sets of ordered attributes and of ordered patterns, for example, we represent  $\{(A, 1), (B, 2), (C, 3)\}$  by  $A^1B^2C^3$  and  $\{(\{A, B\}, 1), (\{B, C\}, 2)\}$  by  $(AB)^1(BC)^2$ . Let  $\mathcal{I}_\varphi = A^1B^1A^2B^3C^3$  be a subset of  $R_O$ , then  $\mathcal{I}_\varphi$  represents a set of ordered patterns, that is,

$$\mathcal{I}_\varphi = A^1B^1A^2B^3C^3 = (AB)^1(A)^2(BC)^3.$$

□

We now discuss the specialization relation on the description language  $\mathcal{L}_a$ . The theory extraction framework requires the specialization relation  $\preceq$  on the sentences in description language  $\mathcal{L}$  [12]. For any language  $\mathcal{L}$  describing powerset  $\mathcal{P}(\mathcal{R})$ , the specialization relation is exactly the inclusion relation between all subsets of  $\mathcal{R}$ , that is,

$$\varphi \preceq \theta \iff I_\varphi \subseteq I_\theta,$$

where  $\varphi, \theta \in \mathcal{L}$  and  $I_\varphi, I_\theta \subseteq \mathcal{R}$ .

**Example 17.** Let us consider again the data set presented in Example 16. Assume two sentences  $\varphi, \theta \in \mathcal{L}_a$  such that  $\varphi$  defines the ordered pattern set  $I_\varphi = (AB)^1(B)^2$  and  $\theta$  defines  $I_\theta = (AB)^1(BC)^2$ , then we have

$$I_\varphi \subset I_\theta \iff A^1B^1B^2 \subset A^1B^1B^2C^2,$$

thus we have

$$\varphi \prec \theta \iff I_\varphi \subset I_\theta.$$

If given another sentence  $\gamma \in \mathcal{L}_a$  such that  $I_\gamma = (AB)^1(B)^3$ , we do not have  $\gamma \prec \theta$  because

$$I_\gamma \not\subseteq I_\theta,$$

that is,

$$A^1B^1B^3 \not\subseteq A^1B^1B^2C^2,$$

and between ordered attribute sets, we have

$$A^1B^1B^2 \neq A^1B^1B^3.$$

□

An ordered attribute is a composition of an attribute and an order, thus it can be viewed as a special attribute. Therefore all propositions and conclusions on mining frequent sets in [12] and other approaches like [2, 9, 13, 20] are suitable for discovering frequent ordered patterns. Here we focus on the application of transversal hypergraph computation.

**Example 18.** Given data set  $r$  with attributes  $R = \{A, B\}$ , let language  $\mathcal{L}_a^2$  define all ordered patterns with the order  $o \leq 2$  in *A-Form*, we depict the language  $\mathcal{L}_a^2$  as a lattice shown in Figure 8.3. Assume a set of sentences  $\mathcal{S} \subseteq \mathcal{L}_a^2$  closed downwards to the relation  $\preceq$ ,

$$\mathcal{S} = \{A^1, B^1, A^2, B^2, A^1B^1, A^1B^2, B^1B^2, A^2B^2, A^1B^1B^2\},$$

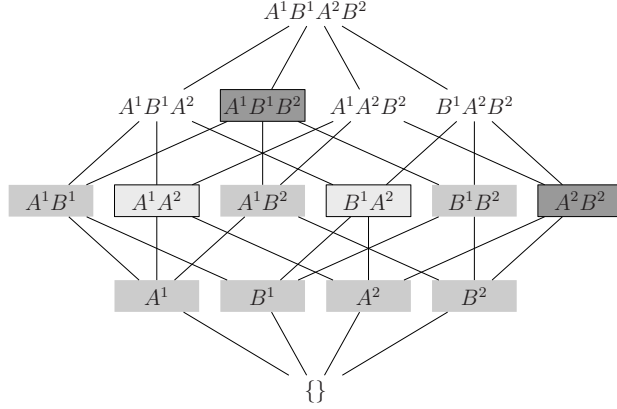


Figure 8.3: Transversal hypergraph computation is applicable to a lattice representing ordered patterns.

and  $\mathcal{S}$  includes the maximal ordered pattern sets  $\{A^1B^1B^2, A^2B^2\}$ . The negative border

$$\mathcal{B}d^-(\mathcal{S}) = \{A^1A^2, B^1A^2\}.$$

For this problem, we already have  $\mathcal{L}_a^2$  represented as sets and the mapping function  $f$  is an identity mapping. With the application of hypergraph transversals, we have therefore

$$\mathcal{B}d^+(\mathcal{S}) = \{A^1B^1B^2, A^2B^2\} \Rightarrow \mathcal{H}(\mathcal{S}) = \{A^2, A^1B^1\},$$

thus we have the minimal transversals of  $\mathcal{H}(\mathcal{S})$

$$\mathcal{T}r(\{A^2, A^1B^1\}) = \{A^1A^2, B^1A^2\},$$

and thus the application of hypergraph transversals returns the correct answer.  $\square$

It is notable that the search space for discovering all interesting ordered patterns is increased by a power of 2 with the order, since the size of  $\mathcal{L}_a$  for given attributes  $R$  and maximal order  $k$  is  $2^{|R| \cdot k}$ .

## 8.4 Representing Sequences

We use a *sequential relation* between the attribute sets in a sequence. The sequential relation is a total order  $\mapsto^o$  that a pattern  $I_\varphi$  is precedent to another pattern  $I_\theta$  if  $I_\varphi \mapsto^o I_\theta$ . Let  $o$  denote the order of the sequential relation <sup>1</sup>, defined as follows: Given a sequence  $s$  with a length of  $k$ , if for

<sup>1</sup>Similar but different to the *order* in the notion *ordered pattern*.

no  $I_\gamma$  in  $s$  we have  $I_\gamma \mapsto^o I_\varphi$ , then  $o = 1$ ; otherwise,  $o = \max(\{o' \mid I_\theta \mapsto^{o'} I_\varphi\}) + 1$ . Note that  $I_\theta \neq I_\varphi$  is not required for computing  $I_\theta \mapsto^{o'} I_\varphi$ . In particular, for a sequence with length  $k$ , we define  $o = k$  if for no  $I_\gamma$  in  $s$  we have  $I_\varphi \mapsto^o I_\gamma$ , and the order  $o$  is therefore an integer such that  $1 \leq o \leq k$ .

Given data set over attributes  $R$ , let  $s^k$  denote a sequence consists of  $k$  patterns,  $s^k$  can therefore be described as follows.

$$s^k = \langle I_{\varphi_1} \mapsto^1 I_{\varphi_2} \mapsto^2 \dots \mapsto^{k-1} I_{\varphi_k} \mapsto^k \emptyset \rangle,$$

where  $I_{\varphi_1}, I_{\varphi_2}, \dots, I_{\varphi_k} \subseteq R$  are  $k$  patterns. We use an empty set to bound a sequence. If we consider a pattern and its following sequential relation as a pair, such as  $(I_{\varphi_i}, \mapsto^o)$ , then we can represent a sequence as a set of pairs which are similar to ordered patterns, that is,

$$s^k = \{(I_{\varphi_1}, \mapsto^1), (I_{\varphi_2}, \mapsto^2), \dots, (I_{\varphi_k}, \mapsto^k)\},$$

where the trailing empty set can be safely removed.

Given a data set with  $k$  rows over binary-valued attributes  $R$ , we already have that a set of ordered patterns with the length of  $n$  represents a *customer sequence*. According to [3], a *sequence* is contained in a customer sequence, it could be a part of the customer sequence, either the whole customer sequence. Thus we can formally define a  $k$ -sequence by ordered patterns as follows.

$$s^k = \{(I_{\varphi_1}, o_1), (I_{\varphi_2}, o_2), \dots, (I_{\varphi_k}, o_k)\},$$

where  $1 \leq o_1 < o_2 < \dots < o_k \leq n$ . Given two ordered pattern sets  $s_1$  and  $s_2$ , which are defined as following,

$$\begin{aligned} s_1 &= \{(I_{\varphi_1}, i_1), (I_{\varphi_2}, i_2), (I_{\varphi_3}, i_3)\} \\ s_2 &= \{(I_{\varphi_1}, j_1), (I_{\varphi_2}, j_2), (I_{\varphi_3}, j_3)\}, \end{aligned}$$

where  $i_1 < i_2 < i_3$ ,  $j_1 < j_2 < j_3$  and there have at least that  $i_1 \neq j_1$  or  $i_2 \neq j_2$  or  $i_3 \neq j_3$ , than  $s_1$  and  $s_2$  are two different ordered pattern sets representing the same 3-sequence

$$s^3 = \{(I_{\varphi_1}, \mapsto^1), (I_{\varphi_2}, \mapsto^2), (I_{\varphi_3}, \mapsto^3)\}.$$

It is not difficult to find that if we replace the sequential relation by consecutive integers, a sequence has the same form with a set of ordered patterns.

**Definition 2** (Sequence). *A sequence can be represented by a set of ordered patterns with consecutive orders starting from 1.*

We can use a *production function*  $p$  to transform a set of ordered patterns to a sequence. Given a sentence  $\varphi \in \mathcal{L}_P$ , such as,

$$\varphi = \{(I_{\varphi_1}, o_1), (I_{\varphi_2}, o_2), \dots, (I_{\varphi_k}, o_k)\}.$$

where  $o_1 < o_2 < \dots < o_k$ . The production  $p(\varphi)$  returns a new sentence  $\theta \in \mathcal{L}_A$ ,

$$\{(I_{\varphi_1}, 1), (I_{\varphi_2}, 2), \dots, (I_{\varphi_k}, k)\},$$

The sentence  $\theta$  consists of ordered patterns with consecutive orders, it represents a sequence. We say the sentence  $\theta$  is the *alias* of the sentence  $\varphi$ .

In the case of using description language  $\mathcal{L}_A$ , the definition of alias is the same, but with different form.

Given a sentence  $\varphi \in \mathcal{L}_A$ , such as,

$$\varphi = \{(R_1, o_1), \dots, (R_{m_1}, o_1), \dots, (R_1, o_k), \dots, (R_{m_k}, o_k)\}.$$

where  $o_1 < o_2 < \dots < o_k$ . The alias  $\theta \in \mathcal{L}_A$  is therefore

$$\{(R_1, 1), \dots, (R_{m_1}, 1), \dots, (R_1, k), \dots, (R_{m_k}, k)\}.$$

It is remarkable that the production function  $p$  is not invertible, so that it cannot be used in representing sequences as sets.

With this representation, the problem of discovering all frequent sequences can be performed within the discovery of frequent ordered patterns. During the discovery of frequent ordered patterns, when a sentence  $\varphi \in \mathcal{L}_P$  holds the quality predicate, the support of the alias  $\theta$  of  $\varphi$  is updated. Therefore, the task of mining sequential patterns can be handled by the discovery of all frequent ordered patterns.

## Chapter 9

# Discovering Ordered Patterns for Sequential Patterns

The results of Chapter 8 show that a sequence can be represented by a set of ordered patterns with consecutive orders starting from 1. On the other hand, a sequential patterns is a sequence, thus this model of sequence representation can be used in representing sequential patterns.

In this chapter we present how to use the model of ordered patterns in mining sequential patterns, with transversal hypergraph computation. The *Dualize and Advance* algorithm [9] is used in the discovery of ordered patterns.

### 9.1 Overview

We are given a database  $\mathcal{D}$  of customer transactions, each transaction consists of transaction time, customer identification and a set of items. We do not consider the quantities of items bought in the transactions, thus each item is represented by a binary valued attribute  $R_i \in R$  where  $R$  stands the set of all items. A *minimal\_support* is also given as the frequency threshold.

Figure 9.1(a) shows a mini instance of such database  $\mathcal{D}$ , it consists in 12 transactions of 4 customers and 2 items. Each customer corresponds to 3 transactions.

It is easy to convert this database  $\mathcal{D}$  to a set of customer sequences by ordering the transaction time and grouping the customer identifications. Each customer sequence is a *slice* of database  $\mathcal{D}$ . In each slice transaction time is sorted by a set of consecutive numbers, i.e., the orders. Figure 9.1(b)

Transaction	Customer	A	B
1	1	1	1
2	2	1	0
3	3	0	1
4	4	1	1
5	1	1	0
6	2	1	1
7	3	0	1
8	4	0	1
9	1	1	0
10	2	1	0
11	3	1	1
12	4	1	0

(a)

Transaction	A	B
1	1	1
2	1	0
3	1	0

Customer 1

Transaction	A	B
1	1	0
2	1	1
3	1	0

Customer 2

Transaction	A	B
1	0	1
2	0	1
3	1	1

Customer 3

Transaction	A	B
1	1	1
2	0	1
3	1	0

Customer 4

(b)

Figure 9.1: (a) A database  $\mathcal{D}$ , it consists in 12 transactions of 4 customers and 2 items. (b) 4 slices of customer transactions.

shows 4 slices of customer transactions.

We use the *A-form* in mining sequential patterns, corresponding to the language  $\mathcal{L}_A$ . Assume there exists  $m$  attributes in  $R$  and the maximal order considered is  $n$ . We first generate the ordered attributes from  $m$  attributes and  $n$  orders, that is,  $|n \cdot m|$  ordered attributes totally for the process of discovering ordered patterns.

We use the *Dualize and Advance* algorithm (detailed in Chapter 4) in finding the positive border of all interesting sentences of  $\mathcal{L}_A$ . The sequential patterns mining process is specified by the quality predicate  $q$ . After the positive border of all sequential patterns with minimal support *minimal\_support* has been computed, we generate all sequential patterns from the positive border and compute the final result. With the *Dualize and Advance* algorithm, we use the transversal hypergraph computation in finding the positive border.

We do not further specify the *permutation* function required by the *AMSS* algorithm (Algorithm 2) in this Master thesis, but we use a lexicographic order for selecting the next attribute.

The *Dualize and Advance* algorithm returns the positive border, thus the result of discovery is all most specific frequent ordered patterns. To complete mining sequential patterns, we generate all sentences from the positive border and then compute the support of each alias separately.

## 9.2 The Quality Predicate for Mining Sequential Patterns

One of the advantages of the framework of Mannila and Toivonen is that we can define individual quality predicate, which is monotone to the specialization relation  $\preceq$ , for individual data mining problem without modifying the algorithm for finding interesting sentences. Thus, by defining such quality predicate for mining sequential patterns with ordered patterns, the *Dualize and Advance* algorithm can therefore be used without modification.

Our proposition of the quality predicate  $q$  for mining sequential patterns is defined as follows. Given a set  $\mathcal{S}$  of slices and a sentence  $\varphi \in \mathcal{L}_A$ ,  $q$  evaluates  $\varphi$  against each  $s \in \mathcal{S}$ . If  $\varphi$  does not exist in any  $s$ ,  $q$  returns *false* without further evaluations. Otherwise,  $q$  computes the alias  $\theta \in \mathcal{L}_A$  of  $\varphi$  and expands  $\theta$  to obtain all sentences  $\mathcal{E}$  ( $\varphi \notin \mathcal{E}$ ) having the same alias  $\theta$ .  $q$  then evaluates  $\tau \in \mathcal{E}$  in each slice, and updates the rank of  $\theta$  for computing the support of  $\theta$ . If the support of  $\theta$  is  $\geq \text{minimal\_support}$   $q$  records  $\theta$  as interesting and returns *true* otherwise  $q$  returns *false*.

**Property 3.** *The above quality predicate  $q$  is monotone to the specialization relation  $\preceq$  on the language  $\mathcal{L}_A$  and the set  $\mathcal{S}$  of slices, where each slice  $s \in \mathcal{S}$  is a customer sequence for mining sequential patterns.*

*Proof.* We already have that the sentences of  $\mathcal{L}_A$  respect the specialization  $\preceq$ . If a sentence  $\varphi \in \mathcal{L}_A$  is interesting, then we have the alias  $\theta \in \mathcal{L}_A$  interesting, means that the sequence  $s_\theta$  represented by  $\theta$  is frequent, and  $\varphi$  exists in at least one slice  $s \in \mathcal{S}$ . And according to the relation  $\preceq$  on  $\mathcal{L}_A$ , any generalization of  $\varphi$  must exist in at least one slice  $s \in \mathcal{S}$ , and any sub-sequences of  $s_\theta$  must be frequent. Thus we have that for  $\gamma \in \mathcal{L}_A$  and  $\gamma \preceq \varphi$ , if  $q(\mathcal{S}, \varphi) = \text{true}$ , then  $q(\mathcal{S}, \gamma) = \text{true}$ .

Next we show that for  $\gamma \in \mathcal{L}_A$  and  $\gamma \preceq \varphi$ , if  $q(\mathcal{S}, \gamma) = \text{false}$ , then  $q(\mathcal{S}, \varphi) = \text{false}$ .

$q(\mathcal{S}, \gamma) = \text{false}$  means that  $\gamma$  does not exist in any slice  $s \in \mathcal{S}$  or the sentence represented by the alias  $\psi \in \mathcal{L}_A$  of  $\gamma$  is not frequent. In the first case, any specialization of  $\gamma$  cannot exist in any slice  $s \in \mathcal{S}$ . In the second case, the sequence represented by the alias of any specialization of  $\gamma$  cannot be frequent. Thus for any sentence  $\varphi \in \mathcal{L}_A$  and  $\gamma \preceq \varphi$ , we have  $q(\mathcal{S}, \varphi) = \text{false}$ .  $\square$

Therefore, the quality predicate defined in this section is monotone to the specialization relation  $\preceq$  on  $\mathcal{L}_A$  and it updates correctly the frequency of sequences with respect to the definition of *support* for sequential patterns.

Thus, the ordered patterns model and this quality predicate to the problem of mining sequential patterns can address the framework of Mannila and Toivonen and the transversal hypergraph computation.

---

**Algorithm 5:** Quality predicate  $q$  for mining sequential patterns.

---

```

1 if exists  $\gamma \preceq \varphi$  not interesting then
2   | return false ;
3 end
4  $alias\_rank \leftarrow 0$ ;
5 foreach  $s \in \mathcal{S}$  do
6   |  $rank \leftarrow$  evaluate  $\varphi$  against  $s$ ;
7   | if  $rank > 0$  then
8     | update  $alias\_rank$  by  $rank$  and remove  $s$  from  $\mathcal{S}$ ;
9   | end
10 end
11 if  $alias\_rank = 0$  then
12   | return false ;
13 end
14  $\theta \leftarrow$  alias of  $\varphi$ ;
15  $\mathcal{E} \leftarrow$  all sentences with the same alias  $\theta$  but excluding  $\varphi$ ;
16 foreach  $s \in \mathcal{S}$  do
17   | foreach  $\tau \in \mathcal{E}$  do
18     |  $rank \leftarrow$  evaluate  $\tau$  against  $s$ ;
19     | if  $rank > 0$  then
20       | update  $alias\_rank$  by  $rank$  and remove  $s$  from  $\mathcal{S}$ ;
21     | end
22   | end
23 end
24 if  $alias\_rank/number\_of\_slices \geq minimal\_support$  then
25   | record  $\theta$  as interesting;
26   | return true ;
27 end
28 return false ;

```

---

Algorithm 5 shows this quality predicate  $q$  for mining sequential patterns. It predicates *true* or *false* by inputting a set  $\mathcal{S}$  of slices and a sentence  $\varphi \in \mathcal{L}_A$ .

It is notable that after each evaluation, if  $\varphi$  or  $\tau$  exists in current  $s$ , then current  $s$  will be temporarily removed from  $\mathcal{S}$  for this session of  $q$ , since

in mining sequential patterns each sequence is counted only once for each customer sequence.

Furthermore, at the beginning of the algorithm, quality predicate  $q$  first check whether a more general sentence  $\gamma$  exists. If there exists a non-interested sentence  $\gamma \preceq \varphi$ , the algorithm returns *false* without any further computing. It may be difficult to record all evaluated sentences, but a mechanism of caching sentences interestingness is still helpful.

### 9.3 Alias Generation

Given a sentence  $\varphi \in \mathcal{L}_A$ , Algorithm 6 returns the alias  $\theta$  of sentence  $\varphi$ , that is,

$$alias(\varphi) = \{(R_1, 1), \dots, (R_{m_1}, 1), \dots, (R_1, k), \dots, (R_{m_k}, k)\}.$$

---

**Algorithm 6:** Alias generation for mining sequential patterns.

---

```

1  $\theta \leftarrow \{\}$ ;
2  $i \leftarrow 0$ ;  $current\_order \leftarrow 0$ ;
3  $last\_order \leftarrow 0$ ;
4 while  $i < length\_of(\varphi)$  do
5   | extract  $(attribute, order)$  from  $\varphi[i]$ ;
6   | if  $order > last\_order$  then
7   |   |  $current\_order \leftarrow current\_order + 1$ ;
8   |   end
9   | append  $(attribute, current\_order)$  to  $\theta$ ;
10  |  $last\_order \leftarrow order$ ;
11  |  $i \leftarrow i + 1$ ;
12 end
13 return  $\theta$ ;
```

---

In this algorithm we assume that all attributes presented in the input sentence  $\varphi$  are sorted in the ascendant of the order. The algorithm resorts all ordered attributes in the sentence  $\varphi$  by consecutive order numbers starting from 1 and returns the new list of ordered patterns as the alias  $\theta$  of  $\varphi$ . More efficient algorithms are need for the routine of alias generation.

### 9.4 Evaluation Process

The evaluation process for quality predicate  $q$  accepts a slice  $s$  and a sentence  $\varphi \in \mathcal{L}_A$  as its input. It determines whether the ordered patterns described

	1	2	3	4	5	6	7	8	9	...
1	0	1	1	0	1	0	0	0	0	...
2	1	1	0	0	0	1	0	1	0	...
3	0	1	1	1	0	0	1	1	0	...
4	1	1	1	0	0	1	1	0	0	...
5	0	0	1	0	1	0	1	0	0	...

Figure 9.2: The bitmap representation for a slice.

by  $\varphi$  is present in  $s$ . If  $\varphi$  is present the function returns 1 otherwise it returns 0.

This function does not compute the support of the input sentence, however it checks the existence of ordered patterns expressed by the sentence. Using the bitmap representation of patterns introduced in Section 9.6, this matching can be efficient.

## 9.5 Slicing Database

Let us recall the context of mining sequential patterns, shown in Figure 2.3. We are given a database  $\mathcal{D}$  of customer transactions, each transaction consists in a transaction time, a customer identification and a set of items corresponding to a subset of  $m$  attributes  $R$ . The task of mining sequential patterns is specified by a user defined frequency threshold *minimal\_support*.

It is easy to convert the given database  $\mathcal{D}$  to a set of customer sequences by ordering the transaction time and grouping the customer identifications. Each customer sequence is a *slice* of database  $\mathcal{D}$ . In each slice transaction time is sorted by a set of consecutive numbers, i.e., the orders.

A slice is split into layers, where each layer stands for a transaction, that is, all ordered patterns with the same order. In each layer, we use a bitmap for representing present attributes. A bitmap representation is useful to data mining tasks, such that in [4] a bitmap representation is used in representing sequences.

**Example 19.** Figure 9.2 shows the bitmap representation for a slice. This slice consists of 8 attributes (represented by numbers from 1 to 8) and 5 order numbers. Thus, there are totally 5 ordered patterns in the slice corresponding to 5 layers, they are  $(\{2, 3, 5\}, 1)$ ,  $(\{1, 2, 6, 8\}, 2)$ ,  $(\{2, 3, 4, 7, 8\}, 3)$ ,

({1, 2, 3, 6, 7}, 4) and ({3, 5, 7}, 5). □

Algorithm 7 represent a customer sequence as a set of bitmaps. The algorithm requires that the customer sequence has been sorted by transaction time and the set  $R$  of attributes has been identified as a set of consecutive integers from 1 to  $m$ .

---

**Algorithm 7:** Bitmap representation of customer sequence.

---

```

1  $m \leftarrow$  size of  $R$ ;
2  $x \leftarrow$  number of bits required to represent  $m$  attributes;
3  $s \leftarrow$  empty array;
4 foreach  $t \in r$  do
5   initialize bitmap with  $x$ ;
6   foreach  $item\_id \in t$  do
7      $p \leftarrow$  position in bitmap of current attribute;
8     set bit to 1 at position  $p$  of bitmap;
9   end
10  expand  $s$  by bitmap;
11 end
12 return  $s$ ;
```

---

## 9.6 The Complexity

In Section 4.3 and Theorem 4 we introduce the time complexity and query complexity of the *Dualize and Advance* algorithm. The results show that the time complexity of this algorithm depends on the time complexity of transversal hypergraph computation. Using the algorithm of Fredman and Khachiyan [8] in computing transversal hypergraph incrementally, the time complexity of the *Dualize and Advance* algorithm is polynomial in  $|\mathcal{MTh}|$  and  $T(|\mathcal{MTh}|, |\mathcal{Bd}^-(\mathcal{MTh})|)$  where  $T(n) = n^{\mathcal{O}(\log(n))}$ .

With our proposition, given a sentence  $\varphi \in \mathcal{L}_A$  and its alias  $\theta \in \mathcal{L}_A$ , let  $\sum \theta^*$  denote the number of all sentences with the same alias  $\theta$ , and let  $\sum \theta$  denote all sentences present in at least one customer sequence and with the same alias  $\theta$ , then in the worst case we have  $\sum \theta = \sum \theta^*$ .

Given a most specific interesting sentence  $\varphi \in \mathcal{L}_A$ , if the alias  $\theta \in \mathcal{L}_A$  of  $\varphi$  holds the *minimal\_support* defined for mining sequential patterns, then the size of  $\mathcal{MTh}$  depends on  $|\sum \theta|$ .

The query complexity shows that the number of queries depends on the size  $|\mathcal{MTh}|$  and  $|\mathcal{Bd}^-(\mathcal{MTh})|$  and the width of  $\mathcal{L}_A$  on  $\preceq$ , which is  $|n \cdot |R||$

where  $n$  is the number of orders we considered for bounding sequences.

Therefore, our proposition for mining sequential patterns with transversal hypergraph computation respects the time complexity and query complexity of the *Dualize and Advance* algorithm.

## 9.7 A Detailed Example

In this section we present a detailed example for mining sequential patterns with transversal hypergraph.

**Example 20.** The input database of customer transactions is shown in Figure 9.1. We are given a tiny database  $\mathcal{D}$  of customer transactions, including 4 customers, 3 transactions maximum for each customer and 2 items  $A$  and  $B$ . Thus we have 4 slices of customer sequences and each slice contains 3 layers of 2 attributes. Therefore, the description language  $\mathcal{L}_A$  is a powerset of ordered attributes  $A^1, A^2, A^3, B^1, B^2, B^3$ .

For simplifying the description, we use **amss** as a shorthand of the *AMSS* algorithm (Algorithm 2), **allamss** as a shorthand of the *ALLAMSS* algorithm (Algorithm 3), and **htr** as a shorthand of the algorithm for transversal hypergraph computation (Algorithm 4). Furthermore, for instance by  $A^1$ , we depict the calling of quality predicate  $q$  without the data set, that is, writing  $q(A^1)$  instead of  $q(\mathcal{S}, A^1)$ . Other notations are used with respect to the same notations in Algorithm 5.

To do not make this example too long, we define *minimal\_support* = 0.5.

**Iteration 1 of allmass** The **allamss** starts at  $\mathcal{C} = \{\}$ , thus it is easy to find that

$$\overline{\mathcal{D}}_1 = \{A^1, B^1, A^2, B^2, A^3, B^3\}$$

and thus by the **htr** we have temporarily

$$\mathcal{T}_1 = \{A^1, B^1, A^2, B^2, A^3, B^3\}.$$

The **htr** further evaluates them by  $q$  and removes the sentences non-interested. There is no sentence to be removed, thus we have

$$\mathcal{T}_1 = \{A^1, B^1, A^2, B^2, A^3, B^3\},$$

$A^1$  is selected to be passed to the **amss**. We have

$$\begin{aligned} q(A^1 B^1) &= true, \\ q(A^1 B^1 A^2) &= true, \\ q(A^1 B^1 A^2 B^2) &= false, \\ q(A^1 B^1 A^2 A^3) &= false, \\ q(A^1 B^1 A^2 B^3) &= false. \end{aligned}$$

Thus **amss** returns  $A^1 B^1 A^2$ . Thus we have

$$\mathcal{C} = \{A^1 B^1 A^2\}.$$

**Iteration 2 of allmass** Now we have

$$\overline{D}_2 = \{B^2 A^3 B^3\}$$

and thus by the **htr** we have temporarily

$$\mathcal{T}_2 = \{B^2, A^3, B^3\}.$$

The **htr** further evaluates them by  $q$  and removes the sentences non-interested.

There is no sentence to be removed, thus we have

$$\mathcal{T}_2 = \{B^2, A^3, B^3\}.$$

$B^2$  is selected to be passed to the **amss**. We have

$$\begin{aligned} q(A^1 B^2) &= true, \\ q(A^1 B^2 B^1) &= false, \\ q(A^1 B^2 A^2) &= false, \\ q(A^1 B^2 A^3) &= true, \\ q(A^1 B^2 A^3 B^3) &= false. \end{aligned}$$

Thus **amss** returns  $A^1 B^2 A^3$ . Thus we have

$$\mathcal{C} = \{A^1 B^1 A^2, A^1 B^2 A^3\}.$$

**Iteration 3 of allmass** Now we have

$$\overline{D}_3 = \{B^2 A^3 B^3, B^1 A^2 B^3\}$$

and thus by the **htr** we have temporarily

$$\mathcal{T}_3 = \{B^3, B^1 B^2, A^2 B^2, B^1 A^3, A^2 A^3\}.$$

The **htr** further evaluates them by  $q$  and removes the sentences non-interested. There is no sentence to be removed, thus we have

$$\mathcal{T}_3 = \{B^3, B^1B^2, A^2B^2, B^1A^3\}.$$

$B^3$  is selected to be passed to the **amss**. We have

$$\begin{aligned} q(A^1B^3) &= false, \\ q(B^1B^3) &= true, \\ q(B^1A^2B^3) &= false, \\ q(B^1B^2B^3) &= false, \\ q(B^1A^3B^3) &= false. \end{aligned}$$

Thus **amss** returns  $B^1B^3$ . Thus we have

$$\mathcal{C} = \{A^1B^1A^2, A^1B^2A^3, B^1B^3\}.$$

**Iteration 4 of allmass** Now we have

$$\overline{\mathcal{D}}_4 = \{B^2A^3B^3, B^1A^2B^3, A^1A^2B^2A^3\}$$

and thus by the **htr** we have temporarily

$$\mathcal{T}_4 = \{A^1B^3, A^2B^3, B^2B^3, A^3B^3, B^1B^2, A^2B^2, B^1A^3\}.$$

The **htr** further evaluates them by  $q$  and removes the sentences non-interested. Thus we have

$$\mathcal{T}_4 = \{B^2B^3, A^3B^3, B^1B^2, A^2B^2, B^1A^3\}.$$

$B^2B^3$  is selected to be passed to the **amss**. We have

$$\begin{aligned} q(A^1B^2B^3) &= false, \\ q(B^1B^2B^3) &= false, \\ q(A^2B^2B^3) &= false, \\ q(B^2A^3B^3) &= false. \end{aligned}$$

Thus **amss** returns  $B^2B^3$ . Thus we have

$$\mathcal{C} = \{A^1B^1A^2, A^1B^2A^3, B^1B^3, B^2B^3\}.$$

**Iteration 5 of allmass** Now we have

$$\overline{D}_5 = \{B^2A^3B^3, B^1A^2B^3, A^1A^2B^2A^3, A^1B^1A^2A^3\}$$

and thus by the **htr** we have temporarily

$$\mathcal{T}_5 = \{A^1B^2B^3, B^1B^2B^3, A^2B^2B^3, B^2A^3B^3, A^3B^3, B^1B^2, A^2B^2, B^1A^3\}.$$

The **htr** further evaluates them by  $q$  and removes the sentences non-interested.

Thus we have

$$\mathcal{T}_5 = \{A^3B^3, B^1B^2, A^2B^2, B^1A^3\}.$$

$A^3B^3$  is selected to be passed to the **amss**. We have

$$\begin{aligned} q(A^1A^3B^3) &= false, \\ q(B^1A^3B^3) &= false, \\ q(A^2A^3B^3) &= false, \\ q(B^2A^3B^3) &= false. \end{aligned}$$

Thus **amss** returns  $A^3B^3$ . Thus we have

$$\mathcal{C} = \{A^1B^1A^2, A^1B^2A^3, B^1B^3, B^2B^3, A^3B^3\}.$$

**Iteration 6 of allmass** Now we have

$$\overline{D}_6 = \{B^2A^3B^3, B^1A^2B^3, A^1A^2B^2A^3, A^1B^1A^2A^3, A^1B^1A^2B^2\}$$

and thus by the **htr** we have temporarily

$$\mathcal{T}_6 = \{A^1A^3B^3, B^1A^3B^3, A^2A^3B^3, B^2A^3B^3, B^1B^2, A^2B^2, B^1A^3\}.$$

The **htr** further evaluates them by  $q$  and removes the sentences non-interested.

Thus we have

$$\mathcal{T}_6 = \{B^1B^2, A^2B^2, B^1A^3\}.$$

$B^1B^2$  is selected to be passed to the **amss**. We have

$$\begin{aligned} q(A^1B^1B^2) &= false, \\ q(B^1A^2B^2) &= false, \\ q(B^1B^2A^3) &= true, \\ q(B^1B^2A^3B^3) &= false. \end{aligned}$$

Thus **amss** returns  $B^1B^2A^3$ . Thus we have

$$\mathcal{C} = \{A^1B^1A^2, A^1B^2A^3, B^1B^3, B^2B^3, A^3B^3, B^1B^2A^3\}.$$

**Iteration 7 of allmass** Now we have

$$\overline{D}_7 = \{B^2A^3B^3, B^1A^2B^3, A^1A^2B^2A^3, A^1B^1A^2A^3, A^1B^1A^2B^2, A^1A^2B^3\}$$

and thus by the **htr** we have temporarily

$$\mathcal{T}_7 = \{A^1B^1B^2, B^1A^2B^2, B^1B^2B^3, A^2B^2, A^1B^1A^3, B^1A^2A^3, B^1A^3B^3\}.$$

The **htr** further evaluates them by  $q$  and removes the sentences non-interested.

Thus we have

$$\mathcal{T}_7 = \{A^2B^2, A^1B^1A^3\}.$$

$A^2B^2$  is selected to be passed to the **amss**. We have

$$\begin{aligned} q(A^1A^2B^2) &= false, \\ q(B^1A^2B^2) &= false, \\ q(A^2B^2A^3) &= true, \\ q(A^2B^2A^3B^3) &= false. \end{aligned}$$

Thus **amss** returns  $A^2B^2A^3$ . Thus we have

$$C = \{A^1B^1A^2, A^1B^2A^3, B^1B^3, B^2B^3, A^3B^3, B^1B^2A^3, A^2B^2A^3\}.$$

**Iteration 8 of allmass** Now we have

$$\begin{aligned} \overline{D}_8 = \{ &B^2A^3B^3, B^1A^2B^3, A^1A^2B^2A^3, A^1B^1A^2A^3, \\ &A^1B^1A^2B^2, A^1A^2B^3, A^1B^1B^3\} \end{aligned}$$

and thus by the **htr** we have temporarily

$$\mathcal{T}_8 = \{A^1A^2B^2, B^1A^2B^2, A^2B^2B^3, A^1B^1A^3\}.$$

The **htr** further evaluates them by  $q$  and removes the sentences non-interested.

Thus we have

$$\mathcal{T}_8 = \{A^1B^1A^3\}.$$

$A^1B^1A^3$  is selected to be passed to the **amss**. We have

$$\begin{aligned} q(A^1B^1A^2A^3) &= false, \\ q(A^1B^1B^2A^3) &= false, \\ q(A^1B^1A^3B^3) &= false. \end{aligned}$$

Thus **amss** returns  $A^1B^1A^3$ . Thus we have

$$C = \{A^1B^1A^2, A^1B^2A^3, B^1B^3, B^2B^3, A^3B^3, B^1B^2A^3, A^2B^2A^3, A^1B^1A^3\}.$$

**Iteration 9 of allmass** Now we have

$$\overline{D_9} = \{B^2A^3B^3, B^1A^2B^3, A^1A^2B^2A^3, A^1B^1A^2A^3, \\ A^1B^1A^2B^2, A^1A^2B^3, A^1B^1B^3, A^2B^2B^3\}$$

and thus by the **htr** we have temporarily

$$\mathcal{T}_9 = \{A^1B^1A^2A^3, A^1B^1B^2A^3, A^1B^1A^3B^3\}.$$

The **htr** further evaluates them by  $q$  and removes the sentences non-interested. Thus we have

$$\mathcal{T}_9 = \{\}.$$

There the **allamss** stops here and output all most specific sentences, that is, the positive border consisting of all most specific frequent ordered patterns this database, the set

$$\mathcal{C} = \{A^1B^1A^2, A^1B^2A^3, B^1B^3, B^2B^3, A^3B^3, B^1B^2A^3, A^2B^2A^3, A^1B^1A^3\}.$$

All frequent ordered patterns can be expanded from  $\mathcal{C}$  and therefore the support of each alias can be also computed.

From the set of most specific ordered patterns, we can easily compute the set of maximal sequential patterns. In this example, it is the set

$$\mathcal{C} = \{\langle(AB)(A)\rangle, \langle(A)(B)(A)\rangle, \langle(B)(B)(A)\rangle\}.$$

□

# Conclusions

In this Master thesis we proposed a new approach to use transversal hypergraph computation in mining sequential patterns.

We first gave an overview of the problems of mining association rules and mining sequential patterns. We then introduced the framework of Manila and Toivonen for generalizing data mining tasks and the application of transversal hypergraph computation for discovering frequent itemsets within this framework. We also presented the *Dualize and Advance* algorithm, which is based on transversal hypergraph computation and has been successfully applied to the problem of mining association rules.

We then considered the problem of representing as sets that restricts the application of transversal hypergraph computation in data mining. We therefore proposed three principal constraints on representing as sets, they are bijection, isomorphism and powerset. We further presented that the sequences for mining sequential patterns cannot be represented as sets.

Next, we presented that the sequences defined for mining sequential patterns cannot be represented as sets, and therefore we proposed the ordered pattern model with respect to the constraints on representing as sets. Thus the transversal hypergraph computation can be used in discovering ordered patterns. We further showed that the sequence can be represented by ordered patterns, and the problem of mining sequential patterns sequence can therefore be transformed to the problem of discovering frequent ordered patterns.

With the model of ordered patterns, we finally showed that transversal hypergraph computation can be used in mining sequential patterns by the *Dualize and Advance* algorithm.

## Perspectives

The approach proposed in this Master thesis can be named *A-discovery* of ordered patterns because the description language  $\mathcal{L}_A$  is based on attributes.

We propose the *P-discovery* for future work, which is based on the *P-form* representation of ordered patterns (cf. Section 8.4).

Given a database  $\mathcal{D}$  of customer transactions on attributes  $R$ , if the size of frequent itemsets is quite small related to the size of all subsets of  $R$ , then *A-discovery* is not optimal for mining sequential patterns. Thus we consider the *P-discovery* for this case.

The *P-discovery* first computes all frequent itemsets in database  $\mathcal{D}$ . With the *P-form* of the ordered patterns generated from all frequent itemset, the task of mining sequential patterns can be optimal.

To further reduce the search space required by the *P-discovery*, we can remove frequent pattern that exists only in its specialization. Therefore all sentences representing a set of ordered patterns with the same order can be ignored during evaluations of the quality predicate.

We are also interested in measuring the efficiency of the *A-discovery* and the *P-discovery* for mining sequential patterns in different cases.

# Bibliography

- [1] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In *SIGMOD Conference*, pages 207–216, 1993.
- [2] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, pages 487–499, 1994.
- [3] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *ICDE*, pages 3–14, 1995.
- [4] Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yiu. Sequential pattern mining using a bitmap representation. In *KDD*, pages 429–435, 2002.
- [5] Thomas Eiter and Georg Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM J. Comput.*, 24(6):1278–1304, 1995.
- [6] Thomas Eiter and Georg Gottlob. Hypergraph transversal computation and related problems in logic and ai. In *JELIA*, pages 549–564, 2002.
- [7] FIMI. The “frequent itemset mining implementations repository” web site. <http://fimi.cs.helsinki.fi/>, 2003.
- [8] Michael L. Fredman and Leonid Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *J. Algorithms*, 21(3):618–628, 1996.
- [9] Dimitrios Gunopulos, Roni Khardon, Heikki Mannila, Sanjeev Saluja, Hannu Toivonen, and Ram Sewak Sharm. Discovering all most specific sentences. *ACM Trans. Database Syst.*, 28(2):140–174, 2003.
- [10] Dimitrios Gunopulos, Roni Khardon, Heikki Mannila, and Hannu Toivonen. Data mining, hypergraph transversals, and machine learning. In *PODS*, pages 209–216, 1997.

- [11] Dimitris J. Kavvadias and Elias C. Stavropoulos. Evaluation of an algorithm for the transversal hypergraph problem. In *Algorithm Engineering*, pages 72–84, 1999.
- [12] Heikki Mannila and Hannu Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Min. Knowl. Discov.*, 1(3):241–258, 1997.
- [13] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Efficient algorithms for discovering association rules. In *KDD Workshop*, pages 181–192, 1994.
- [14] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Min. Knowl. Discov.*, 1(3):259–289, 1997.
- [15] Florent Masegla, Pascal Poncelet, and Maguelonne Teisseire. Incremental mining of sequential patterns in large databases. *Data Knowl. Eng.*, 46(1):97–121, 2003.
- [16] Florent Masegla, Maguelonne Teisseire, and Pascal Poncelet. Extraction de motifs séquentiels. problèmes et méthodes. *Ingénierie des Systèmes d'Information*, 9(3-4):183–210, 2004.
- [17] Srinivasan Parthasarathy, Mohammed Javeed Zaki, Mitsunori Ogihara, and Sandhya Dwarkadas. Incremental and interactive sequence mining. In *CIKM*, pages 251–258, 1999.
- [18] Marc Plantevit, Yeow Wei Choong, Anne Laurent, Dominique Laurent, and Maguelonne Teisseire. M<sup>2</sup>sp: Mining sequential patterns among several dimensions. In *PKDD*, pages 205–216, 2005.
- [19] Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *EDBT*, pages 3–17, 1996.
- [20] Takeaki Uno and Ken Satoh. Detailed description of an algorithm for enumeration of maximal frequent sets with irredundant dualization. In *FIMI*, 2003.
- [21] Jianyong Wang and Jiawei Han. Bide: Efficient mining of frequent closed sequences. In *ICDE*, pages 79–90, 2004.