École doctorale IAEM Lorraine

Analyse formelle de concepts et structures de patrons pour la fouille de données structurées

THÈSE

présentée et soutenue publiquement le 6 Octobre 2015

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Aleksey BUZMAKOV

Composition du jury

Présidents :	Pr. Bruno CRÉMILLEUX — Université de Caen
Rapporteurs :	Pr. Jean-François BOULICAUT — Université de Lyon
	Pr. Arno SIEBES — Universiteit Utrecht (Pays-Bas)
Examinateurs :	Pr. Bernhard GANTER — Technische Universität Dresden (Allemagne)
	Pr. Sergei O. KUZNETSOV — NRU Higher School of Economics (Moscou – Russie)
	Dr. Amedeo NAPOLI — Directeur de recherche CNRS
	Dr. Henry SOLDANO — Institut Galilée – Université Paris-Nord

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Remerciements

First of all, I would like to thank my supervisors Sergei Kuznetsov and Amedeo Napoli who proposed me to do this thesis and shared their experience all over this journey; I thank them for nice discussions, for sharing ideas, for showing future directions, for reading my texts, which were sometimes completely incomprehensible, especially at the beginning of my work. I also thank them for pleasant time spent together in non-so-scientific environments.

I want to thank the people who kindly accepted being members of my defense committee. I thank Bruno Crémilleux for being the president of the committee. I thank very much Jean-François Boulicaut and Arno Siebes for accepting to be reviewers of my manuscript; for evaluating my manuscript and expressing great interest in my work. I also thank Bernhard Ganter and Henry Soldano for a very good feedback on my work during the defense. I want to thank all members of the committee for asking deep motivating questions, even though they were not easy ones. These questions and discussion showed that a thesis defense can be an exciting event.

I thank the BioIntelligence project for financing the first 3 years of my thesis and the University of Lorraine for financing the last year of the thesis.

I want to thank the team in Caen we have collaborated with, especially Bertrand Cuissart for making this collaboration possible.

I thank the Orpailleur team and the Loria/Inria laboratories for their hospitality, for providing opportunity for efficient work and discussions with a lot of people. I thank Chedy Raïsi for sharing scientific experience with me and very positive communication. I thank Miguel Couceiro for nice discussions around math and giving me deeper knowledge of mathematical terminology. I thank Yannick Toussaint for a plenty of scientific discussions we had. I thank Bernard Maigret for all discussions around chemoinformatics and for sharing strong motivation for solving different chemical and biological problems. I thank other PhD students Mehwish Alam, Victor Codocédo, Arseniy Gorin, Sébastien Da Silva, Elias Egho, Mohsen Sayed, Julien Stevenot, Thao M. Tang for sharing many different parts of my PhD life within and outside of science. I thank other members of Orpailleur team: Yasmine Asses, Thomas Bourquard, Jeremie Bourseau, Sami Ghadfi, Laura Infante-Blanco, Thi Nhu Nguyen Le, Olfa Makkaoui, Luis-Felipe Melo-Mora, Matthieu Osmuk, Nicolas and Alice Pépin-Hermann, Mickael Zehren. It was a pleasure to know you and have fun with you. Most of the people from the lab were always ready to help and I thank them for providing me a very precious help when I needed it. Finally I thank Emmanuelle Deschamps who helped a lot with all formalities all around the thesis.

I want to thank especially my wife Sofya Kulikova and my mother Svetlana Buzmakova who strongly supported me all this time and I thank my son Vladimir for making our evenings full of fun. Finally, I want to thank all those people that I unfortunately forgot to mention.

To those I love, especially
to my mother, my wife, and my son.

Contents

Résumé en Français

Aujourd'hui de plus en plus les données de différents types sont accessibles. Ces données variées contiennent des informations précieuses qui peuvent aider à resoudre les problèmes pratiques ou servir à la science fondamentale. *Mais comment peut-on extraire cette information précieuse ?* Ce processus d'extraction s'appelle fouille de données (data mining). Pour l'extraction on doit toujours penser quelle pièce d'information est importante ou si cette pièce est une artefact de cet ensembles de données. Tous d'abord on défini une langue pour exprimer ces pièces d'information qui s'appellent des patrons. Cette langue doit être capable de exprimer toutes les pièces intéressantes que on veut trouver. Une patron est valide pour une partie de la base de données. Tel patron peut être considéré comme une modèle de cette partie-là. Les patrons sont triés par rapport a la *a priori* taille de la base de données couvert, i.e., parmi des patrons il y a patrons plus générals et patron plus spécifiques.

La langue de patrons exprime une grand ensemble de patrons possibles. Donc il est nécessaire d'appliquer des contraintes qui sont typiquement basés sur une base (ou ensemble) de données. La base de données peut être considérer comme une mapping entre les objets (ou transactions) et leur descriptions typiquement exprimé dans la même langue que le patrons à chercher. Le premier contraint utilisé est la fréquence. En fait pour chaque patron on peut associer la fréquence, le nombre d'objets (le volume de la base de données) qui sont couverts par le patron. Et après on n'est pas intéressé que au patrons avec le grand fréquence par rapport d'un seuil. Apart de ce contraint il y a beaucoup de contraints étudies.

Chaque nouveau langue de patrons ou contraint lance une cherche pour des approches efficaces a trouver le patron pour cette langue et cette contrainte. Dans la thèse on travaille avec des problèmes de cette type-là pour les langues exprimants les données plutôt avec une structure complexe comme des graphes ou séquences.

Si on veut classifier des approches existants, on peut sélectionner trois critères qui sont appliqués au grand variété des approches.

Le langue de pattrons utilisé par l'approche. C'est la plus important critère comme il montre la limitation d'utilisation de l'approche pour le données on veut analyser.

Les contraintes dont l'approche travaille avec. Normalement les contraintes sont données comme le classe de contraintes avec de propriétés spécifiques qui sont utilisés par l'approche.

Le type d'algorithme utilisé pour trouver le patrons. Néanmoins on peut imaginé beaucoup de type d'algorithmes, la plupart de ces types ne peut être applique que au quelques approches. Les plus grand types sont:

- **les algorithmes completes** trouvent un ensemble complet des patrons sélectionnés par le contrainte. C'est la plus grands type des approche la plupart de quelle sont assez lent car le nombre de patrons à parcourir est grand.

les algorithmes indéterministes trouvent un sous-ensemble des patrons sélectionnés par le contrainte en base d'une procédure aléatoire. Les propriétés des cette procédure garanti que chaque patron peut être générer avec la même probabilité. Normalement il sont plus lents par rapport d'un patron trouvé mais par contre ils sont créer pour créer un sous-échantillon des patrons cherchés de taille assez petit qu'ils peuvent faire très vite.

les algorithmes heuristique trouvent un sous-ensemble des patrons cherchés qui est proche au ensemble complet. Ces approches garantissent ce propriété d'être proche au l'ensemble demandé. Normalement ces approche sont plus vites que les algorithmes complètes.

Dans la thèse on révise des approchés différents et les trie par rapport de critères ci-dessus. Le majorité des approches connus ne sont capables de travailler que avec une langue de patrons. L'Analyse Formelle de Concepts (AFC) et les pattern structures sont des systèmes formels et permettent de traiter les données ayant une structure complexe et différente. Cette généralité de l'AFC est la justification d'utilisation de cette systèmes formelles spécifiques dans la thèse.

L'AFC d'une base de données, i.e., le correspondance entre des objets (des transactions) de la base et leurs descriptions, trouve les concepts formels, i.e., les pièces élémentaires d'information. Les concepts correspondent aux patrons avec une information supplémentaire qui code la partie de la base de données couverte par le patron. *Mais comment peut-on les appliquer en pratique ?* De plus, le nombre de concepts trouvé par l'AFC est fréquemment très grand.

Pour faire face à ce problème, on peut simplifier la représentation des données, soit par projections de pattern structures, soit par introduction de contraintes pour sélectionner les concepts les plus pertinents. *Quelle est la meilleure simplification de données? Comment peut-on efficacement trouver les concepts satisfaisants une contrainte donnée?* Ce sont les questions que nous abordons dans ce thèse.

Le manuscrit commence avec l'application de l'AFC à l'exploration des pièces importantes d'informations à partir de structures moléculaires. Ces structures moléculaires sont codés comme les ensembles d'attributs. Chaque attribut est un souspartie important des molecules. Par exemple, les groupes fonctionnels chimiques sont parmi les attributs. Ces sousparties sont trouvés par la fouille de graphes moléculaires. Même pour ce codage simple et sans contraintes supplémentaires, l'AFC est capable d'extraire des informations importantes dans de petits ensembles de données. En fait, l'AFC trouve tel ensembles d'attributs quels ne peuvent pas être étendu par des autre attributs sans changement dans l'ensemble de molécules qui supportent l'ensemble d'attributs considéres.

Par exemple, on a appliqué cette approche-là pour trouver les ensembles de groupes fonctionnelles responsables pour l'inhibition du protéine c-Met. C'est une tache de classification dirigée, i.e., pour chaque objet de la base de donnée il y a une classe associée. Le classe dit quelle type d'inhibition du protéine c-Met chaque molécule a. Donc parmi tous les concept formelles on trouve tel concepts qui sont associe avec des objet ayant la même classe. Un patron assez simple trouvé par cette procédure semble d'être une bonne caractéristique pour une classe spécifique d'inhibition. En plus, les patrons trouves par cette procédure peuvent être utilisé comme les nouveaux attributs dont améliorent le regroupement (clustering) de molécules.

Avec l'augmentation de la taille des bases de données, les bonnes contraintes deviennent essentielles. Pour cela on explore la stabilité d'un concept. La stabilité est une contrainte formelle bien-fondé qui correspond au probabilité de trouver le patron si la base de données est modifiée. En particulier, chaque base de données avec l'enlèvement d'un ensemble des objets est considérée comme la sousbase de données. Donc la stabilité est proportionelle au nombre de sousbases de données dont le patron est trouvé. On montre expérimentalement que c'est un bon choix. Pour cela on divise la base de donnée en deux parties de taille égal et compare la stabilité pour les concepts trouvées dans ces deux basés de donnée. On a trouvé que la stabilité a le comportement similaire sur des bases des données generes a partir de la même distribution générale.

Après on a compare la stabilité avec des concurrents par rapport de son comportement dans la tache de classification dirigée. Pour cela, on prend des bases de donnée et supprime l'association entre les objets et les classes. On trouve le patron les plus intéressant par rapport de contrainte diffèrent (la stabilité et ses concurrents) et juste après on utilise l'association entre les objets et les classe pour trouver les patrons associés à la classification. C'est nécessaire pour éviter le préjugé au tache de classification, parce que pendant la computation des contraintes l'association entre les objets et les classes est cachée. La stabilité est trouvée d'être efficace dans cette procédure. L'autre approche qui a l'efficacité similaire est l'influence (leverage) mais ce mesure-là ne peut être utilise que pour la langue la plus simple, le lange des ensemble des attributs. Donc la stabilité est un contrainte préférable a utilisé.

A la fin de la première partie de la thèse, on applique l'AFC et la stabilité à l'exploration d'un ensemble de données de substances chimiques mutagènes. Donc on a des substance chimiques avec son graphe moléculaire et deux classes, le substances mutagènes et non-mutagènes. Le recherche de concepts stables associés avec les mutagènes dans cet ensemble de données a permis de trouver les nouveaux groupes fonctionnelles mutagènes possibles qui peuvent être interprétés par les chimistes et utilisés pour les détection et création de molécules mutagènes.

Cependant, dans les cas plus complexes, la représentation simple de l'attribut des données n'est pas suffisant. En conséquence, on se tourne vers les pattern structures qui peuvent traiter différents types de descriptions. Le point important sur les pattern structures est qu'elles permettent la simplification des données au moyen de projections. On étend le formalisme original de projections pour avoir plus de liberté dans la simplification de données. En particulier, on permit d'enlever les petits patrons sans enlèvement de patrons correspondants au objets de la base de données. On a aussi montré comment cette changement de l'ensemble de patrons possibles change la représentation de la base de données. On a trouvé aussi que la stabilité a un comportement très intéressant sur lequel la troisième partie de la thèse est basée. En particulier, la stabilité ne peut pas augmenter son valeur après l'application d'une projection. Cela a dire que si on applique une projection les patrons stables vont être trouvé même sous la projection.

On montre que cette extension est essentiel pour analyser les trajectoires des patients, décrivant l'historique de l'hospitalisation des patients. En effet, les trajectoires des patients sont les séquences d'hospitalisations et chaque hospitalisation est décrite par une description hétérogène qui explique le motif d'hospitalisation, le location de l'hôpital, les procédures médicaux appliqué au patient. On modèle chaque trajectoire comme un séquence basé sur les éléments structurés dans la forme de un semi-treuil. Ces données sont très riches et donc produisent un grand nombre de concepts. Pour analyser ces donnés on applique des projections de nouveau type pendant les computations, qui permet une réduction efficace de la notion d'espace. En plus, en combinaison avec des contraintes de stabilité on peut trouver des trajectoires communes importantes que peuvent être passé aux experts.

En outre, les projections sont utiles pour corriger les données ouvertes et liées (linked open data) où les erreurs sont inévitables. On peut efficacement trouver certaines erreurs par notre approche basée sur les pattern structures. En particulier, on travail avec les données généré à partir de Wikipedia. Il y a deux types de relations qui on peux y trouver. Le premier type de relations correspond au système de types de données ouvertes et liées. Le deuxième type correspond au folksonomy, un taxonomie créer par les utilisateur de Wikipedia grâce au système de tags. Ce deux types de relations sont indépendants et c'est probable que il y a des erreurs parmi de l'information associé aux relation de deux ces types-là. Par contre il y a l'information partagé qui peut être utiliser pour associer des relation de types diffèrent et corriger l'erreurs. Cela est fait par une pattern structure hétérogène. Chaque description est divisée en deux partie correspondants aux deux types. Grâce a nos projection on peut enlever beaucoup de patrons et parmi l'autre on trouve l'associations fortes entre les deux parties de la description. On considère que cela est une justification forte que il y a une problème dans les relation et suggère les corrections.

L'autre application de pattern structures est la recherche d'interactions médicamenteuses dans un corpus du texte. On est capable d'extraire et d'expliquer les structures syntaxiques codant ce genre des relations. En fait, chaque sentence est codé par son arbre syntaxique. On définit une opération de similarité entre des arbres de cette forme-là qui nous permit d'utiliser les pattern structures pour l'analyse de cette base de données. Comme l'opération de similarité est assez compliqué à computer on utilise une projection qui transforme une arbre syntaxique au l'ensemble de chemin commencé au racine. Après on nous étudions sur une partie de la base de données et sont capable de prévoire les relations dans l'autre partie de la base.

L'approches présentées jusque là ne permettent pas la découverte directe de patrons sous la contrainte de stabilité. En conséquence, le manuscrit se termine par une approche originale et très efficace qui permet de trouver directement des patron stables. Cette approche est appelée $\Sigma o \varphi \iota \alpha$ (cela a dire Sofia, Searching for optimal formal intents) et est capable de trouver les meilleurs patrons stables en temps polynomiale. L'efficacité est essentielle pour l'analyse de grands ensembles de données et cela souligne l'importance de $\Sigma o \varphi \iota \alpha$. L'algorithme est basé sur les conséquents augmentions de la détalisation de la base de données. Par example, on commence par la base de donnée vide, i.e., elle ne contient pas des attributs. Cela à dire, que il n'y a pas déférence dans les descriptions des objets différentes. Dans la prochaine itération on ajoute la premier attribut dans la base de données. Après on ajoute le deuxième, troisième, etc. Avec chaque augmentation de la détalisation de la base de données on compute les patrons stables correspondants à la base de données à cette niveau de la détalisation au façon incremental. Cette procédure-là peut être formalisé comme une chaîne de projections. Quand une projection est appliquée au patron x, le résultat montre de quelle patrons le patron x est généré pendant l'augmentation de la détalisation de la base de données.

$\Sigma o \varphi \iota \alpha$ permit nous travailler avec une type de contraints appelées *antimonotonic par rapport de projections* parmi lesquels il y la stabilité. Ils sont défini par rapport de une chaîne de projections comme les contraints ayant la valeur plus grand pour une patron x que pour les antécédents du patron x par rapport de chaque projection de la chaîne.

Si on a une chaîne de projections et une contraint antimonotonic par rapport de cette chaîne, alors on peux trouver les patrons intéressants par rapport de cette contrainte-là dans la manière prochaine. Chaque fois quand on change la détalisation de la base de données on ne trouve les antécédents que pour le patrons intéressants trouvés sur l'étape précédant. Après on trie les antécédents pour préserver que le patrons intéressants. Grâce à antimonotonicity par rapport de la chaîne on peux être sure que on trouvera tous les patrons intéressants sans considération d'une grand partie des patrons inutiles. En plus, si on limite le nombrer de patrons que on peut sauvegarder après chaque passage au meilleur détalisation de la base de données par la modification correspondante de la seuil, $\Sigma o \varphi \iota \alpha$ devient l'algorithme polynomial.

On évalue ce nouvel algorithme sur des ensembles de données de types binaires et numériques. Les expériences montrent l'amélioration significative de $\Sigma o \varphi \iota \alpha$ par rapport à ses concurrents pour les données des attributs et des n-uplets d'intervalles. En plus le concurrents étaient mis en conditions trés profitables pour les concurrents $\Sigma o \varphi \iota \alpha$ est toujours mielleux. En outre, $\Sigma o \varphi \iota \alpha$

ouvre une nouvelle direction de recherche pour l'exploitation de différents types de patron en temps polynomial qui est très important dans le monde des mégadonnées.

Pour conclure, on remarque que on a développé et appliqué les méthodes efficaces pour analyser des données complexes. Leur application a permit de trouver des patrons intéressants étaient trouvés pour des domaines. Finalement, il y a beaucoup des directions à travailler à améliore l'efficacité de méthodes, l'interprétabilité des résultats et la couverture des langues de patrons.

General Introduction

Nowadays, more and more data of different kinds are available. These various datasets hide valuable information that may help to solve some practical problems or to support some advances in fundamental science. *But how can one extract these precious pieces of information?* Such extraction process is typically called knowledge discovery. Working on this question one faces a problem of what is interesting, what is not and what is an artifact of a dataset. First of all, one defines a language that should be able to express interesting pieces of information that are words in that language and are called patterns. A pattern can be matched to a part of the dataset, i.e., it can be considered as a model of this part of data.

Pattern languages are typically general and able to describe a huge set of patterns. Examples of such languages are itemsets (Agrawal et al. 1993a), sequences (Agrawal and Srikant 1995), graphs (Kuznetsov 1999; Yan and Han 2002), *etc.* Since the set of possible patterns expressed by such languages is huge, it is necessary to define a constraint that would point out the most important patterns. One of the first constraints was on pattern frequency, i.e., the number of records in a dataset where the pattern is contained. A huge number of other constraints was suggested and studied since then.

Every new pattern language or a new constraint raise a problem of how to efficiently mine patterns within the language satisfying the constraint. Many efficient approaches were suggested in order to deal with certain languages or constraints. We will discuss some of these algorithms in conjunction with the corresponding pattern languages and constraints hereafter in Section 1.

In this work we are particularly interested in pattern languages that express structured data. By structured data we mean data that are represented as sequences or graphs in contrast to itemset representation. From such kind of data we are interested in extracting structured patterns. Although a pattern language and a data language are not necessarily similar, throughout this work we consider them to be similar. In many cases it is natural to consider them similar and this allows us to rely on pattern structures (Ganter and Kuznetsov 2001), a rich formalism for processing different kinds of data that is based on Formal Concept Analysis (Ganter and Wille 1999). We discuss contributions of this thesis and the structure of the manuscript in details in Section 2 after the overview of data mining.

1 Data Mining and Knowledge Discovery

Following Fayyad et al. (1996) we would say that data mining is a step within a process of knowledge discovery in databases (KDD). Fayyad et al. (1996) list the following steps of the knowledge discovery process:

Data selection is a step where, given a goal of knowledge discovery, one finds the appropriate data and forms a dataset. This step is mostly done manually and involves domain experts.

Preprocessing is a step when one deals with noise and missing values in the dataset. In some cases noise and/or missing values can be treated in later steps, however in many cases the preprocessing is a required step.

Transformation step changes the dataset in order to be further processed by a data mining method. In other words, in this step we select data and pattern languages and express the data in terms of this languages. For example, a molecule can be expressed as a set of functional groups or as a molecular graph. In the first case the whole set of functional groups corresponds to items and a molecule is described by an itemset. In the second case a molecule is expressed as a mathematical graph. These two representations are suitable for different data mining methods, thus, having one in mind, we should modify the dataseta accordingly.

Data mining step is used for creating a model from the dataset in hand. Given a dataset described by some language it searches for patterns in the target languages satisfying a constraint and then based on these patterns a model can be created, e.g., the set of patterns by itself or a classification model.

Interpretation/Evaluation is a step when the found model is applied for analysis of the dataset. For example, a classification model can be evaluated on a test part of the dataset, while a set of patterns can be filtered and interpreted by a domain expert, e.g., in an exploratory analysis.

In this work we mainly discuss the data mining step of the KDD process. Although it is probably the most automated step, it is also very time consuming and diverse in he terms of languages it ought to work with. Below we discuss the most applicable pattern languages and the corresponding algorithms. In particular we discuss itemsets, sequences and graphs as pattern languages. It is not an exhaustive list of pattern languages, e.g., we do not discuss business intelligence models mined by process mining (Aalst 2011), but this set is a representative set of pattern languages. Furthermore, our goal is not to provide an exhaustive overview of literature but rather an overview of different techniques used for processing a certain type of data. In a big scale, data mining approaches can be divided based on the following criteria:

Pattern language provides the type of pattern an approach is working with. We divide the current section based on this category. We start from itemset mining, the most studied and probably the oldest pattern language, and then continue with sequential pattern and graph pattern minings in Sections 1.1–1.3.

Constraints give a binary predicate that selects interesting patterns from the possible patterns of the language. Many different methods rely on support, while some others constraints are discussed below for every type of patterns.

Algorithm type is a specialty of an algorithm in hand. In fact, it is hard to find a useful but general classification of the algorithms. So below we discuss the following three main algorithm types:

Full algorithms ensure finding the whole set of patterns for a dataset, given a pattern language and constraints. This type of algorithms is the largest in the number of algorithms and we will not specially highlight that an algorithm belongs to this class.

- **Nondeterministic** algorithms allow one to find a subset of patterns given a pattern language and constraints. The non-determinism allows one to develop efficient algorithms that find a sample of patterns with certain guaranties.
- **Heuristic** algorithms are deterministic algorithms that do not ensure finding the whole set of patterns given by pattern language and constraints, but provide a certain guaranty that the found set of patterns is close to the requested one. We explicitly highlight if an algorithm belongs to Nondeterministic or Heuristic types.

1.1 Itemset Mining and Basic Definitions

Itemset mining is one of the oldest and elaborated directions in data mining research. It was first introduced by Agrawal et al. (1993b) for basket data analysis in order to mine association rules. Their algorithms rely on *Apriori* principle that states that frequency is a anti-monotonic constraint and, thus, if one finds an infrequent itemset, any superset of it is also infrequent.

More formally, a *dataset* is a triple (G, M, I) where G is a set of objects, M is a set of attributes and $I \subseteq G \times M$ is a relation between them stating that an object $g \in G$ has an attribute $m \in M$ if and only if $(g, m) \in I$. It is not an original notation by Agrawal et al. but rather the notation of Formal Concept Analysis (Ganter and Wille 1999), which we discuss in a later subsection. We use this notation in order to be consistent throughout the manuscript.

Itemeset is any subset of attributes M. Given an itemset X, the *image* (also called *tidset*) of X is the set of all objects covering X, i.e., $\mathtt{Img}(X) = \{g \in G \mid (\forall x \in X)(g, x) \in I\}$. *Support* of an itemset X is the cardinality of the image of X, i.e., $\mathtt{Supp}(X) = |\mathtt{Img}(X)|$; *frequency* of X is the support of X normalized w.r.t. the dataset size, i.e., $\sigma(X) = \frac{\mathtt{Supp}(X)}{|G|}$. By analogy, we can define the image, the support and the frequency of any pattern, including sequential and graph patterns. Frequency can be used to construct a predicate for itemset selection. One can notice that any interesting itemset should be supported by a significant number of dataset objects, i.e., given a threshold θ, we can define a frequency constraint for pattern selection. In particular only patterns X satisfying $\sigma(X) > \theta$ are selected.

The frequency constraint is one of the most used constraints (in some cases in conjunction with other constraints). As it was noticed independently by Agrawal et al. (1993b) and Mannila, Hannu Toivonen, et al. (1994), frequency constraint satisfies the *Apriori* principle. In order to discuss it we should notice that we can define a partial order on itemsets (as well as on any patterns) w.r.t. subset inclusion. Then *Apriori* principle states that frequency constraint predicate is anti-monotonic w.r.t. this partial order, i.e., if $X \subseteq Y$, then $\sigma(X) \geqslant \sigma(Y)$.

Since then, many different approaches have been developed in order to efficiently find frequent itemsets, i.e., the itemsets satisfying the frequency constraint. For a wider review of the developed methods one can address some of the overviews (Aggarwal et al. 2014; Han, Cheng, et al. 2007). A naïve algorithm for mining frequent itemsets consists of a unique enumeration of all itemsets satisfying frequency constraint. In particular, we can fix a linear (total) order on attributes $<_M$. Then, given an itemset X, we can extend it by adding any attribute $m \in M$ such that $\forall \tilde{m} \in Xm >_m \tilde{m}$. It can be seen that such kind of extension enables unique enumeration of itemsets. Then the main challenge is to compute the frequency of a single itemset. Many different techniques were suggested, some of them can be found in Uno, Kiyomi, et al. (2005).

One of them is a so-called *conditional database*. The basic idea is the following. Given an itemset X, if an itemset $X \cup \{m\}$ is infrequent then for any $Y \supset X$ we have $Y \cup \{m\}$ is infrequent. Moreover, if $X \cup \{m\}$ has the same support as X, then it is valid for any $Y \supset X$. Both kind of attributes can be registered and used for optimization of computing support of all superitemsets of X. Some others are related to the way of representing and storing sets of objects. Indeed, from

Zaki and Gouda (2003) it is known that the "vertical" representation of a dataset enables efficient support counting. In contrast to the "horizontal" representation where every object is encoded as an itemset, in the vertical representation every attribute is encoded by the set of objects that support it. Hence, support of a pattern X can be computed by intersection of the sets of objects supporting single attributes. Thus, an efficient representation of sets of objects is essential for this kind of support counting. The bitset representation, i.e., where every object is represented by one bit of an array, is an efficient representation for dense datasets, while for a sparse dataset one can use a list of attributes that allows skipping huge number of consequent zeros in the bitset representation. Prefix tree is another way to store itemsets and to efficiently compute support of them. A common prefix of two itemsets is stored on the same path in the prefix tree, while on the first different attribute the tree is divided into two paths. There are some works that combine different representations in order to achieve a better performance. For example, Uno, Kiyomi, et al. (2005) combine prefix trees and bitsets by introducing `LCMv3` algorithm.

Since Pasquier et al. (1999) it is known that the set of all frequent itemsets is redundant. Indeed support of any frequent itemset can be easily derived from only closed frequent itemsets. A *closed* itemset X (as well as a closed pattern) is defined by means of support of X and its superitemsets. If support of any superitemset is smaller than support of X, then X is a closed itemset. For mining closed itemsets in addition to the aforementioned problem one should be more careful about the uniqueness of the enumerated itemsets. Many algorithms were introduced to tackle exactly this problem. Already Norris (1978) suggested a first solution for this problem. Later several algorithms were introduced by the community of Formal Concept Analysis (FCA) (Ganter 1984; Kuznetsov 1993; Merwe et al. 2004; Nourine and Raynaud 2002) (we discuss FCA in a later subsection in more details). In data mining community there was an independent search for closed itemset algorithms (Burdick et al. 2001; Grahne and J. Zhu 2003; Uno, Asai, et al. 2004; Zaki and Hsiao 2005). In particular, the well-known LCM algorithm (Uno, Asai, et al. 2004) is a reformulation of `Close by One` algorithm (Kuznetsov 1993) introduced much earlier in the FCA community. Let us consider how they solve the problem of uniqueness of mined closed itemsets. In contrast to frequent itemset mining, an itemset X has two types of attributes, i.e., the ones that were added by an extension of a smaller itemset Y ($X \supset Y \cup \{m\}$) and the ones that are added by a closure operation. A closure operation converts an itemset X to the maximal superitemest having the same support. Thus, if we extend an itemset X only by larger attributes (in the sense of naïve algorithm for itemset mining) than the attributes of the first group, we are able to uniquely enumerate the closed itemsets.

Frequent closed itemsets allow for finding the support of any frequent itemset. Sets of itemsets with this property are called *condensed representation* of all frequent itemsets and the set of closed itemsets for a given dataset is not the smallest condensed representation. For example, non-derivable itemsets are closed itemsets of special kind (Calders and Goethals 2002). It can be shown that support of some closed itemsets can be derived from the support of other closed itemsets and, thus, the preservation of such itemsets is not necessary for finding support of any frequent itemsets. Other kinds of condensed representation are discussed in (Calders, Rigotti, et al. 2006).

Frequency and closedness are basic constraints for many kinds of patterns. Let us now discuss some of the other constraints. One can be interested in finding *maximal frequent* itemsets, i.e., the frequent itemsets that have no frequent superitemsets. For example, MAFIA (Burdick et al. 2001) and GenMax (Gouda and Zaki 2005) are approaches searching for maximal itemsets. MAFIA relies on a depth-first search strategy to traverse the space of itemsets. In addition it uses several branch cutting techniques that significantly increase its efficiency for mining maximal itemsets, e.g., if an itemset X is a subset of an already found maximal itemset, then it is frequent and, thus, there is no need to find the support of X.

One can also be interested in the counterpart of closed itemsets, i.e., minimal generators. A *minimal generator* X is an itemset such that any subset of it has different support. Minimal generators are the smallest itemsets of a class of equivalence given by the same image (closure), while a closed itemset is the maximal element of its equivalence class. Accordingly, the set of minimal generators is also a condensed representation of all frequent itemsets.

The constraint based on minimal generators is a monotonic constraint, i.e., any subset of a minimal generator is necessary a minimal generator. Thus, the main challenge of mining minimal generators is the order of attributes in which they are added to an itemset in order to traverse the search space and generate as less nongenerators as possible (Szathmary, Valtchev, Napoli, and Godin 2009; Szathmary, Valtchev, Napoli, Godin, et al. 2014). The further development of the idea of minimal generators is δ-free itemsets (Boulicaut et al. 2000; Hébert and Crémilleux 2005). The support of a δ-free itemset X should be not only different from the support of its subsets but it should be smaller at least by δ than support of any subset. Such kind of itemsets plays an important role in mining *association rules*, i.e., the rules of the form $X \to Y$, where X and Y are itemsets and the set of objects including Y is close to the set of objects including X.

Many other constraints have been introduced for mining itemsets. It can be stability (Kuznetsov 1990, 2007) and robustness (Tatti et al. 2014) that measure a probability of an itemset to have certain properties (like closedness) under removal of some objects. The margin closedness is another approach introduced on top of some other constraints: given a set of itemsets (probably already filtered by a constraint) the margin closedness constraint selects the itemsets that have support which is significantly different from support of any superpattern (Moerchen et al. 2011). Thus, margin-closedness can be considered as a counterpart of δ-freeness.

The number of constraints that are introduced in order to find more relevant itemsets is really huge. For a more detailed review one can address Vreeken and Tatti (2014). Since there are a lot of constraints, a unified framework is needed. Accessible systems (Boley et al. 2010) are a way for such kind of unification. A system is defined as a pair $(\mathcal{F}, M)$, where M is the set of attributes, and $\mathcal{F} \subseteq 2^M$ an arbitrary set of itemsets. This set $\mathcal{F}$ can model a constraint on the set of itemsets. Boley et al. (2010) introduce a way for efficient finding of closed itemsets in the case where a system is accessible, i.e., if $\forall X \in \mathcal{F}, X \neq \varnothing$ there is an attribute $m \in X$ such that $X \backslash \{m\} \in \mathcal{F}$. For dealing with new constraints, Soulet and Crémilleux (2005) introduced a primitive-based framework. It allows combining (anti-)monotone primitives into a new constraint by certain operations. Another framework for dealing with new constraints is Soulet and Crémilleux (2008). The authors suggest to introduce a new closure operator adequate to the constraint. Accordingly, if the number of adequate-closed patterns is small, than the patterns w.r.t. to the constraint can be mined efficiently.

Another way to deal with the variety of constraints is to introduce a framework that allows for constraint combination. Soulet, Raïssi, et al. (2011) suggest an algorithm for finding itemsets that dominate any other itemset w.r.t. at least one of the constraints in question. Another way for an efficient combining of different constraints is constraint programming. Guns et al. (2011a) showed how one can encode various constraints of itemsets in terms of constraint programming. This approach is very efficient when the number of involved constraints is large.

One of the important questions of itemset mining is how one can limit the number of found itemsets. If there is a quality measure of an itemset (that can be associated to a constraint by introducing the threshold of a goodness for an itemset w.r.t. the measure), then one can be interested in finding top-K itemsets with the highest values of the measure. One of the first approaches for finding top frequent closed itemsets is described in Han, Wang, et al. (2002). Later the top-K search was adapted to association rules discovery (Webb and S. Zhang 2005), to mining nonredundant top-K patterns (Xin et al. 2006), to constraint programming for combining constraints (Guns et al. 2011b).

Another approach for limiting the number of result itemsets is to find a representation set of itemsets. A representation set of itemsets is often called a summary of a dataset. There is a number of different measures for the quality of a summary, most of them are related to the diversity of the itemsets in the summary and to the coverage of this summary w.r.t. the dataset (Zbidi et al. 2006). Vreeken, M. v. Leeuwen, et al. (2011) introduce `KRIMP` algorithm that finds a summary of a dataset covering the whole dataset. This approach is based on minimal description length (MDL) principle and looks for a set of itemsets that compresses the dataset in the best way. Because of MDL the redundant itemsets cannot be found in the same summary and thus it provides a good summary of the dataset.

Although the overview given above is not exhaustive, we hope that most of the directions of itemset mining research are considered there. More information can be found in special review literature such as Aggarwal et al. (2014), Geng and Hamilton (2006), Hilderman and Hamilton (1999), Masood and Soong (2013), and McGarry (2005). Let us now switch to more complex types of data such as sequences or graphs.

1.2 Sequential Mining

A sequential dataset can be formalized in a similar way as a binary dataset, i.e., it is a triple $(G, (S, \leqslant), \delta)$, where G is a set of objects, also called transactions, S is a set of sequences ordered by a subsequence relation and $\delta : G \to L$ is a mapping from objects to the corresponding sequence. Several pattern languages are related to sequential data mining. In all cases *Sequential pattern*, i.e., the element of S, is an ordered list of elements from an alphabet. However, a partial order can be given on the alphabet. This partial order affects the partial order of sequential patterns. If there is no order, then an element can be matched to itself only when checking the subsequence relation. Otherwise, an element can be matched also to any "larger" element of the alphabet. For example, let us consider the alphabet $(2^M, \subseteq)$, one of the most studied and used alphabet. In this case every element is a subset of a given set of attributes M and a set can be matched against any superset, e.g., $\{a\}$ can be matched against $\{a, b\}$ and $\langle\{a\}, \{b\}\rangle$ is a subsequence of $\langle\{a, b\}, \{a, b\}\rangle$.

Agrawal and Srikant (1995) is one of the first work for mining sequential patterns with $(2^M, \subseteq)$ as the alphabet. Since this seminal work many approaches have been developed. There are several overviews that one could address (Mabroukeh and Ezeife 2010; Mooney and Roddick 2013). The comparison of the computational efficiency of them can be found in Kum et al. (2007). Here we briefly overview the main works.

Agrawal and Srikant (1995) solve the sequential data mining problem in several steps. First, they find frequent sets of attributes M describing a single element of the alphabet. Then, they encode every sequence based on the found frequent itemsets and find frequent sets of itemsets. These sets of itemsets correspond to frequent subsequences of the dataset. Finally only the largest frequent sequences are reported. Later Srikant and Agrawal (1996) introduced a generalization of this approach called GSP. GSP is the breadth-first search approach that – based on sequences of length k – constructs sequences of length $k + 1$ and filters out the duplicates. Already in this work some constraints on sequential patterns are introduced.

PSP algorithm (Masseglia et al. 1998b) improves the previous approaches by introducing an efficient tree-like structure for storing sequences. Just after PSP a new approach for dealing with user constraints appeared in Garofalakis et al. (1999). They explain how regular expressions can be embedded into the sequence mining framework.

Since vertical representation for itemset mining was shown to be very efficient (Zaki and Gouda 2003), there are works that adapt this technique for mining sequential data. Vertical representation for sequences is given by registering for all attributes from M the objects and the elements numbers in the object description where this attribute is found (Zaki 2001). SPADE algorithm and its modifications are based on vertical representation (Zaki 1998, 2000, 2001). In addition cSPADE allows for dealing with user constraints such as length limitation on sequences, width limitation on elements of the sequences, minimum and maximum gaps for the consecutive elements for subsequence relation and supervised classification task. Later SPADE was improved by introducing hashing in the algorithm (Orlando et al. 2004).

The efficiency of SPADE and its modifications is based not only on vertical representation, but also on the depth-first traversal of the search space. Accordingly, a big branch of the algorithms is constituted from the algorithm based on depth-first search. FP-Growth is one of the basic algorithms from this group (Han and Pei 2000). It is based on a monotonic extension of a sequence without switching to other sequences. FreeSpan (Han, Pei, et al. 2000) and PrefixSpan (Pei, Han, Mortazavi-Asl, H. Pinto, et al. 2001) are extensions of FP-Growth and they are based on prefix trees enabling very efficient pattern enumeration. Prefix-tree enables to efficiently restrict the database in hand to the objects covered by the current sequence. Based on these algorithms CloSpan (Yan, Han, and Afshar 2003) is introduced, which can be considered as a standard for mining sequences based on alphabet $(2^M, \subseteq)$. The main difference of CloSpan is that it finds closed rather than all frequent subsequences and as we know the set of closed frequent subsequences is a condensed representation of all frequent sequences. It relies on the observation that some branches of the search space contain only non-closed sequences, thus these branches can be safely pruned providing elimination of the most nonclosed sequences. Later, an approach for mining maximal sequences was introduced by Luo and Chung (2004). The authors rely on sampling for efficient finding of maximal sequences.

To overcome limitations of the alphabet $(2^M, \subseteq)$, Plantevit, Choong, et al. (2005) introduce multidimensional alphabet where an element is taken from the direct product of several simple alphabets. Later Plantevit, Laurent, et al. (2010) have extended this work to multilevel and multidimensional data. In addition to every single dimension one can also have a taxonomy showing the generality relation between attributes. Finally, Egho, Jay, et al. (2014) further develop this direction and enable for every dimension to be a set of attributes in addition to taxonomy. All these methods transform the task in hand to the task suitable for CloSpan by finding frequent elements of the alphabet and transforming them into single attribute from $(2^M, \subseteq)$ necessary for CloSpan.

1.3 Graph Mining

In graph mining, a graph dataset can be given either by a single large graph or by a triple $(G, \mathcal{G}, \delta)$, where G is the set of objects or graph names, $\mathcal{G}$ is the set of graphs and $\delta : G \to \mathcal{G}$ is a mapping from objects to their descriptions. There is a lot of pattern languages associated to a graph dataset. Below we discuss some of them. As in previous sections, we provide here a short overview of graph mining methods, for a more detailed overviews see Deshpande et al. (2005), Jiang et al. (2013), and Krishna et al. (2011).

If there is only a single graph as an input, then we can be interested in finding either subgraphs that can be found in many places within the graph (Cook and Holder 1994; Kuramochi and Karypis 2004, 2005) or in finding cliques or quasi-cliques (a dense graphs having nearly all possible edges for the given set of vertices) of the graph (Jianyong Wang et al. 2006; Zeng, Wang, Zhou, et al. 2006). Below we discuss the case when a graph dataset is given by a triple, since this setting is more close to the work discussed in this manuscript.

The first pattern language we are going to consider is a set of graphs. Thus, every pattern is a graph, and the patterns are ordered by means of a subgraph isomorphism relation $(\mathcal{G}, \leqslant)$. It is a well-known fact that subgraph isomorphism is a NP-hard task and, thus, efficient computations are extremely important for graph mining. One of the first work for mining frequent graphs is WARMR (King et al. 2001). It is based on inductive logic programming and has a lot of hidden isomorphism checking operations. Consequently, it is not efficient and is seldom used in practice. FSG (Kuramochi and Karypis 2001), MoFa (Borgelt and Berthold 2002), gSpan (Yan and Han 2002), and GASTON (Nijssen and Kok 2005) are the most used graph mining algorithms. It was shown that gSpan and GASTON are more efficient than the others in most of the datasets while the winner among them can be hardly determined (Wörlein et al. 2005).

gSpan starts from a single vertex as a frequent subgraph and then iteratively extends it to a larger patterns. It relies on a canonical representation of graphs and whenever it finds a duplicate it never continues extending it. GASTON uses a different canonical representation of graphs and different strategy of traversing the search space. It starts from a search for frequent pathes, i.e., a connected graph having at most two edges for any vertex. Then, based on found paths, it constructs frequent trees, i.e., a graph without cycles. Finally, starting from trees it adds cycles in order to find frequent graphs.

Since the number of frequent graphs is typically huge, one is interested in a compact representation of them. In particular a set of closed frequent graphs is one of such representations. CloseGraph (Yan and Han 2003) is an algorithm based on gSpan for mining closed frequent graphs. Although it finds significantly less patterns than gSpan, its computational efficiency remains comparable to gSpan. FOGGER (Zeng, Wang, J. Zhang, et al. 2009) is a complementary approach to CloseGraph. It finds minimal generator graphs, i.e., the minimal elements of equivalence classes of equally supported patterns.

Maximal graph mining is another direction for introducing constraints for result patterns. If graph pattern is maximal then any supergraph of it is infrequent. MARGIN (L. T. Thomas et al. 2010) and SPIN (Huan et al. 2004) are examples of this graph mining direction. In particular, SPIN following the strategy of GASTON firstly finds the set of frequent trees and then, based on algorithms for maximal itemset mining and on some branch cutting techniques, it finds maximal graphs.

In order to decrease the number of the result graphs some approaches incorporate user constraints (Papadopoulos et al. 2008; F. Zhu et al. 2007). For some of the constrains it is possible to apply them during the computation and thus to save the computational time. For example, Papadopoulos et al. (2008) rely on the number of vertices in a graph and on the connectivity of the graph, i.e., the minimal number of edges that should be removed in order to get a disconnected graph.

Other possibilities for saving computational time are given by heuristic or nondeterministic approaches. SUBDUE (Cook and Holder 1994), one of the first heuristic methods for graph mining, relies on minimal description length principle and on approximate checking of subgraph isomorphism. It enables very efficient computation but without insuring the completeness of the result. However the authors of this paper have shown experimentally that the found patterns are important from the practical point of view. Shelokar et al. (2013) extend SUBDUE algorithm for mining patterns w.r.t. to several constraints.

Another heuristic method is Leap Search (Yan, Cheng, et al. 2008). Based on previous traversed branches of the search space this algorithm is able to efficiently estimate frequencies of the graphs in current branch and, thus, efficiently avoid branches that are likely to be unpromising. GemsBond by Pennerath et al. (2010) is a greedy algorithm for search of graphs explaining chemical reactions. They find the smallest connectivity graph for a given pair of molecules maximizing correct classification of pairs of molecules.

Although heuristic methods enable efficient processing of large graph datasets, one could be interested in even further gain of efficiency. There is a bunch of nondeterministic algorithms that are based on a random search strategy and return a representation set of result graphs. ORIGAMI (Hasan, Chaoji, et al. 2007) is an approach for finding representation set of frequent graphs such that all graphs within the representation set are significantly different. It is based on random generation of graph patterns and is associated by a verification that a new graph pattern is different from the previous found ones. Later Hasan and Zaki (2009a,b) have shown that the probabilities of generating different frequent graphs in ORIGAMI are non-uniform. Accordingly, they introduce approach MUSK, which is based on Markov chains and which is able to generate a given number of graph patterns w.r.t. a uniform distribution over the whole set of graph patterns.

Another interesting nondeterministic approach is GAIA (Jin et al. 2010) based on a genetic algorithm. The authors limit the number of graphs that can "survive", i.e., be generated. And then based on the classification accuracy of graphs they define the survival function.

All the discussed approaches find patterns represented by a single graph. In some domains, e.g., chemical domains, a pattern represented by a set of graphs is more meaningful. Indeed, a molecule interacts with other molecules by means of one or several functional groups. Since a functional group is a graph, only sets of functional groups can explain interaction or certain properties of molecules. Poezevara et al. (2011) introduce an approach for mining such kind of patterns in order to find functional alerts in datasets of mutagenetic and nonmutagenetic molecules. It is a multistep approach that firstly finds frequent graphs by means of previously discussed methods and then it combines them into a multigraph pattern considering single graphs as attributes in a large binary dataset. A similar approach can be found in the domain of FCA (Blinova et al. 2003; Kuznetsov 1999; Kuznetsov and Samokhin 2005). Although Kuznetsov and Samokhin (2005) do not rely on frequent graphs, they find a set of graphs limited in terms of numbers of vertices but without taking into account the frequency of graphs and then they combine them into closed multigraph patterns by means of FCA.

Such kind of patterns requires more computational efforts than the approaches for mining single-graph patterns. Thus heuristic approaches are important here. Jin et al. (2009) introduce an approach for a greedy search for multi-graph patterns. They show that multigraph patterns consisting of small graphs are able to outperform single-graph patterns in supervised classification tasks.

Some other pattern languages based on graphs have been considered in order to increase efficiency of an approach or in order to grasp a specific features of the dataset. Schietgat et al. (2008) introduce an approach for mining only outer-planar graph, i.e., the graphs that can be drawn on a plane without intersections of edges and such that every vertex lies on the border of this drawing. This limitation to outer-planar graphs enables the author to construct a very efficient polynomial algorithm for mining frequent graphs.

Lozano et al. (2010) construct a heterogeneous pattern that contains graphs together with QSAR descriptiors, special descriptors for biologically meaningful fragments of chemical compounds (Devillers and Balaban 1999). This allows the authors to introduce domain knowledge to the pure graph representation. Another approach for introducing the domain knowledge into patterns is done by Sherhod, Gillet, et al. (2012). A pattern in this work is accompanied by the number of π-bounds and the distance between pairs of atoms.

Maunz et al. (2010) introduce latent structures, a special kind of patterns that combine different frequent graphs. For every edge of a graph, Maunz et al. (2010) add a special weight that characterizes the frequency of this edge in the database w.r.t. to the rest of the graph.

1.4 Formal Concept Analysis

Lattice theory is shown to be useful for many machine learning and data mining tasks (Barbut and Monjardet 1970; Boldyrev 1974; Najdenova 1963; Oosthuizen 1988; Plotkin 1970). Correspondingly, formal concept analysis (FCA) is created as a proxy from pure lattice theory to applications such as machine learning and data mining (Ganter and Wille 1999). It makes a correspondence between a formal context, a triple (G, M, I) where G is the set of objects, M is the set of attributes and $I \subseteq G \times M$ is a binary relation between them, and a lattice of formal concepts, where a formal concept is given as a pair of corresponding sets of objects and attributes. Formal concept analysis can be considered as one of the best formalization of data mining field. A wide range of papers related to FCA can be found in Kuznetsov and Poelmans (2013), Poelmans, Ignatov, et al. (2013), and Poelmans, Kuznetsov, et al. (2013). The first two papers review the models that are based on FCA, while the third one reviews the applications.

The notion of a formal concept can be relaxed for dealing with noisy binary data (Besson, Pensa, et al. 2006; Besson, Robardet, et al. 2005). In particular, one can allow for objects in the extent of a concept that contain mostly all attributes from the intent. Accordingly, every attribute from the intent is included in most of objects from the extent.

Originally, FCA was introduced to deal with binary data. Later, it was extended for any kind of descriptions forming a semilattice. Ganter and Kuznetsov (2001) introduced pattern structures for dealing with such kind of data. In this case a data set is called pattern structure and is given by a triple $(G, (D, \sqcap), \delta)$, where G is a set of objects, $(D, \sqcap)$ is a semilattice of descriptions with $\sqcap$ as a similarity operation between descriptions, and δ is a mapping from objects to their descriptions. They have also introduced projections of pattern structures, special functions $\psi : D \to D$ that allow for simplification of semilattce and the practical computational efforts. It was also shown that any pattern structure can be represented by a formal context. However, the number of attributes in this context can be huge.

There are several practical studies of pattern structures for different types of descriptions. Kuznetsov and Samokhin (2005) study semilattice of graphs that was computed based on the representation context of pattern structures. Later pattern structures were used for mining interval tuple data, i.e., the data where every object is described by a list of numerical intervals providing an interval for possible values of an attribute for this object (Kaytoue, Kuznetsov, and Napoli 2011; Kaytoue, Kuznetsov, Napoli, and Duplessis 2011). Later a pattern structure for the analysis of ontology-based annotations was discussed by Coulet et al. (2013). Finally, pattern structures were also applied for mining text data by incorporating syntactic and semantic relations between text constituents (Galitsky, Ilvovsky, et al. 2014; Galitsky, Kuznetsov, et al. 2013).

Apart from FCA Pernelle et al. (2002) introduce independently an approach very similar to pattern structures. In addition they define extensional and intentional projections. Based on these projections they introduce a software ZooM for conceptual clustering of multi-valued data. Starting from simpler representation of the data in hand, they allow for "zooming" into interesting parts of the lattice with more detailed projections having three level of granularity. This work is followed by Ventos et al. (2004) studying alpha lattices. The authors join initial set of objects into groups such that these groups of objects are considered as new indivisible objects providing a smaller lattice as the result called the abstract lattice. Then, given a threshold $0 \leqslant \alpha \leqslant 1$, the authors relax this grouping by allowing subsets of these groups that are close to the groups w.r.t. the threshold α. Later it was shown that abstract lattices have a relation to modal logics (Soldano and Ventos 2011). A good example of application of abstract lattices is social network graph (Soldano and Santini 2014). Nodes of this graph correspond to objects and every node is accompanied with an additional information represented by sets of attributes. Then the characteristics of the network graph are used to find abstractions of nodes, and based on these abstractions an abstract lattice can be generated. Recently, this approach is generalized to structures called pre-confluences (Soldano 2015). In fact, pre-confluence is a structure embedding several lattices and in every embedded lattice a new closure operator can be defined.

Many other extension of pattern structures can be related to either pattern structures or their projections. For example, Ferré and Ridoux (2002) formulate a dataset of objects described by logical rules, so called logical concept analysis (LCA). Later LCA was applied for mining formal contexts with a supplementary attribute taxonomy that can be used to deduce some attributes based on the known attributes (Cellier et al. 2008).

There is a number of works introducing some limitations on possible intents of formal contexts. For example, attribute dependencies are such kinds of limitations, when some attributes require another attribute to be included into the same intent (Bělohlávek and Sklenář 2005; Belohlavek and Vychodil 2009; Messai et al. 2008). Another limitation is constraints on the closure operator of attributes (Bělohlávek and Vychodil 2006). Only some subsets of attributes can be considered as intents. Later, Kwuida et al. (2010) introduce projections that come from database domain to sets of attributes or objects. However, all these approaches can be represented as projections of pattern structures representing formal contexts.

We should notice that there are other important extensions of formal concept analysis that cannot be directly expressed as pattern structures. Bêlohlávek (1999) introduce fuzziness into formal contexts. In this work instead of crisp sets of objects and attributes, the authors suggest to use fuzzy sets. Since sets of objects are fuzzy here, it is not possible to express this structure as a pattern structure. However, for data mining the crisp case is common and we remain within the crisp case. Hacene et al. (2013) introduce relational concept analysis that takes several formal contexts and several relations between objects from different contexts. Further the formal contexts are iteratively scaled by means of these relations that allow one to extract new types of attributes and the corresponding new concepts. This approach has a strong relation to multirelational data analysis and description logics. However, in this work we remain in the classical setting of pattern mining, where patterns are extracted from a single formal context.

2 Contribution and Structure of the Thesis

This work consists of three parts corresponding to the main contributions of the thesis. The first part is denoted to mining structured data by means of standard FCA. In particular, we are interested in how one can select the best concepts within a huge lattice. We study stability of a formal concept, a well-founded measure of concept quality, and show how it can be applied to mining chemical alerts in data of mutagenetic compounds.

The second part discusses how pattern structures can be applied for mining structured data. In particular, we extend the formalism of projections of pattern structures in order to have more freedom in simplification of semilattice descriptions. It enables efficient computation by filtering of irrelevant concepts. Later we apply pattern structure for checking completeness in the web of data (i.e. in RDF data), for detecting syntactical structures that support some specific relations in a collection of texts, and for analyzing patient trajectories.

The third part of this manuscript combines together what we have studied in the previous two chapters, i.e., pattern structures and stability of a formal concept. In this chapter we answer to the following question. *Is it possible to directly find stable concepts without finding the whole set of concepts?* This approach is based on projections as well as on Δ-measure derived from stability.

The first part consists of 4 chapters (from 1 to 4). In Chapter 1 we approach mining concepts related to cMet inhibitors. The dataset is small and only a small number of patterns is found. We rely on standard FCA here. Objects are molecules from the dataset and attributes are frequent graphs of this dataset. This chapter relies on Asses, Buzmakov, et al. (2012). For dealing with larger datasets one can rely on a constraint limiting the number of patterns. Correspondingly, in Chapters 2 and 3 we study stability as a source of a possible constraint on patterns. Chapters 2 discusses behavior of stability as a constraint comparing the values of concepts obtained from independent datasets that come from the same general distribution (the chapter is based on Buzmakov et al. (2014a,b,d)). Chapter 3 evaluates stability and some other measures showing that stability has a good qualitative performance (Buzmakov et al. 2014c). Finally, in Chapter 4, we return to an application and we study how FCA enables finding structural alerts from a mutagenetic dataset. This dataset is large and the number of formal concepts is very huge. Thus, we rely on stability for selecting the most interesting concepts. The experimental study reveals new possible structural alerts that will be further studied by chemists (Métivier et al. 2015).

The second part of this manuscript is devoted to pattern structures and also consists of 4 chapters (from 5 to 8). The 5th chapter introduces pattern structures and their projections, extends the original projections for dealing with a wider set of possible projections and corresponding constraints (Buzmakov et al. 2013b, 2015c). In the consequent chapters we study some applications of pattern structures and their projections. The 6th chapter relies on heterogeneous pattern structures that are constructed as a direct product of different semilattices. This chapter discusses how one is able to find and correct errors in linked open data by fining certain types of associations. This chapter is based on Alam et al. (2015). The 7th chapter discusses how pattern structures of rooted trees can be used for mining syntactic structures that describe relations between drugs. Because of complexity of the similarity operation here, we introduce projections enabling the efficient processing of the dataset (Leeuwenberg et al. 2015). The last chapter of this part shows how pattern structures can be used for mining sequential patterns from a dataset on patient hospitalization history. The sequences that are involved in this chapter are complex, i.e., every single element of a sequences (an element of the alphabet) contains heterogeneous data such as hospital location, medical procedures, and the repetition number of the hospitalization. This chapter shows the importance of our extension of projections and is based on Buzmakov et al. (2013a,b,c, 2015a).

The third part introduces a new efficient algorithm for direct mining of concepts w.r.t. a projection-antimonotonic constraint, e.g. stable concepts. This last part of the manuscript is based on Buzmakov et al. (2015b) an consists of 2 chapters (9 and 10). The 9th chapter introduces the theoretical basis of the algorithm: the notion of projection-antimonotonicity and algorithm Σοφια for mining patterns w.r.t. a projection-antimonotonic constraint. Stability and some other constraints are shown to be projection-antimonotonic. The 10th chapter applies Σοφια for mining itemsets (the case of standard FCA) and interval tuples (the case of interval pattern structure) and shows that it is much more efficient than the approaches based on indirect mining of such kind of patterns, i.e., by means of a postpruning.

Finally we conclude the manuscript and discuss possible future research directions w.r.t. this thesis.

Part I

Formal Concept Analysis for Complex Data

Introduction

In this part of the manuscript we discuss how standard Formal Concept Analysis (FCA) (Ganter and Wille 1999) can be applied for analysis of complex data. In the case of standard FCA it is necessary to scale data into a formal context, i.e., a binary table containing a relation between objects and attributes. In this part we mostly work with data described by graphs, more precisely by molecular graphs. A possible way for scaling such kind of data is to find frequent graph fragments that are further used as attributes. That allows reducing the complexity of a data in hand to a binary table that can be processed by methods of FCA.

We start our study from a discussion of a relatively small dataset on compounds that inhibit the activity of cMet protein in Chapter 1. During this discussion we introduce FCA and suggest an approach that combines a graph mining tool for the search of frequent fragments, e.g., gSpan (Yan and Han 2002), with FCA. However, this approach cannot be directly applied to a big dataset. The problem is that the number of frequent fragments can be huge, i.e., the number of attributes in a formal context is high. Consequently, the number of formal concepts is also high. However, one of the main reason of using symbolic methods such as FCA instead of, e.g., statistics is to discover new knowledge in a well-interpretable way. If a set of patterns is huge it is a hard problem to discover a new knowledge among them.

This problem is typically approached by postfiltering a set of patterns with constraints (Vreeken and Tatti 2014). Every pattern is associated with its interestingness by a specially developed measure. Then only the patterns with a high value of interestingness w.r.t. a threshold are retained for the further analysis. FCA is packed with some interestingness measures including stability of a formal concept (Kuznetsov 2007). Correspondingly, in Chapters 2 and 3 we motivate our choice of stability as an interestingness measure. In particular, in Chapter 2 we study if stability can be considered as an interestingness measure and introduce efficient stability bounds that allow for efficient computation of stability. Later, in Chapter 3 we compare stability with some of other interestingness measures by means of classification.

Finally in Chapter 4 we discuss an application of FCA and stability for searching of structural alerts in a big dataset on mutagenic molecules. This chapter shows the possibility of meaningfull applications of FCA-based approaches to the analysis of large datasets with complex descriptions, e.g., molecular graphs.

Chapter 1

Formal Concept Analysis for Small Datasets of Complex Data. ***An Application to Search for Relationship between Chemical Structures and Binding Modes. c-Met Kinase Inhibitors as a Test Case.***

Contents

1.1 Introduction

The *in silico* prediction of the detailed interactions involved in the binding processes between active compounds and their target protein is a core step in the drug discovery pipeline as they could speed up the laborious and costly experimental understanding of the protein/ligand recognition (Leach et al. 2006). Most of these *in silico* methods are motivated by the observation that similar drugs tend to target their protein binding site in a similar way, therefore implying similar chemical functional groups. Such prediction would be particularly useful in the case of flexible protein targets for which several binding possibilities have been recognized concerning a large body of ligands presenting a large chemical diversity.

In this case, the question is, starting with a given training set representative of the binding diversity for a given flexible target, how to identify the concerned chemical functions and next to predict the proper binding modes of other active molecules for the same target? What is the most suitable prediction method (chemo-informatics and other QSAR ligand-based approaches, structure-based docking, etc.) to tackle this problem? Is it possible to avoid time-consuming protein-ligand interaction computations and to use only less costly methods? In this respect, what ligand-based method, if any, could be the most efficient for this purpose?

We have investigated these problems using a typical flexible protein of important pharmacological interest, namely c-Met kinase. This protein is involved in the metastatic problem and is therefore an important target for the discovery of potent anti-cancer agents(Jung et al. 2012). This kinase has been shown to present an important flexibility with several different binding characteristics resulting in large ligand diversity (Asses, Leroux, et al. 2009). Moreover, as the X-Rays 3D structures of more than 30 c-Met/ligand complexes have been presently solved, important protein/ligand interaction knowledge is available for setting a training procedure which could use the binding modes that have already been identified concerning c-Met. Thus the main questions is here if it is possible to classify, prior to any X-ray or NMR experiment, any inhibitor binding mode according to the comparison between this molecule chemical moieties and both the chemical functional groups and the particular substructures detected in the training set?

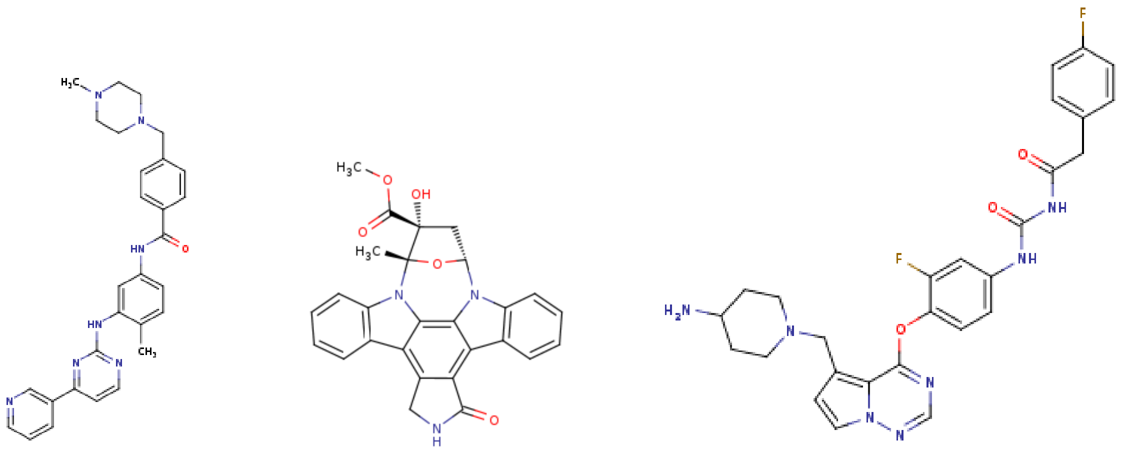
We have undertaken this problem using ligand-based chemo-informatics methods and FCA. Our goal here is not, as in most ligand-based investigations, to predict the activity/inactivity of compounds, but to classify the binding modes of already active molecules according to their decomposition into chemical functional groups and specific substructures. More precisely, this study should be done in order to:

- Validate the ability of the chemical group descriptors to cluster candidates having similar chemical characteristics freely regard to the structure of c-Met receptor.
- Possibly correlate the chemical similarities of candidate molecules with their binding mode previously annotated.
- Propose a mechanism of action/binding for un-annotated candidates on c-Met receptor for the future work.

For tackling these points, we introduced a combined classification/prediction process involving supervised and unsupervised classification within the framework of FCA, graph mining and the so-called "Jumping Emerging Patterns" (JEPs). This chapter is based on Asses, Buzmakov, et al. (2012).

1.2 Methods

First, we classified a set of molecules (the training set) according to their known chemical structures and binding modes. For that purpose, we considered these molecules as molecular graphs



(a) Imatinib or Gleevec®

(b) K-252a

(c) CKK

Figure 1.1: Examples of molecules from database.

and have applied graph mining techniques (Nijssen and Kok 2005; Yan and Han 2002) to extract maximal and frequent substructures. These substructures were used as attributes in a formal context where objects were the molecules of the training set. This formal context was "augmented" in the sense that, for each molecule in the training set, we associated a "type" or a "class" according to its binding mode. A concept lattice was built from the formal concept and the class information was used for characterizing, in this concept lattice, the concepts whose extents include objects of a single class or a binding mode. These intents of these particular concepts are "Jumping Emerging Patterns" or JEPs. JEPs are computed thanks to a combination of unsupervised (FCA) and supervised classifications. The last step involved a clustering process (i.e. hierarchical agglomerative clustering). Inhibitor molecules are represented as vectors where components are filled with functional groups and JEPs. Finally, a dendrogram was used for explaining the "proximity" of some inhibitors.

1.2.1 Database Setting

The dataset that have been used for our study contains public and known c-Met inhibitors. We have collected a working set constituted of 100 molecules including ligands coming from Protein Data Bank (PDB) c-Met complex (exampled on Figure 1.1b with with `K-252a`), and molecules on different clinical trial phase (exampled on Figure 1.1 with `BMS-907351` in phase III) and already commercialized drugs (exampled on Figure 1.1 with `Imatinib`). The molecules constituting this dataset express a large chemical diversity and present an efficient biological activity at the *nmol.* level. This heterogeneous list reinforces the importance of c-Met flexibility, already observed (Asses, Venkatraman, et al. 2012; Rickert et al. 2011), and leads to different ways of binding to c-Met. In a previous study, we have shown that the c-Met binding pocket can adopt at least 3 different binding modes due to its inhibitors diversity (Asses, Leroux, et al. 2009). This has been confirmed with the analysis of an increasing number of available c-Met crystallographic structures (Norman et al. 2012; Tiedt et al. 2011). Nowadays over than 30 c-Met crystallographic structures have been released on the PDB database and the associated ligands will constitute our training set (see Table A.1).

1.2.2 Chemical Functional Groups

For all c-Met inhibitors of our working set we decomposed their chemical structures in order to evaluate their constitutive chemical functional groups. The notion of a functional group, i.e., the capacity for a dedicated group of atoms to participate in specific chemical interactions and/or reactions, has been recognized as an important one in many fields related to drug design (He et al. 2010; Villanueva-rosales and Dumontier 2007). Several methods are available for decomposing a molecule into a list of functional groups such as Chemaxon[1], and in our case, we have retained the MOLDB5R package (Haider 2010) for performing this preliminary task of our pipeline. These PHP scripting language combinations allow us to analyze and compare a chemical dataset according to the functional groups included according to Checkmol/matchmol component.

Checkmol can recognize more than 200 functional groups classified on 3 levels of specificities. For instance, One of the `Alcohol` principal group is `1,2-Aminoalcohol` while `1,2-diol` is secondary functional group and a subgroup of `1,2-Aminoalcohol` primary group. In our case, the c-Met inhibitors dataset covers 21 principal functional groups and 42 secondary functional groups. All correspondences between principal and secondary functional groups retrieved for our dataset are detailed on Figure A.2. As presented below, the fact that a molecule has or does not have a certain functional group is used in the description of a molecule as one of the components of a description vector. These descriptions are further processed with hierarchical agglomerative clustering.

1.3 Formal concept analysis

FCA is a formalism that can be used for guiding data analysis and knowledge discovery (Ganter and Wille 1999). FCA starts with a formal context and builds a set of formal concepts organized within a concept lattice.

Definition 1.1. *A formal context is a triple (G, M, I), where G is a set of objects, M is a set of attributes and I is a relation between G and M, $I \subseteq G \times M$.*

In Table 1.1, a cross table for a formal context is shown. The rows are the set of objects G, the columnts are the set of attibutes M, and every cross represents an element of the relation I.

A Galois connection between G and M is defined as follows:

$$A' = \{m \in M \mid \forall g \in A, (g, m) \in I\}, \qquad A \subseteq G$$
$$B' = \{g \in G \mid \forall m \in B, (g, m) \in I\}, \qquad B \subseteq M$$

The Galois connection maps a set of objects to the maximal set of attributes shared by all objects and reciprocally. For example, $\{\texttt{13}, \texttt{10}\}' = \{\texttt{CAD}, \texttt{O=}\}$, while $\{\texttt{CAD}, \texttt{O=}\}' = \{\texttt{13}, \texttt{10}, \texttt{12}\}$, i.e., the set of objects $\{\texttt{13}, \texttt{10}\}$ is not maximal. Given a set of objects A, we say that A' is the description of A.

Definition 1.2. *A formal concept is a pair (A, B), where $A \subseteq G$ is a subset of objects, $B \subseteq M$ is a subset of attributes, such that $A' = B$ and $A = B'$, where A is called the extent of the concept, and B is called the intent of the concept.*

[1] http://www.chemaxon.com/jchem/doc/user/Fragmenter.html

Table 1.1: Running Example. In 1.1a, objects (the rows) are molecules; attributes (the columns) are functional groups. A cross in the cell (i, j) means that the molecule i includes the functional group j as a substructure. In 1.1b the last column designates the "class" of an object, i.e. the binding mode of the molecule.

	H	CAD	OH	P	AAE	F	O=
12	x	x			x	x	x
13	x	x		x		x	x
10		x	x				x
4	x			x	x	x	

(a) Formal Context.

Molecule	Binding Mode
12	DFG-out
13	DFG-out
10	Type-1
4	Type-1

(b) Molecule Binding Modes.

Table 1.2: A set of formal concepts w.r.t context on Table 1.1a.

Concept	A Set of Molecules (Extent)	A Set of Substructures (Intent)
$\mathcal{C}_0$		H, CAD, OH, P, AAE, F, O=
$\mathcal{C}_1$	4	H, P, AAE, F
$\mathcal{C}_2$	12	H, CAD, P, F, O=
$\mathcal{C}_3$	13	H, CAD, AAE, F, O=
$\mathcal{C}_4$	10	CAD, OH, O=
$\mathcal{C}_5$	12, 13	H, CAD, F, O=
$\mathcal{C}_6$	13, 4	H, P, F
$\mathcal{C}_7$	12, 4	H, AAE, F
$\mathcal{C}_8$	12, 13, 10	CAD, O=
$\mathcal{C}_9$	12, 13, 4	H, F
$\mathcal{C}_{10}$	12, 13, 10, 4	

A formal concept corresponds to a pair of maximal sets of objects and attributes, i.e. it is not possible to add an object or an attribute to the concept without violating the maximality property. Table 1.2 shows all concepts that can be found for the context in Table 1.1.

Formal concepts can be partially ordered w.r.t. the extent inclusion (dually, intent inclusion). For example, $(\{13\}; \{\texttt{CAD}, \texttt{O=}, \texttt{OH}\}) \leqslant (\{13, 10, 12\}, \{\texttt{CAD}, \texttt{O=}\})$. This partial order of concepts is shown in Figure 1.2. The number of formal concepts for a given context can be exponential w.r.t. the cardinality of the set of objects or the set of attributes. It is easy to see that for the context (G, G, I_G), where $I_G = \{(x, y) \mid x \in G, y \in G, x \neq y\}$, the number of concepts is equal to $2^{|G|}$. Moreover, even the problem of computing the size of the concept lattice is #P-complete (Kuznetsov 1989, 2001).

There are many algorithms for computing formal concepts and the associated concept lattice (Ganter 1984; Kuznetsov 1993; Merwe et al. 2004). A comparison of different algorithms for construction of concept lattices can be found in Kuznetsov and Obiedkov (2002).

1.4 Running Example

A running example is shown in Table 1.1. Molecules are objects (rows) of a formal context and they are given by their numbers according to Table A.1. Molecules are described by substructures (columns), corresponding to the attributes of the formal context. For example, molecule `4` includes the following substructures: `H` (Halogen), `P` (Primary Amine), `AAE` (Alkyl Aryl Ether), and `F` (Figure 1.3a) while molecule `10` includes `CAD` (Carboxilic Acid Derivative), `OH` (OH-Compound), and `O=` (Figure 1.3b).

In this case a formal context (A, B) –where A is a set of molecules and B is a set of substructures– gives the maximal set of molecules included all substructures from B and the maximal set of substructures that are included into all molecules from A.

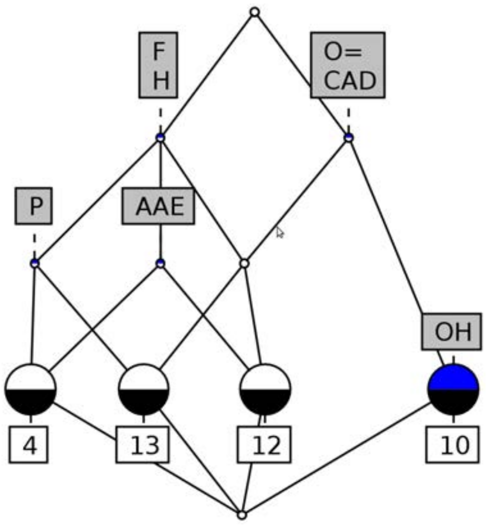


Figure 1.2: The FCA-lattice for the context on Table 1.1.

Additional information associated with the molecules is given in Table 1.1b. The table for every molecule indicates its binding mode with the c-Met protein. This additional column allows processing the context in a supervised way.

Among concepts in Table 1.2, it is possible to select concepts whose extent contains only molecules of the same class, e.g., C_1, C_2, C_3, C_4, C_5. The sets of substructures in the intents of these concepts are considered as JEPs, i.e., Jumping Emerging Patterns(Dong and Li 1999), and they describe the sets of molecules with the same binding mode.

It can be noticed that concept C_5 is more general than concepts C_2 and C_3 since the extent of C_5 includes a wider set of molecules and its intent includes a narrower set of substructures than the extents and the intents of C_2 and C_3 correspondingly. Since the extent of C_5 only contains molecules of the same type ("DFG-out"), it can be inferred that the substructures in the intent of C_5 characterize this binding mode in a "general and sufficient" way. Accordingly, we are interested in the most general concepts able to describe the binding modes. Here, we obtain

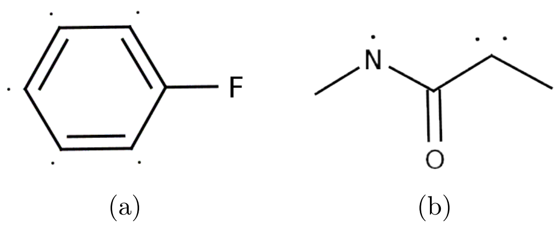


Figure 1.3: Some substructures for running example in Table 1.1.

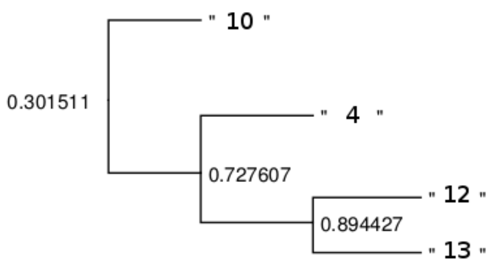


Figure 1.4: The clustering result for the context on Table 1.1a.

concepts C_1, C_4, C_5, and their intents correspond to the most general JEPs.

Since every most general JEP is likely a characteristic of a binding mode, it is worth including these JEPs into molecule descriptions for any clustering or classification purposes. Molecules of the running example can be clustered as shown in Figure 1.4. This figure should be read as follows: molecules 12 and 13 are close to each other, and are forming a cluster. This cluster is close to molecule 4 and thus molecules 12, 13, and 4 are forming a cluster at the next level. Finally, the four molecules are agglomerated into one larger cluster. This clustering process shows the "proximity" of each molecule w.r.t. the binding mode. In this way, clustering can be used to predict the binding mode of an unknown molecule.

1.5 The Classification Flow

A typical supervised classification task involves a dataset divided into a training set and a test set. The training set and the test set are sets of objects with their descriptions, where every object of the training set is labeled with a given class. Then a supervised classification method searches for rules in the training set, which can classify objects of the test set.

In our case, a dataset consists of public and known inhibitors of the c-Met protein. Here we consider 100 molecules, 30 in the training set and 70 in the test set (some molecules are shown in Figure 1.1). As indicated in the introduction, inhibitors can interact with the c-Met protein w.r.t. three different binding modes, plus one hypothetical binding mode under study (Asses, Leroux, et al. 2009). Thus, in this work, four binding modes are used for labeling molecules in the training set. The objective here is to cluster molecules according to their binding mode and then to predict the binding modes of the molecules lying in the test set.

Figure 1.5 depicts the global classification flow. The first step is to choose the way how a molecule should be described. One way is to take into account domain knowledge and to consider a molecule as a set of functional groups that are involved into interactions. But some other substructures are also involved into interactions, which are detected as follows:

1. Molecules from a dataset are considered as graphs, where vertices correspond to atoms and edges to bonds between atoms.

2. A graph mining method is used to find all frequent subgraphs, i.e. subgraphs that belong to a significant part of molecules in the dataset.

3. A formal context is built in the following way:

   - Molecules are considered as objects.
   - Extracted substructures are considered as attributes.

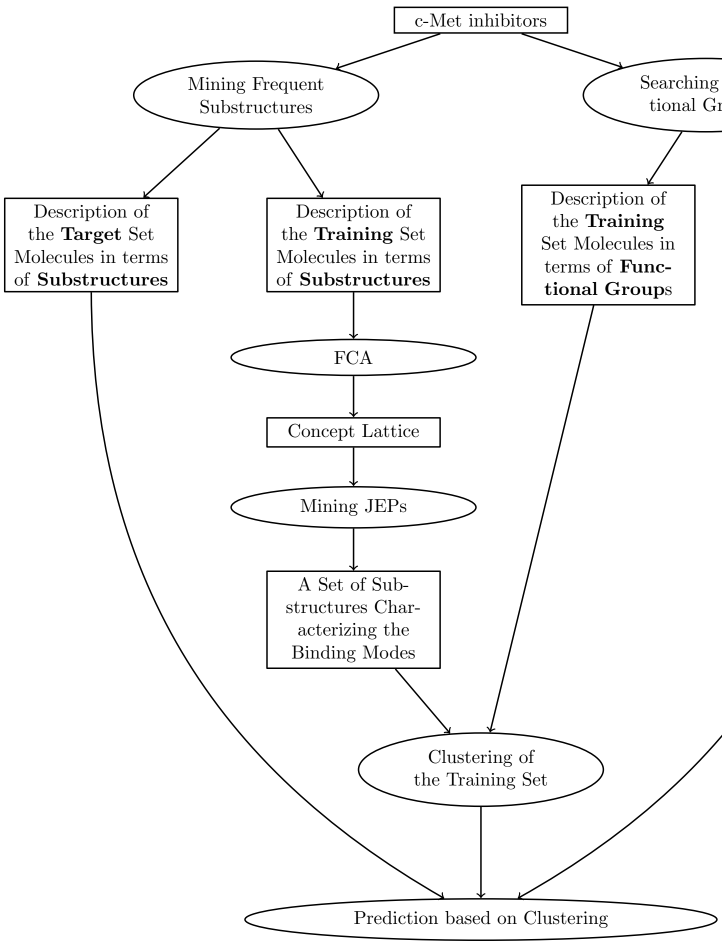

c-Met inhibitors
Mining Frequent Substructures
Searching Functional Groups
Description of the **Target** Set Molecules in terms of **Substructures**
Description of the **Training** Set Molecules in terms of **Substructures**
Description of the **Training** Set Molecules in terms of **Functional Groups**
Description of the **Target** Set Molecules in terms of **Functional Groups**
FCA
Concept Lattice
Mining JEPs
A Set of Substructures Characterizing the Binding Modes
Clustering of the Training Set
Prediction based on Clustering

Figure 1.5: Diagram of the Classification Flow.

- A molecule m and a substructure s are related if and only if the molecule m includes s as a substructure.

4. JEPs (the sets of attributes that characterize only objects of the same class) are extracted from the formal context.

In the supervised classification task, the extracted substructures are used with functional groups to cluster molecules and to predict the binding mode of molecules in the test set.

1.6 Jumping Emerging Pattens (JEPs)

JEPs were introduced as a means for classification in itemset mining (Dong and Li 1999; Poezevara et al. 2011), but the underlying idea had appeared and had been studied much earlier, e.g., within the framework of disjunctive version spaces (Ganter and Kuznetsov 2003; Mitchell 1978; Nikolaev and Smirnov 1996; Sebag 1996) or JSM-hypotheses (Finn 1991; Ganter and Kuznetsov 2000). Consider an "augmented context", i.e., a context (G, M, I) taken with an additional "class attribute" giving "class information", i.e., the class of each object in G. For a concept (A, B) the set of attributes B is a JEP if all objects in A are of the same class. In Table 1.1, the set of attributes {`F`, `O=` } is a JEP because objects `12` and `13` including these attributes are of the same class “DFG-out”.

Since a JEP characterizes a class of objects, it can be used to analyze this class and to guide a clustering method. Usually, the set of attributes associated with a single object is trivially a JEP, but there are especially interesting JEPs characterizing a class of objects. The set of JEPs is partially ordered w.r.t. the subset relation: if there are two JEPs J_1 and J_2 such that J_1 is a subset of J_2, then J_1 is more general, since it describes all objects described by J_2 and some other objects. For example, the JEP J_1 ={`H`, `CAD`, `F`, `O=` } is more general than the JEP J_2 ={`H`, `CAD`, `P`, `F`, `O=` } since J_2 describes object `13` while J_1 describes objects `13` and `12`.

Relying on the JEP definition, the intent of a formal concept is a JEP if all objects in the concept extent are in the same class. Thus it is possible to compute the set of concepts for a given context and then to extract the JEPs by checking the class of objects in the concept extents. Moreover, the most general JEPs can be selected for further analysis and for clustering.

1.7 Graph Mining

A molecule is a complex structure composed of atoms connected by bonds, that can be considered as a graph. Vertexes of the molecule graph correspond to the atoms of the molecule and are labeled with atom names. The edges of the molecule graph are labeled with types of bonds between the corresponding atoms. To apply FCA and to find a set of JEPs, a molecular graph can be described as as a set of subgraphs. Then, a formal context can be built with G as a set of molecules, M as a set of subgraphs or substructures and I the relation meaning that a molecule g has a substructure m. The problem now is to find “valid” and “interesting” substructures.

One of the ways to select valid and interesting substructures is to search for frequent subgraphs –that often appear in molecular graphs– using graph mining. For a set of graphs G and a frequency threshold F_{min}, a graph s is frequent if and only if s is a subgraph of at least F_{min} graphs from G, i.e. $|\{g \in G \mid s \subseteq g\}| \geqslant F_{min}$.

For example, considering the set of molecular graphs G in Figure 1.1 and $F_{min} = 3$, the subgraphs “N-H” and “O=C” are frequent as they occur in all molecular graphs while subgraph “C-OH” only occurring in graph (b) (Figure 1.1b) and subgraph “F-C” only occurring in graph (c) (Figure 1.1c) are not frequent.

To discover frequent subgraphs, different graph mining algorithms may be applied (Nijssen and Kok 2005; Yan and Han 2002). Here we used `gSpan` and set $F_{min} = 10$ for the dataset of 100 molecular graphs. This frequency threshold is sufficiently low to have a set of specific subgraphs characterizing every molecule, and it is sufficiently high to obtain feasible processing time.

The set of mined subgraphs can be divided into groups, where a group consist of a set of subgraphs appearing in the same set of molecular graphs. Thus, the group forms an equivalence class and can be represented by only one subgraph. Furthermore, the largest subgraphs preserve the sufficient information on substructures related to binding modes. In the present experiment, around 10^6 frequent subgraphs were extracted, then divided into 10^4 groups.

It can be noticed that if there are two frequent subgraphs g_1 and g_2 such that $g_1 \subseteq g_2$ then every closed JEP containing the subgraph g_2 contains the subgraph g_1. Thus, if a JEP contains g_2, there is no need to consider g_1.

1.8 Hierarchical Agglomerative Clustering (HAC)

Here we describe a hierarchical agglomerative clustering process based on the extracted JEPs and background knowledge on functional groups (Murtagh 1983). Molecules are described by vectors having 55 components, including 42 functional groups[2] and 13 JEPs. This 13 JEPs are selected as the most representative for the molecules in the training set. Each attribute of the vector therefore corresponds either to a chemical functional group or to a substructures of the JEPs with value set to 1 when this chemical function/substructure is present and 0 otherwise. The choice of a proper similarity is crucial for ensuring the quality of the clustering. Here, the cosine similarity was chosen according to the results of several specialized studies (Qian et al. 2004; Yamagishi et al. 2006). If m_1 and m_2 are the description vectors of two molecules, then $((\vec{m_1}, \vec{m_2})$ denotes the scalar product of two vectors):

$$sim_{cos}(\vec{m_1}, \vec{m_2}) = \frac{(\vec{m_1}, \vec{m_2})}{|\vec{m_1}| \cdot |\vec{m_2}|} \tag{1.1}$$

The "centroid" of a cluster of molecules C, denoted by $\vec{centr}(C)$, is calculated as follows:

$$\vec{centr}(C) = \frac{1}{|C|} \sum_{m_i \in C} \vec{m_i} \tag{1.2}$$

Similarity between two clusters or between a molecule and a cluster is calculated with the same formula (1.1) by substituting the cluster C with its centroid $\vec{centr}(C)$.

The HAC clustering is a bottom-up process working as follows. For every molecule a unique cluster is created. Actually, all clusters are progressively merged until only one unique cluster remains. Considering at some step the set of remaining clusters $\mathfrak{C} = \{C_1, C_2, .., C_k\}$, a new cluster C_{k+1} is created by merging two clusters C_i and C_j maximizing the similarity measure between them. The new cluster is added to the set of clusters while C_i and C_j are removed from $\mathfrak{C}$. Finally, the process stops when only one cluster remains, $|\mathfrak{C}| = 1$.

$$\begin{aligned}
(C_i, C_j) &= \underset{C_i, C_j \in \mathfrak{C}, C_i \neq C_j}{argmax} \; sim_{cos}(\vec{centr}(C_i), \vec{centr}(C_j)) && (1.3)\\
C_{k+1} &:= C_i \cup C_j && (1.4)\\
\mathfrak{C} &:= \mathfrak{C} \cup \{C_{k+1}\} \backslash \{C_i, C_j\} && (1.5)
\end{aligned}$$

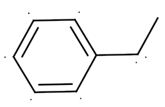
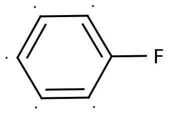
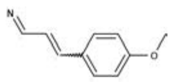
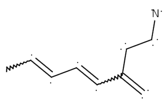
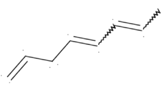
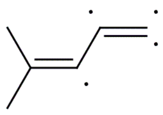
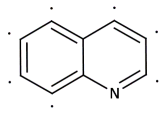
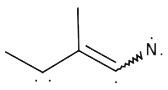
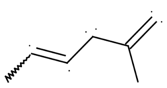
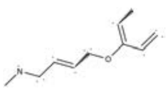
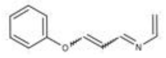
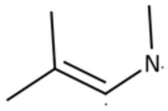
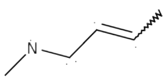
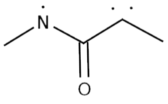
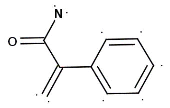
The result of HAC is shown on a dendrogram (see Figures 1.4 and 1.6). Each "vertex" of the dendrogram corresponds to a merging step of the algorithm. The number attached to the vertex represents the similarity between the two clusters at the lower level. The correlation between chemical similarities and binding modes is discussed below.

1.9 Results and Discussion

The three different c-Met binding modes observed are: `Type1` where we can found the major part of the PDB c-Met structures, `DFG-out` and `C-helix-out` commonly known as Type 2. During

[2] The functional groups were extracted by means of specialized algorithm "Checkmol" http://merian.pch.univie.ac.at/~nhaider/cheminf/cmmm.html.

Table 1.3: Examples of the result JEPs. Every column corresponds to one JEP. Only some of structures for every JEP were exemplified. According to the dataset, all the molecules including all the substructures of the second (for example) column are of DFG-out binding mode. These sets of substructures belongs to disjoint sets of molecules.

Type1	DFG-out	C-helix-out	Type1bis
			
			
			
			

the present study we have updated our c-Met catalytic site analysis and have also observed a kind of different `Type1` binding modes for `3RHK`, `3QTI`, `3ZZE` and `3ZXZ` as reported by Eathiraj et al. (2011) and Tiedt et al. (2011), that we called `Type1bis`. With our methodology, we propose now to retrieve information from the 30 XRD ligands see Table A.1 considered as a training set and then to predict the behaviour of the 70 remaining molecules in c-Met binding pocket.

1.9.1 Descriptors found from JEPs

After applying graph mining on the set of molecules, a formal context including 30 objects (molecules) and 10^4 attributes (substructures) is built. The cardinality of the sets of most general JEPs for the different binding modes are distributed as follows:

- 35 JEPs for `Type1` binding mode;
- 1 JEP for `DFG-out` binding mode;
- 1 JEP for `C-helix-out` binding mode;
- 3 JEPs for `Type1bis` binding mode.

Examples of extracted JEPs for different binding modes are shown in Table 1.3. Every column corresponds to one JEP being characteristic for some binding mode. Every JEP can have a lot of substructures and only three to four largest ones are retained for clustering and shown on that table.

1.9.2 Clustering combining Emerging Pattern and Chemical functional groups descriptors

Dendrogram interpretation

The dendrogram reported in Figure 1.6 represents existing similarities between the 30 molecules from crystallographic 3D representations collected from the PDB database and which represent

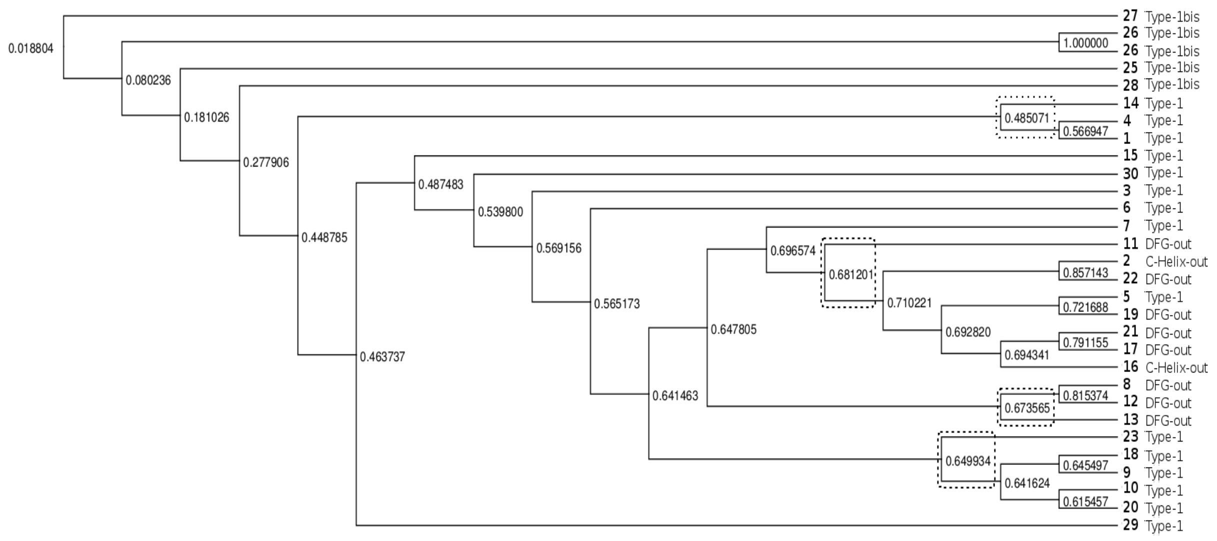


Figure 1.6: The clustering result for 30 known molecules, described by functional groups and JEPs.

the so-called “training set”. A study of similarity using cosine similarity of vectors representing the properties of these molecules has allowed us to have this type of dendrograms. These feature vectors include the properties of chemical functional groups of the 30 inhibitors as well as the existence of certain structural units represented by the previously found JEPs. We have annotated on this dendrogram the binding modes types of the molecules according to our analysis. The observation of the dendrogram shows us two important things:

1. The separation between the different binding modes has been made by clustering (classification) coupled to the analysis of JEPs.

2. The separation of `Type1` into two subgroups.

We have a very distinct `Type1` that includes molecules: `14`, `4`, `1`, `15`, `30`, `3`, and `6`. Heterogeneity of the second group seems normal.

Interaction between binding modes

The `DFG-out` and `C-helix-out` binding modes are fairly close in agreement with the conclusions of Dussault and Bellon (2008) on two existing binding modes for c-Met. The existence of the coupled `Type1` and `DFG-out` inhibitors with a similarity coefficient equal to 0.72 can be explained by the fact that this inhibitor in particular have a very strong interaction with a group of the `DFG-out` moiety (more particularly the residue `Asp1222`). The two subtypes of `Type1` are distinguished by the interactions of varying intensity with groups approaching the `DFG-out` motif (area). When analysing the dendrogram representation (Figure 1.6) in terms of common chemical moieties, it appears that `Type1` bound molecules are less connected to the `DFG-out` ones (lowest similarity coefficients) and therefore present less common chemical residues. Indeed the first subtype 1 (molecules: `14`, `4`, `1`, `15`, `30`, `3`, `6`) have no interaction with the group characterizing the type `DFG-out`. Meanwhile, the second subtype 1 (molecules: `23`, `18`, `9`, `10`, `20` and `29`) has interactions that are significant enough to bring the type of `DFG-out` without being part of this latest binding mode type.

1.10 Conclusion

Here, an original combination of supervised and unsupervised classification methods, associated with graph mining and clustering techniques, was used to successfully solve a problem concerning the binding modes of c-Met kinase inhibitors. The process used in the present chapter, which implied a variety of knowledge discovery methods, is original and its application could be generalized to other problems. Another contribution of this work concerns fragment-based lead discovery approaches (Rees et al. 2012) as the jumping emerging patterns obtained here could be used to define more precisely the necessary chemical fragments associated to a given binding mode. This should boost the efficiency of such fragment-based methods as providing the strongest drops of knowledge ensuring such methods to be successful.

However, it should be noticed that given a larger set of objects, e.g., molecules, one can discover huge sets of concepts and JEPs. There are two challenges that one should address here. First, huge sets of concepts and JEPs make the analysis of the sets very hard. One can hardly check the utility of a million of patterns. Moreover, a large part of these patterns is either redundant or corresponds to artefacts of the dataset. The second problem is computational. In order to find JEPs it is necessary to find the set of all concepts. If this set is large, the computational time becomes to be significant. Correspondingly, in the rest of Part I we discuss the first problem of a such kind of analysis, while in Parts II and III we address the second problem.

Chapter 2

Interestingness of a Concept by means of Stability and its Scalable Estimates

Contents

2.1 Introduction

Given a dataset, data mining methods may reveal a huge number of patterns, so filtering patterns w.r.t. some interestingness measures can be necessary. The question of how much a pattern is interesting arises in many areas of data mining, including those that employ tools of Formal Concept Analysis (FCA) (Ganter and Wille 1999). We remind that FCA aims at computing concepts and their lattices from a formal context, a triple (G, M, I) where G is a set of objects (elements or transactions of a dataset), M is a set of attributes used to build the description of every object, and $I \subseteq G \times M$ is a relation between objects and attributes. The number of all formal concepts of a context can be in the worst-case exponential in the size of the object and attribute sets. Even the problem of computing this number (i.e., the size of the concept lattice) is #P-complete (Kuznetsov 1989, 2001). Thus, a special procedure for selecting the most interesting concepts is needed. Two options can be distinguished. The first one is to introduce background knowledge into the procedure for computing concepts (Belohlavek and Vychodil 2009; Bělohlávek and Vychodil 2006; Buzmakov et al. 2013b; Dias and Vieira 2013; Ganter and Kuznetsov 2001). Background knowledge allows one to sort concepts which are likely to be useful

	m_1	m_2	m_3	m_4	m_5	m_6
g_1	x					x
g_2		x				x
g_3			x			x
g_4				x		x
g_5					x	

Table 2.1: A toy formal context.

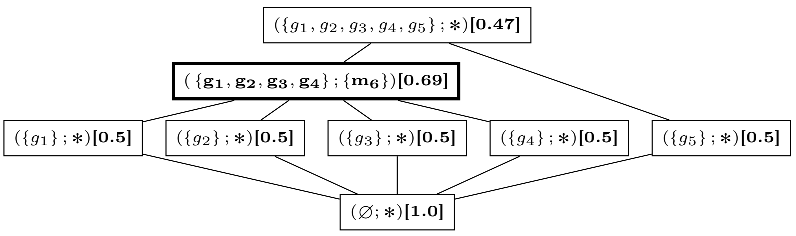


Figure 2.1: Concept Lattice for Table 2.1 with corresponding stability indexes.

for the current goal. In this case, although the number of concepts can be significantly reduced, the size of the lattice can still be huge. The second option is to rank concepts in the lattice using an interestingness measure.

Belohlavek and Trnecka (2013) provide several measures for ranking concepts that stem from human behavior. Stability is another measure for ranking concepts, introduced by Kuznetsov (1990) and later revised by Kuznetsov (2007), Kuznetsov, Obiedkov, and Roth (2007), and Roth et al. (2008). Several other methods are considered by Klimushkin et al. (2010), where it is shown that stability is more reliable for artificially noised data. Although there is a number of methods for ranking concepts, there is neither a reliable comparison nor a deep research on interestingness of the selection methods mentioned above. In this work we focus on the stability measure and its estimates. The intuition behind stability is the probability of preserving the concept intent when some objects of the context are removed. In this chapter we study the behavior of stability computed in several datasets coming from the same general population. It is done by splitting datasets into two disjoint subsets called reference and test datasets. The stability behaviour is shown to be similar in reference and test datasets independently of the general population.

Since computing stability is #P-complete (Kuznetsov 1990, 2007) one needs to use estimates or approximations in order to compute stability over large lattices. Correspondingly, in the second part of this chapter we introduce estimates of stability. It is shown empirically that their performance is better then the performance of the known Monte Carlo approximation introduced by Babin and Kuznetsov (2012).

This chapter is based on Buzmakov et al. (2014b,d) and organized as follows. Section 2.2 introduces the formal definition of stability, its estimate and Monte Carlo approximation and discusses their relation. In Section 2.3 experiments on interestingness of stability are explained and discussed. Then Section 2.4 validates the introduced estimate.

```
1 Function FindStabilityLimits
    Data: A context $\mathbb{K} = (G, M, I)$, a concept $\mathcal{C}$.
    Result: $< Left, Right >$, a pair of left and right limits for the stability.
2   $Left \leftarrow 1$;
3   $Right \leftarrow 1$;
4   $children \leftarrow$ FindChildren($\mathbb{K}$,$\mathcal{C}$) ;                /* $O(|N| \cdot |M|^2$ */
5   $minDiffSize \leftarrow \infty$;
6   foreach $ch \in children$ do                     /* $O(|M|)$ iterations at most */
7       $diffSize \leftarrow |$Ext($\mathcal{C}$)\Ext($ch$)$|$;
8       $minDiffSize \leftarrow \min(minDiffSize, diffSize)$;
9       $Left \leftarrow Left - 2^{-diffSize}$;
10  $Right \leftarrow 1 - 2^{-minDiffSize}$;
11  return $< Left, Right >$;
```

Algorithm 1: An algorithm computing stability bounds according to (2.2)

```
1 Function FindStabilityLimitsPlusMC
    Input: A context $\mathbb{K} = (G, M, I)$; a concept $\mathcal{C}$; a precision $\varepsilon$ and an error rate $\delta$ for
           Monte-Carlo.
    Output: $< Left, Right >$, a pair of left and right limits for stability.
2   $< Left, Right > \leftarrow$ FindStabilityLimits($\mathbb{K}$,$\mathcal{C}$);
3   if $Right - Left > 2 \cdot \varepsilon$ then
4       $stabilityMC \leftarrow$ FindStabilityByMonteCarlo($\mathbb{K}$,$\mathcal{C}$,$\varepsilon$,$\delta$);
5       $Left \leftarrow \max(Left, stabilityMC - \varepsilon)$;
6       $Right \leftarrow \min(Right, stabilityMC + \varepsilon)$;
7   return $< Left, Right >$;
```

Algorithm 2: An algorithm based on combination of (2.2) and Monte-Carlo approach.

2.2 Stability of a Formal Concept

2.2.1 The Definition of Stability

Stability is an interestingness measure of a formal concept introduced by Kuznetsov (1990) and later revised by Kuznetsov (2007), Kuznetsov, Obiedkov, and Roth (2007), and Roth et al. (2008).

Definition 2.1. *Given a concept $\mathcal{C}$, concept stability* $\texttt{Stab}(\mathcal{C})$ *is defined as*

$$\texttt{Stab}(\mathcal{C}) := \frac{|\{s \in \wp(\texttt{Ext}(\mathcal{C})) \mid s' = \texttt{Int}(\mathcal{C})\}|}{2^{|\texttt{Ext}(\mathcal{C})|}} \tag{2.1}$$

i.e., the relative number of subsets of the concept extent (denoted by $\texttt{Ext}(\mathcal{C})$*), whose description (i.e., the result of $(\cdot)'$) is equal to the concept intent (denoted by* $\texttt{Int}(\mathcal{C})$*) where $\wp(P)$ is the powerset of a set P.*

Example 2.1. *Figure 2.1 shows a lattice for the context in Table 2.1, for simplicity some intents are not given. The extent of the highlighted concept $\mathcal{C}$ is* $\texttt{Ext}(\mathcal{C}) = \{g_1, g_2, g_3, g_4\}$*, thus, its power set contains 2^4 elements. The descriptions of 5 subsets of* $\texttt{Ext}(\mathcal{C})$ *($\{g_1\}, \ldots, \{g_4\}$ and $\varnothing$) are* *different from* $\mathtt{Int}(\mathcal{C}) = \{m_6\}$*, while all other subsets of* $\mathtt{Ext}(\mathcal{C})$ *have a description equal to* $\{m_6\}$*. So,* $\mathtt{Stab}(\mathcal{C}) = \frac{2^4-5}{2^4} = 0.69$*.*

Stability measures the dependence of a concept intent on objects of the concept extent. More precisely this intuition behind stability can be described by the following proposition introduced by Roth et al. (2006, 2008).

Proposition 2.1. *Let* $\mathbb{K} = (G, M, I)$ *be a formal context and* $\mathcal{C}$ *a formal concept of* $\mathbb{K}$*. For a set* $H \subseteq G$*, let* $I_H = I \cap H \times M$ *and* $\mathbb{K}_H = (H, M, I_H)$*. Then,*

$$\mathtt{Stab}(\mathcal{C}) = \frac{|\{\mathbb{K}_H \mid H \subseteq G \text{ and } \mathtt{Int}(\mathcal{C}) \text{ is closed in } \mathbb{K}_H\}}{2^{|G|}}$$

The proposition says that stability of a concept $\mathcal{C}$ is the relative number of subcontexts where there exists a concept with the intent $\mathtt{Int}(\mathcal{C})$. A stable concept can be found in many such subcontexts, and therefore is likely to be found in an unrelated context built from the population under study. This "likely" was never studied and one. Correspondingly in this chapter we check if stability can be used to find significant patterns within the whole population.

It was shown that, given a context and a concept, the computation of concept stability is #P-complete (Kuznetsov 1990, 2007). One of the fastest algorithm for processing concept stability using a concept lattice L is proposed by Roth et al. (2008), with a worst-case complexity of $O(L^2)$, where L is the size of the concept lattice. This theoretical complexity bound is significantly higher than that of algorithms computing all formal concepts and in practice computing stability may take much more time than lattice building algorithms (Buzmakov et al. 2013c). Moreover, this algorithm needs the lattice structure to be computed, requiring additional computations and memory usage. Thus, finding a good estimate of concept stability is a crucial question. Here we present an efficient way for such an estimate.

2.2.2 Estimation of Stability

Given a concept $\mathcal{C}$ and its descendant $\mathcal{D}$, we have $(\forall s \subseteq \mathtt{Ext}(\mathcal{D}))(s'' \subseteq \mathtt{Ext}(\mathcal{D}) \wedge s' \supseteq \mathtt{Int}(\mathcal{D}) \supset \mathtt{Int}(\mathcal{C}))$, i.e., $s' \neq \mathtt{Int}(\mathcal{C})$. Thus, we can exclude all subsets of the extent of a descendant while computing the numerator of stability in (2.1). On the other hand only subsets of the extents of descendants should be excluded from the numerator in (2.1). Thus, if we exclude the subsets of the extents of all immediate descendants, we exclude everything that is needed but probably some subsets can be excluded several times. Hence we obtain a lower bound for stability:

$$1 - \sum_{\mathcal{D} \in \mathtt{DD}(\mathcal{C})} \frac{1}{2^{\Delta(\mathcal{C},\mathcal{D})}} \leqslant \mathtt{Stab}(\mathcal{C}) \leqslant 1 - \max_{\mathcal{D} \in \mathtt{DD}(\mathcal{C})} \frac{1}{2^{\Delta(\mathcal{C},\mathcal{D})}}, \tag{2.2}$$

where $\mathtt{DD}(\mathcal{C})$ is a set of all direct descendants of $\mathcal{C}$ in the lattice and $\Delta(\mathcal{C}, \mathcal{D})$ is the size of the set-difference between extent of $\mathcal{C}$ and extent of $\mathcal{D}$, i.e., $\Delta(\mathcal{C}, \mathcal{D}) = |\mathtt{Ext}(\mathcal{C}) \setminus \mathtt{Ext}(\mathcal{D})|$. The pseudo-code for computing this estimate is shown in Algorithm 1. The time complexity of this approach for a concept is equal to the complexity of finding immediate descendants of the concept, i.e., $O(n \cdot m^2)$.

Example 2.2. *If we want to compute stable concepts (with stability more than* 0.97*), then according to the upper bound in (2.2) we should compute for each concept* $\mathcal{C}$ *in the lattice* $\Delta_{\min}(\mathcal{C}) = \min_{\mathcal{D} \in \mathtt{DD}(\mathcal{C})} \Delta(\mathcal{C}, \mathcal{D})$ *and select concepts obeying* $\Delta_{\min}(\mathcal{C}) \geqslant -\log(1 - 0.97) = 5.06$*.*

The upper bound of the equation can be found in (Roth et al. 2008), while the lower bound has not been studied yet. We know that given a context (G, M, I), the number of children for any concept is limited by cardinality of M. Every summand in the lower bound of stability in (2.2) is smaller than $2^{-\Delta_{\min}(\mathcal{C})}$. This gives the following estimate.

$$1 - |M| \cdot 2^{-\Delta_{\min}(\mathcal{C})} \leqslant 1 - \sum_{\mathcal{D} \in \mathtt{DD}(\mathcal{C})} 2^{-\Delta(\mathcal{C},\mathcal{D})} \leqslant \mathtt{Stab}(\mathcal{C}) \tag{2.3}$$

This suggests that stability can have an exponential behavior w.r.t. the size of the context and, thus, most of the concepts have stability close to 1 when the size of the context increases. This behavior of stability is also noticed by authors of (Jay et al. 2008) for their dataset. So, to use stability for large datasets it is worth computing logarithmic stability for every concept $\mathcal{C}$:

$$\mathtt{LStab}(\mathcal{C}) = -\log_2(1 - \mathtt{Stab}(\mathcal{C})) \tag{2.4}$$

Taking into account the bounds in (2.2) and in (2.3), we have the following:

$$\Delta_{\min}(\mathcal{C}) - \log_2(|M|) \leqslant -\log_2(\sum_{\mathcal{D} \in \mathtt{DD}(\mathcal{C})} 2^{-\Delta(\mathcal{C},\mathcal{D})}) \leqslant \mathtt{LStab}(\mathcal{C}) \leqslant \Delta_{\min}(\mathcal{C}) \tag{2.5}$$

This approach is referred as the *bounding method*. It can efficiently bound stability for any concept of the lattice. However, the tightness of this bound cannot be ensured.

Babin and Kuznetsov (2012) suggest a method for approximating concept stability based on a Monte Carlo approach. Given a concept $\mathcal{C}$, the idea is to randomly count the number of "good" subsets $s \subseteq \mathtt{Ext}(\mathcal{C})$ of the extent of $\mathcal{C}$ such that $s' = \mathtt{Int}(\mathcal{C})$. Then knowing the number of iterations N and the number of "good" subsets N_{good}, stability can be calculated as the relation between them: $\mathtt{Stab}(\mathcal{C}) = \frac{N_{good}}{N}$. In their paper the authors provide the following approximation of the number of iterations:

$$N > \frac{1}{2\varepsilon^2} \ln \frac{2}{\delta} \tag{2.6}$$

where ε is the precision of the approximation and δ is the error rate, i.e., if one have computed stability approximation s, then the exact value of stability is within the interval $[s - \varepsilon; s + \varepsilon]$ with the probability $1 - \delta$. This method will be later referred as the *Monte Carlo method*.

Example 2.3. *In order to approximate stability with precision $\varepsilon = 0.01$ and error rate $\delta = 0.01$, it is necessary to make at least $N = 2.65 \cdot 10^4$ iterations.*

Example 2.3 shows that the number of iterations for one concept can be huge and, thus, the Monte Carlo method should be less efficient than the bounding method. Nevertheless the Monte Carlo method can ensure a certain level of tightness. Consequently the bounding method and the Monte Carlo method can be used in a complementary way as follows. First, the stability bounds are computed. Second, if the tightness of the bounding method is worse than the tightness of the Monte Carlo method, the latter should be applied. The pseudo-code of this approach is shown in Algorithm 2. Hereafter, it is referred as the *combined method*.

Recall that there are three other estimates of stability (Kuznetsov 1990, 2007; Roth et al. 2008) whose study is out of the scope of the present work. Two of these estimates are applicable incrementally, i.e., when stability is known for a concept from some context and several objects are added to this context authors estimate the stability of the corresponding concept in the new lattice. For the third estimate no efficient computation is known for the moment.

In the next section we present two types of experiments. In Subsection 2.3.1 an experiment on the predictability of stability is presented. The discussion continues in Subsection 2.3.3 with the behaviour of stability thresholds and in Subsection 2.3.4 with stability ordering ability.

Table 2.2: Datasets used in the experiments. Column 'Shortcut' refers to the short name of the dataset used in the rest of the chapter; 'Size' is the number of objects in the dataset; 'Max. Size' is the maximal number of objects in a random subset of the dataset the lattice structure can be computed for; 'Max. Lat. Size' is the size of the corresponding lattice; 'Lat. Time' is the time in seconds for computing this lattice; 'Stab. Time' is the time in seconds for computing stability for every concept in the maximal lattice.

Dataset	Shortcut	Size	Max. Size	Max. Lat. Size	Lat. Time	Stab. Time
Mushrooms[1]	Mush	8124	8124	$2.3 \cdot 10^5$	324	57
Plants[2]	Plants	34781	1000	$2 \cdot 10^6$	45	10^4
Chess[3]	Chess	3198	100	$2 \cdot 10^6$	30	$7.4 \cdot 10^3$
Solar Flare (II)[4]	Flare	1066	1066	2988	< 1	< 1
Nursery[5]	Nurs	12960	12960	$1.2 \cdot 10^5$	245	5

[1]http://archive.ics.uci.edu/ml/datasets/Mushroom
[2]http://archive.ics.uci.edu/ml/machine-learning-databases/plants/
[3]http://archive.ics.uci.edu/ml/datasets/Chess+(King-Rook+vs.+King-Pawn)
[4]http://archive.ics.uci.edu/ml/datasets/Solar+Flare
[5]http://archive.ics.uci.edu/ml/datasets/Nursery

2.3 Experiment on Predictability of Stability

The experiments are run on an "Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz" computer with 8Gb of memory under Ubuntu 12. The algorithms are not parallelized. Public datasets available from the UCI repository (Frank and Asuncion 2010) are used for the experimentation. These datasets are shown in Table 2.2. With their different size and complexity, these datasets provide a rich experimental basis. Complexity here stands for the size of the concept lattice given the initial number of objects in the corresponding context. For example, `Chess` is the most complex dataset as for only 100 objects in the context there are already $2 \cdot 10^6$ of concepts in the lattice.

2.3.1 The Experiment Flow

Recall that stability of a concept $\mathcal{C}$ can be considered as the probability for the intent of $\mathcal{C}$ to be preserved in the lattice when some objects are removed. However, when computing stability, one wants to know if the intent of a stable concept is a general characteristic rather than an artefact specific for a dataset. For that it is necessary to evaluate stability w.r.t. a test dataset different from the reference one. Reference and test datasets are two names of disjoint datasets on which stability behaviour is evaluated. In order to do that the following scheme of experiment is developed:

1. Given a dataset $\mathbb{K} = (G, M, I)$ of size $K = |G|$ objects, experiments are performed on dataset subsets whose size in terms of number of objects is N. This size is required to be at least half the size of K. For example, for a dataset of size $K = 10$ the size of it subset can be $N = 4$.

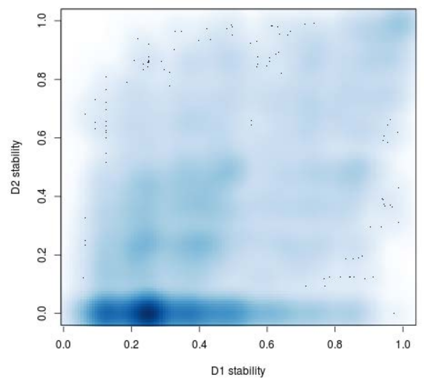
2. Two disjoint dataset subsets $\mathbb{K}_1$ and $\mathbb{K}_2$ of size N (in terms of objects) of dataset $\mathbb{K}$ are generated by sampling, e.g., $\mathbb{K}_1 = (\{g_2, g_5, g_6, g_9\}, M, I)$ and $\mathbb{K}_2 = (\{g_3, g_7, g_8, g_{10}\}, M, I)$. Later, $\mathbb{K}_1$ is used as a reference dataset for computing stability, while $\mathbb{K}_2$ is a test dataset for evaluating stability computed in $\mathbb{K}_1$.

3. The corresponding sets of concepts $\mathfrak{L}_1$ and $\mathfrak{L}_2$ with their stability are built for both datasets $\mathbb{K}_1$ and $\mathbb{K}_2$.

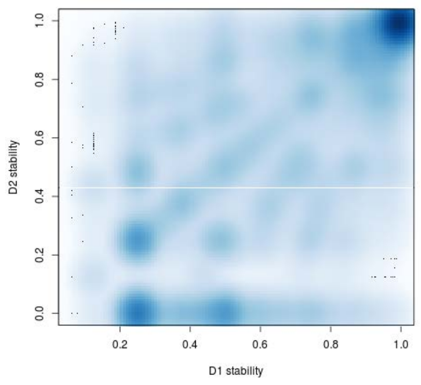
4. The concepts with the same intents in $\mathfrak{L}_1$ and $\mathfrak{L}_2$ are declared as corresponding concepts.
5. Based on this list of corresponding concepts, a list of pairs $S = \{\langle X, Y \rangle, \dots\}$ is built, where X is stability of the concept in $\mathfrak{L}_1$ and Y is stability of the corresponding concept in $\mathfrak{L}_2$. If an intent exists only in one dataset, its stability is set to zero in the other dataset (following the definition of stability). Finally, the list $LS = \{\langle X_{\log}, Y_{\log} \rangle, \dots\}$ includes the stability pairs in S in logarithmic scale as stated in formula (2.4).
6. Then sets of pairs S and LS are further used to study the behaviour of stability on disjoint (independent) datasets coming from the same general population.

The idea of evaluating stability computed on a reference dataset w.r.t. a test dataset comes from the supervised classification methods. Moreover, this idea is often used to evaluate statistical measures for pattern selection and can be found as a part of pattern selection algorithms with a good performance (Webb 2007).

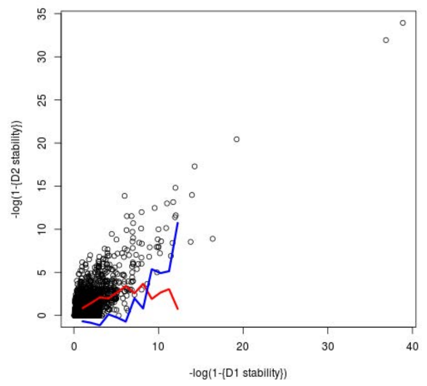
2.3.2 The General Behaviour of Stability



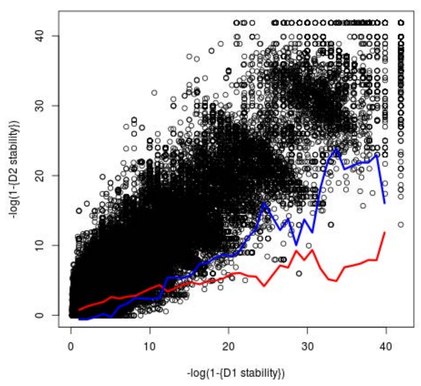
(a) Mush120



(b) Mush4000



(c) Mush120 logarihtmic scale



(d) Mush4000 logarihtmic scale

Figure 2.2: Stability in the test dataset w.r.t the reference one.

Sets of pairs S and LS can be drawn by matching every point $\langle X, Y \rangle$ to a point in a 2D-plot. The best case is $y = x$, i.e., stability for a concept in $\mathfrak{L}_1$ is equal to stability of the corresponding concept in $\mathfrak{L}_2$, meaning that stability is not dependant on the dataset. However, this is hardly the case in real-world experiments. All interestingness measures depend on the dataset, while any measure should be able to predict its value independently of the dataset. Figures 2.2a and 2.2b show the corresponding diagrams for the datasets `Mush120` and `Mush4000`[3]. These figures also highlight the fact that many concepts have stability close to 1, and that the larger the dataset, the larger the number of concepts with stability close to 1. It is in accordance with the work (Jay et al. 2008) where most of the concepts have stability close to 1. However, when the logarithmic set LS is used, a blurred line $y = x$ can be perceived in Figures 2.2c and 2.2d. Moreover, selecting the concepts which are stable w.r.t. a high threshold, say θ_r, in the reference dataset $\mathbb{K}_1$, the corresponding concepts in $\mathbb{K}_2$ are stable w.r.t. a lower threshold, say θ_t. Thus, we can conclude that stability is more tractable in the logarithmic scale, and then we only consider this logarithmic scale in the rest of this chapter.

2.3.3 Setting a Stability Threshold

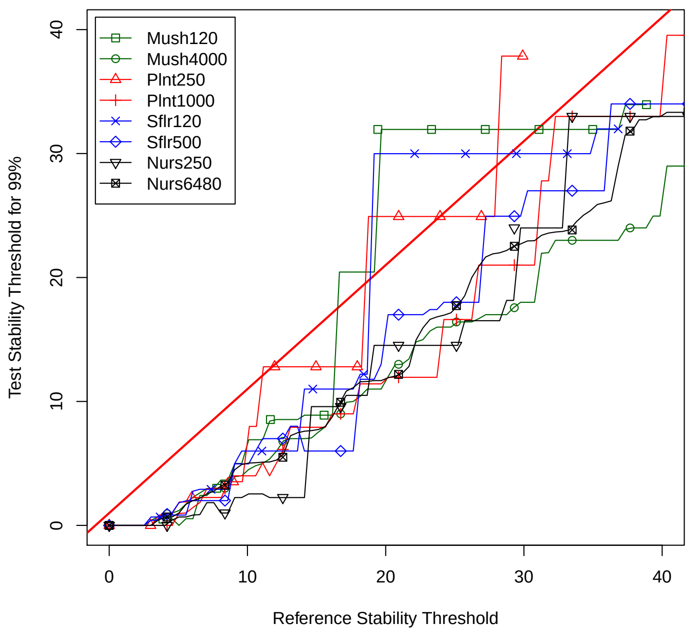


Figure 2.3: Stability threshold in the test dataset ensuring that 99% of concepts corresponding to stable ones in the reference dataset are stable.

The dependency between two thresholds θ_r and θ_t of stability are shown in Figure 2.3. The x-axis corresponds to the stability threshold in the reference dataset $\mathbb{K}_1$, while the y-axis corresponds to the stability threshold in the test dataset $\mathbb{K}_2$. The lines correspond to the 99% level, i.e., given stability in K_1, what should be the stability threshold in the test dataset $\mathbb{K}_2$ such that 99% of stable concepts in K_1 are also stable in K_2. In this figure one can see that lines begin to grow from 5 meaning that given stability threshold less than 5 in $\mathbb{K}_1$ no stability

[3] From here, the name of a dataset followed by a number such as '$NameN$' refers to an experiment based on the dataset $Name$ where $\mathbb{K}_1$ and $\mathbb{K}_2$ are of the size N.

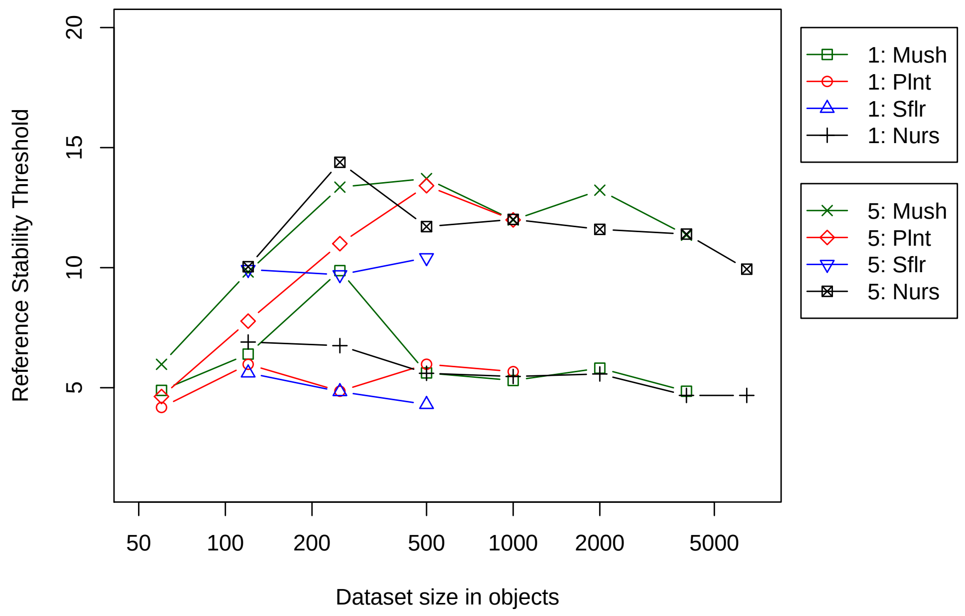


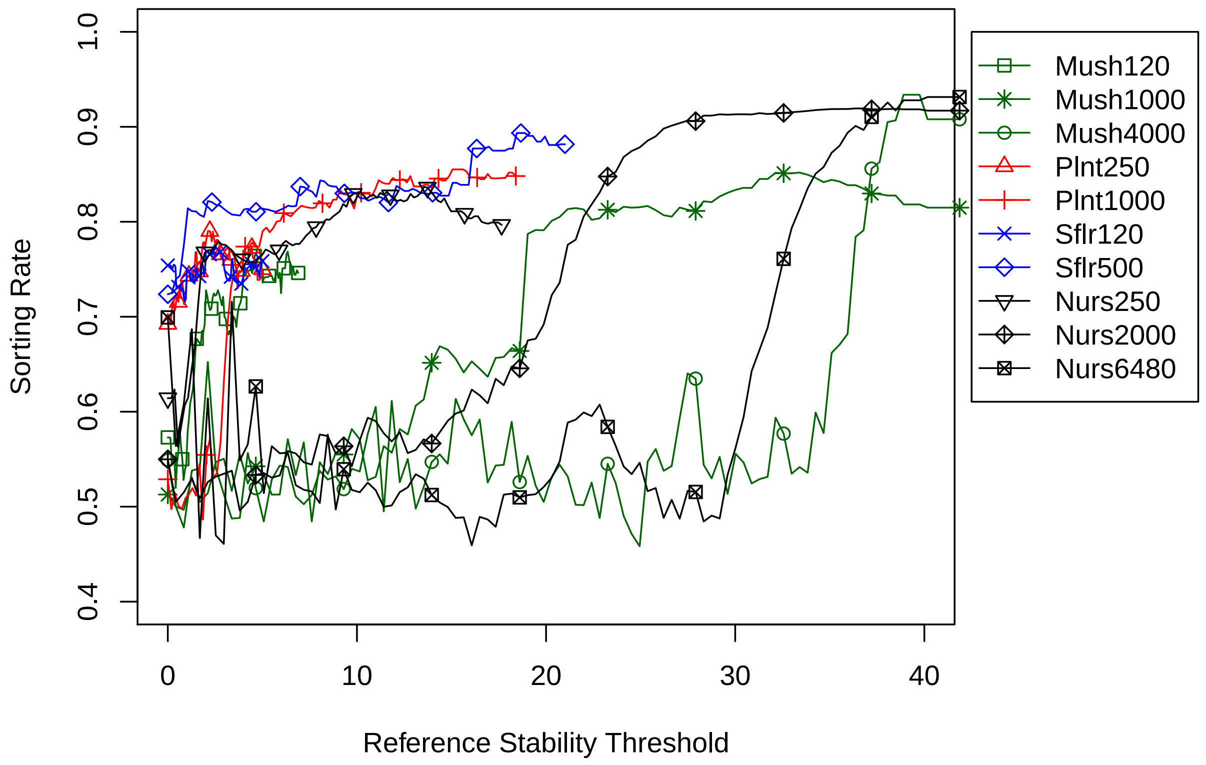
Figure 2.4: Stability threshold in the reference dataset ensuring that 99% of concepts in the test dataset corresponding to stable concepts in the reference dataset are stable with stability thresholds 1 or 5.

threshold in the test dataset $\mathbb{K}_2$ can ensure 99% of stable concepts. We can also see two types of lines. The lines with stairs correspond to the datasets with small number of stable concepts, while the others behave nearly the same. The stairs for the small datasets reflects the fact that the number of stable concepts is low and when a concept disappeared the 99% level change significantly because of the disappeared concept. This behavior suggests that in order to ensure that a concept remains stable in another dataset with threshold $\theta_{\log}$, its stability in the reference dataset should be within $[\theta_{\log} + 5, \theta_{\log} + 10]$.

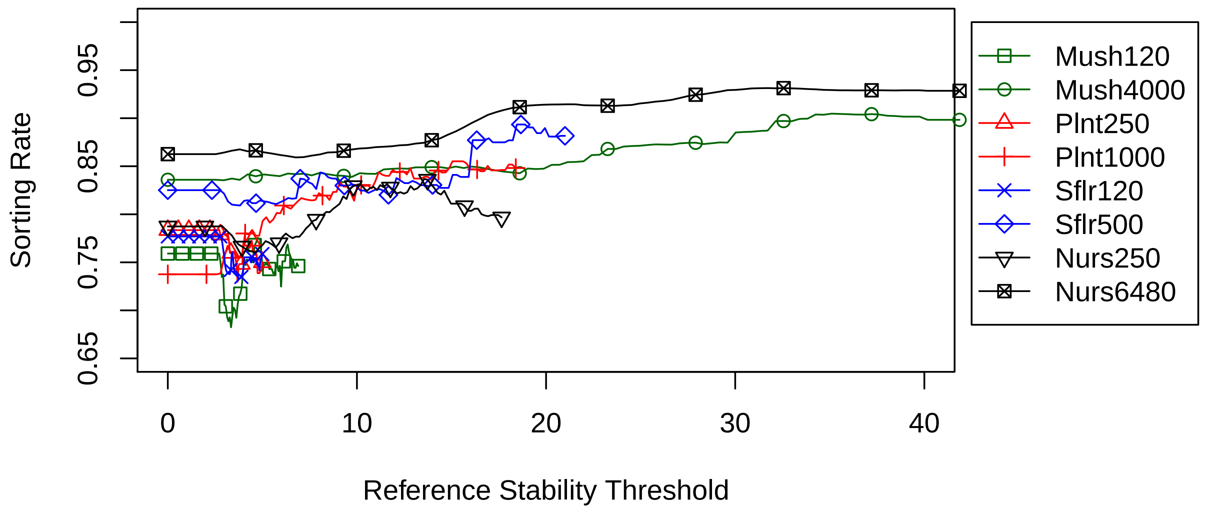
Let us consider the behavior of the stability thresholds w.r.t the size of the dataset. The dependency between the dataset size and the difference between stability thresholds in the reference ($\mathbb{K}_1$) and in the test ($\mathbb{K}_2$) datasets is shown in Figure 2.4. The x-axis corresponds to the dataset size, the y-axis corresponds to the stability threshold in K_1 such that 99% of concepts selected by this threshold are stable in the test dataset $\mathbb{K}_2$ with a certain threshold (1 or 5). For example, the line '5: Mush' corresponds to the stability threshold θ ensuring that all concepts having stability more than θ in $\mathbb{K}_1$ correspond to concepts having stability at least 5 in the test dataset $\mathbb{K}_2$. We can see that for large datasets the stability threshold is independent of the dataset, while for small datasets the diversity is higher. Here for large datasets the stability threshold should be set to 5–6 in a reference dataset in order to ensure that 99% of stable concepts have corresponding concepts in another dataset. This threshold should be set to 12 in order to ensure that 99% of stable concepts correspond to concepts having stability at least 5 in another dataset.

2.3.4 Stability and Ranking

Stability can be used for ranking concepts by decreasing its value. Thus, it is useful to study the linear order corresponding to the ranking relation. A way to study an order of an array ar is to compute its sorting rate r, i.e., the relative number of pairs in the array sorted in the ascending order: $r = 2 \cdot \frac{\{(i,j) | i<j \text{ and } ar_i \leqslant ar_j\}}{|ar| \cdot (|ar|-1)}$. A sorting rate equal to 1 means that the array is in



(a) Local sorting rate. The rate is computed for the test dataset concepts corresponding to the first 1000 stable concepts in the reference dataset with stability above a given threshold.



(b) Global sorting rate.

Figure 2.5: Sorting rate for different datasets.

the ascending order, while 0 means that it is in the descending order; the value 0.5 means that there is no order at all. Figure 2.5a shows local sorting rate (LSR), i.e., given a threshold the first 1000 stable concepts in $\mathbb{K}_1$ are taken and the sorting rate for the array of stabilities of the corresponding concepts in $\mathbb{K}_2$ is computed. This plot shows that for large datasets, the LSR is high (around 0.8–0.9) only for high stability thresholds in $\mathbb{K}_1$. For the smaller datasets the local sorting rate is around 0.7–0.8 for all thresholds. It means that stability preserves LSR only for the most stable concepts where the difference in stability between concepts is high enough, i.e., an error in order is less likely.

Finally, Figure 2.5b shows the global sorting rate (GSR) for different datasets, i.e., the sorting rate of stabilities in $\mathbb{K}_2$ for all concepts corresponding to the concepts selected by a threshold in K_1. We can see that the GSR for all datasets is slowly increasing and for small thresholds it is higher than the LSR. It shows that stability gives a global ordering of concepts, while the local ordering is not reliable for small thresholds.

2.4 Computing an Estimate of Stability

Table 2.3: Execution time for different steps on different datasets. `Size` is the number of concepts in the lattice; `Lattice` is the time for lattice computation with its structure; `Stab.` is the time for computing exact stability; `FCbO` is the time for computing the set of concepts by FCbO; `Freq.` is the frequency threshold applied for big datasets; `Est. Method` is the execution time for computing the estimate of stability by the estimate method; `Comb. Method` is the execution time for computing the estmate of stability be the combined method; the percentage here means that the program has been stopped after a certain amount of work; `MC calls` is the number of calls to the Monte-Carlo routine. All times are given in seconds.

Dataset	Size	Lattice	Stab.	FCbO	Freq.	Est. Method	Comb. Method	MC calls
Mush8124	$2.3 \cdot 10^5$	324	57	0.7	0	$2 \cdot 10^3$	$6 \cdot 10^3$	$6 \cdot 10^4$
Plnt1000	$2 \cdot 10^6$	45	10^4	78	0	181	446	$3 \cdot 10^3$
Chss100	$2 \cdot 10^6$	46	10^4	3.5	0	90	192	$2.3 \cdot 10^3$
SFlr1066	2988	< 1	< 1	< 1	0	0.7	11	284
Nurs12960	$1.2 \cdot 10^5$	245	5	0.2	0	425	$1.2 \cdot 10^3$	$4 \cdot 10^4$
Chss3196	$4.4 \cdot 10^6$	–	–	42	1000	$2 \cdot 10^4$	$3.5 \cdot 10^4$ (2%)	?
Plnt34781	$5.8 \cdot 10^6$	–	–	795	1750	$4.1 \cdot 10^5$	$4.6 \cdot 10^5$ (4.7%)	?

In this section we study the efficiency of computing various estimates of stability. Table 2.3 shows computation times for different methods and datasets. The lattice structure is built by our implementation of `AddIntent` (Merwe et al. 2004) and the set of concepts is computed by `FCbO` (Krajca et al. 2010)[4]. The datasets selected for experiments are the datasets of maximal tractable size (see Table 2.2) plus `Chess` and `Plants` with all objects. For the last two datasets the numbers of concepts are huge. Such datasets can be analyzed by finding only frequent concepts, i.e., concepts with significantly large extents. Although an incomplete set of concepts without lattice structure cannot be processed by the algorithm from Roth et al. (2008), stability can be estimated using formula (2.5), by Monte Carlo approach or their combination. For the cases where the estimation of stability takes too much time, the percentage of the processed concepts before termination is shown in the brackets. For the sake of efficiency, an estimation or an approximation of stability for a concept is stopped whenever it is clear that the concept is unstable i.e., stability is less than 3 in the logarithmic scale.

[4] The implementation is taken from `http://icfca2012.markuskirchberg.net`.

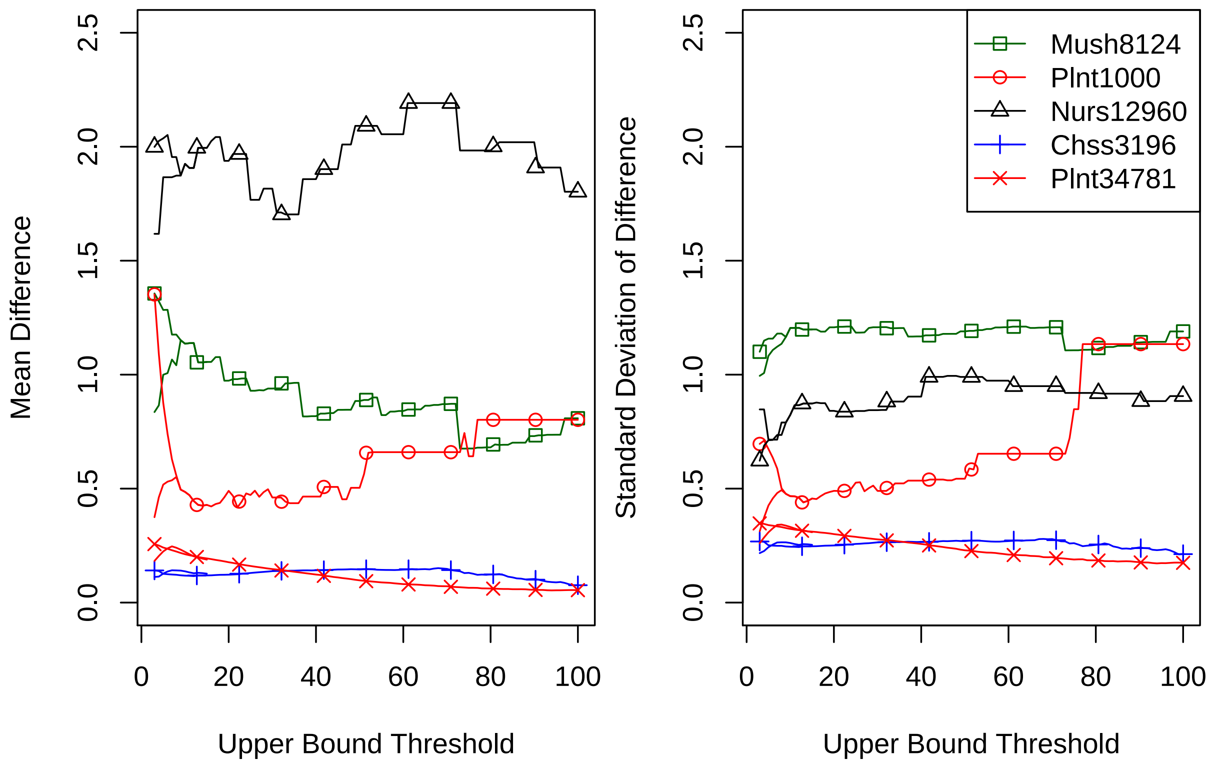


Figure 2.6: The mean and the standard deviation of the stability estimate interval

We can see that even the combined method is significantly slower than the bounding method and, hence, there is no reason to only work with the Monte Carlo method as it is slower and does not provide a better precision. Moreover, although the number of calls to Monte Carlo routine is small in the combined method, the computational efficiency of the stability estimate can dramatically decrease, making the usage of combined method unfeasible. The estimates are more efficient in terms of computational time for large lattices, i.e., lattices with a high number of concepts for one object from the context. We can see that in some cases the estimates for small lattices take much more time than the estimates for large lattices. This can be explained by the fact that the corresponding contexts contain many objects and attributes and that the computational efficiency of the estimates is highly dependent on the size of the context.

The tightness of the estimates is shown in Figure 2.6. On the x-axis the values of the upper bound stability threshold are plotted while on the y-axis the mean difference in the estimate are plotted. The plots are split in area of $[0, 10]$; the bottom line corresponds to the improvement achieved by additional use of Monte Carlo in the combined method. According to formula (2.5) Monte Carlo can give any improvements only in the case where stability upper bound is less than 13 (taking into account that for these datasets there are less than 100 attributes, and Monte Carlo parameters are in accordance with Example 2.3). In practice, however, this bound is even smaller (less then 10). These plots show that generally mean and standard deviation of the estimate difference do not change w.r.t. the upper bound, however they can significantly depend on the dataset. In our experiments it appears that the well-structured dataset (`Mush`, `Nurs`) has higher mean value then the unstructured ones, while the big datasets with only frequent concepts have low mean-values and standard deviations.

Taking into account (2.5), we can try to find stable concepts w.r.t. to one of the bounds.

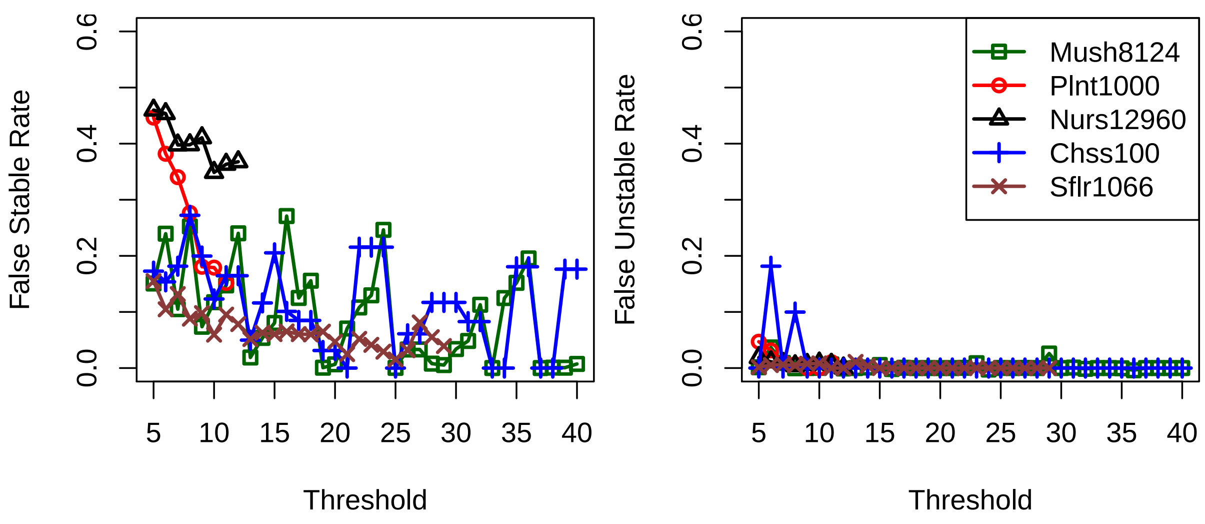


Figure 2.7: Over- and under- estimation rate for selecting stable concepts w.r.t. upper and lower bound of stability.

If we use upper bound than we never lose stable concepts, while we can mark some unstable concepts as stable. Oppositly, if we find stable concepts by the lower bound, we lose some stable concepts, while everything found is really stable. Figure 2.7 shows frequencies of false stable and false unstable discoveries. Here we can see that with the upper bound we can found up to 40% of additional concepts which are unstable. However the number of false stable discoveries can vary quite a lot along the stability threshold. While with lower bound most of unstable concepts are really unstable, i.e., we can lose normally only a few of stable concepts.

But having a stability bounds how well can we order the patterns w.r.t. stability? Figure 2.8 shows the losing rate of the estimates, i.e., the relative number of concept pairs which cannot be compared by the estimate. Normally, we lose less then 20% of concept relations independently from the threshold. In the interval $[0, 10]$ for the threshold we can find that the losing rate can be high. However, in this interval the Monte-Carlo approach can be applied, and, thus, can significantly reduce the losing rate.

2.5 Conclusion

In this paper we study concept stability and its estimates on different datasets. It is shown that stability computed in the logarithmic scale is more easy to interpret. Our experiments show that stability of a concept is correlated with the probability that the concept intent occurs in another dataset with high stability, i.e., it is an efficient measure for ranking patterns. However, independently of a dataset, as found experimentally, a concept should have a value of logarithmic stability greater than 5 in order to reflect any property of the population. Moreover, if the stability threshold in a reference dataset is θ, then the stability of the corresponding concept in another dataset is likely to be higher than $\theta - 10$ or even $\theta - 5$. We also remarked that stability is able to sort concepts in two independent datasets with nearly the same order by selecting concepts with stability greater than a certain threshold. However, the sorting rate of the first 1000 concepts from two independent datasets with stability above a certain threshold is high if the threshold is very high.

In the second part of this paper we showed that the introduced estimate is an efficient way

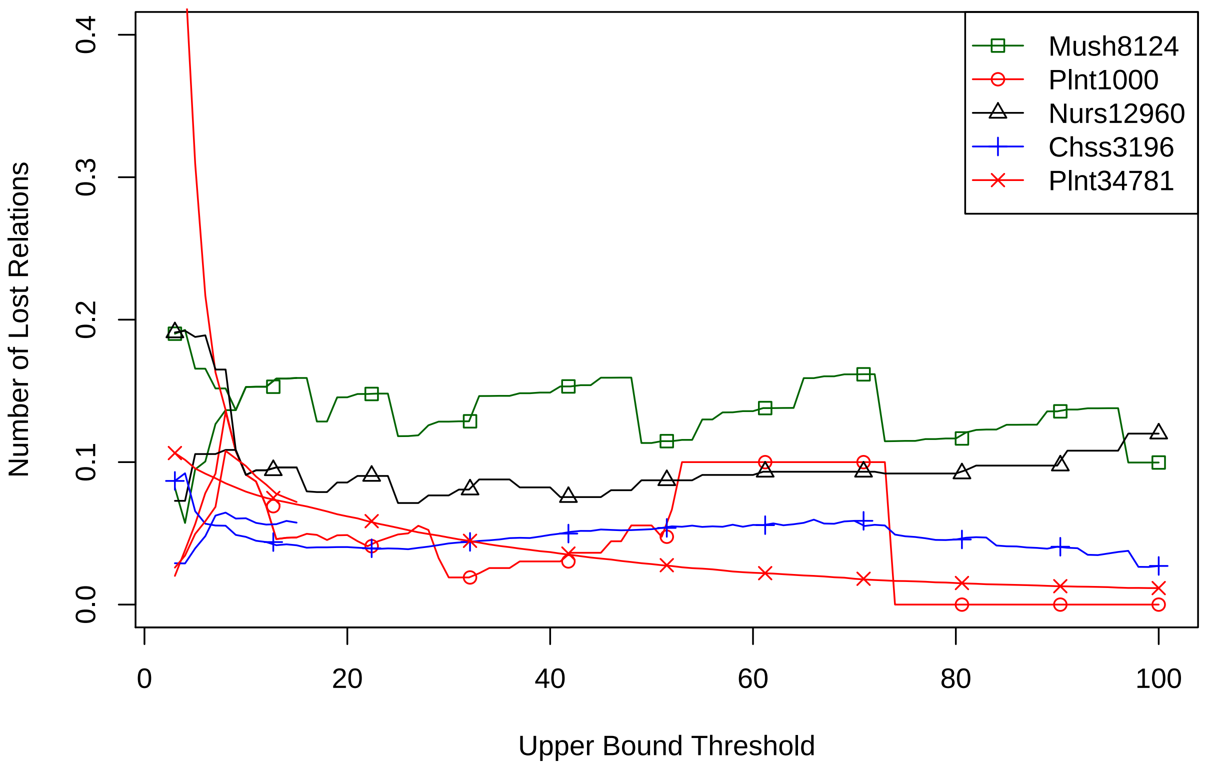


Figure 2.8: Losing rate of relations for stability estimate

for ranking concepts w.r.t. stability. It can be applied for an incomplete set of concepts and, hence, has more potential applications than the exact methods. The introduced approach can be meaningfully combined with a Monte Carlo method, providing better precision for weakly stable concepts by means of additional computational time. The precision and the sorting rate of the studied approximations are reasonably high and can be efficiently used for the stability computation.

However, the proposed methodology and experiments does not prove that stability is an “ideal” measure for concept selection. Indeed, in this chapter we have only checked the necessary condition for a measure to be considered as an interestingness measure, i.e., any measure should select similar concepts (or patterns) in datasets coming from the same general population. In the next chapter we discuss how different interestingness measures can be compared between each other.

Chapter 3

On Evaluating Interestingness Measures for Closed Itemsets

Contents

3.1 Introduction

One of the most important and frequent tasks in artificial intelligence is selection of the best option(s) among a huge set of possibilities. For example, in data mining one should often determine which patterns are of high interest. Usually patterns are evaluated w.r.t. a formal interestingness measure. Webb and S. Zhang (2005) says that measures cannot often reflect the true value of patterns because they "often depend on many factors that are difficult to formalize". *Are we able to evaluate how well a measure approximates an expert interest?*

One way to do that is to evaluate patterns with experts (Carvalho et al. 2005). In that case we evaluate how close the selected patterns approximate the expert knowledge. It is an expensive strategy requiring many experts for a domain. Thus, if one wants to compare measures on datasets from different domains the experiments become very expensive. Additionally such an experimentation requires a lot of time to be carried out.

Another way to evaluate patterns is to use artificial datasets (Zimmermann 2013), where the target patterns are known. The drawback of this approach is the interestingness of the artificial datasets w.r.t. real ones. Thus, *the first questios* discussed in this chapter is a methodology for comparison of interestingness measures for itemsets without involving experts or artificial datasets. Below we use indifferently "pattern" or "itemset", since our methodology can be applied for any kind of patterns, while itemset datasets are the most available and studied ones.

Our methodology *is based on* semi-supervised classification, where every data entry has a class label but labels are not directly involved into computation of a measure. We consider labels as an additional information to entry descriptions modeling domain knowledge or expert intent. The basic idea is to rank patterns with an interestingness measure and, then, find among them the patterns that are relevant to classification. And if a measure $\mathcal{M}_1$ is better than another measure $\mathcal{M}_2$ w.r.t. expert interest, i.e., $\mathcal{M}_1$ attributes more systematically high ranks to more relevant patterns, thus increasing the performance of a classifier based on $\mathcal{M}_1$.

This methodology can be applied when a measure does not rely on class labels. Then it can find itemsets that are suitable for expert interest (and not biased towards the classification task).

The second question discussed in this chapter is evaluation of some measures for closed itemset ranking. We evaluate leverage (Webb 2010) and stability (Kuznetsov 2007) measures that seem to be well adapted to closed itemset ranking. Indeed, stability can be only applied for closed itemsets and express the idea of pattern preservation under a permutation of the dataset, while leverage achieves, as it is shown later, its maximal values on closed itemsets and selects statistically significant patterns. We also introduce Δ-measure that comes from an estimate of stability (Buzmakov et al. 2014d). This measure is computed faster than stability. As it is widely used, the support of an itemset is used as a baseline measure. Finally, we also consider another leverage measure (rule leverage) which in contrast to the afore mentioned measures, relies on class labels. This rule leverage measure provides an idea of rule ranking measures w.r.t. itemset ranking measures.

Finally, we show that although there is no evident winner among stability and leverage measures, stability seems to be better on average. It is also shown that Δ-measure and stability have similar behaviour. But according to previous studies, Δ-measure is faster to compute (Buzmakov et al. 2014d). We can summarize the discussed *questions* of this chapter as follows:

1. Methodology for comparison of measures for itemset ranking.

2. Comparison of leverage and stability measures for closed itemsets.

This chapter is based on Buzmakov et al. (2014c) and is organised as follows. Section 3.2 discusses the related work. Then in Section 3.3 we introduce a running example and some preliminary information. Later in Section 3.4 we define and discuss Δ and leverage measures that are compared with stability. The next section describes the comparison methodology. And finally before the end of the chapter we discuss the experiments with the measures.

3.2 Related Works

Probably, the most elaborated area of mining interesting patterns is association rule mining. Most of the measures created in this area optimize a formal criterion using statistical methods (Masood and Soong 2013). Although there is a number of interestingness measures, there are only few comparisons between them (Azevedo and Jorge 2007; Carvalho et al. 2005). The main reasons for that are the diversity of formal criteria and the fact that no measure wins in all criteria. In (Carvalho et al. 2005) the authors evaluate different measures by means of expert interest. This is an important approach for pattern evaluation as it directly measures the relation between a formal measure and an expert interest. The drawback of this approach is the cost and diversity of datasets: for every dataset it is necessary to hire several experts which is costly. In (Azevedo and Jorge 2007), the authors evaluate measures by their performance in the classification task in a supervised setting, e.g., how confident is a rule concluding on a given class. In such case, the

Table 3.1: A toy dataset

	a	b	c	d	e	f	Label
g_1	x				x	x	+
g_2		x			x	x	-
g_3			x		x	x	+
g_4				x	x	x	-
g_5		x	x			x	+
g_6		x	x		x	x	?(+)
g_7	x		x			x	?(+)
g_8	x		x	x		x	?(-)

aim of an expert is expressed by labeling of a training set. This is in contrast with our approach which follows a semi-supervised setting, i.e., measures do not depend on labelling.

Another group of interestingness measures consists of measures created for itemset ranking. It is a less studied group. Several measures can be found in (Zimmermann 2013). Some of them are related to the distribution of partitions induced by every attribute from the considered pattern. Others are related to measures of association rule mining. Another approach introduces the measure of leverage that corresponds to the difference between frequency of an itemset and the maximal expected frequency based on subsets of the itemset (Webb 2010). Finally, some measures can be found in the domain of formal concept analysis (Belohlavek and Trnecka 2013; Kuznetsov 1990; Roth et al. 2008). Stability measure is one of the most interesting among them, because it is often used in domain specific areas where experts are often involved. Moreover, in contrast to all above mentioned measures for itemsets, stability is computed on object side making it possible to apply it for ranking any types of patterns, e.g., sequential patterns (Agrawal and Srikant 1995).

One comparison of interestingness measures of itemsets can be found in (Zimmermann 2013) where the authors introduce `Quest Generator`, i.e., a tool generating a dataset from a given set of "goal" itemsets in the presence of possible noise. Then, the interestingness measures can be evaluated w.r.t. their ability for finding the "goal" itemsets. For all artificial tests there is always a question about the degree to which generated datasets reflect real data. Thus, in this chapter we discuss an alternative approach for evaluating interestingness measures of itemsets on real datasets without involving experts. In the next sections we consider and compare these measures in details.

3.3 Running example and Preliminaries

Let as consider an example dataset shown in Table 3.1. This dataset contains 8 objects and 6 attributes labeled positive or negative by means of column "Label". The set of objects is divided into two groups: the train and the test sets. The train set of the dataset we formalize as a formal context $\mathbb{K} = (G, M, I)$ (see Definition 1.1), where $G = \{g_1, g_2, g_3, g_4, g_5\}$, $M = \{a, b, c, d, e, f\}$ and I is shown by the cross table.

Given a set of attributes or an itemset $Y \subseteq M$, *the image* of Y is the set of objects sharing Y, i.e., $(Y)'$. The cardinality of the image of Y is called *support* of Y, $\texttt{Supp}(Y) = |(Y)'|$, while the value $\sigma(Y) = \frac{\texttt{Supp}(Y)}{|G|}$ is called *frequency* of Y. In particular, given a formal concept $\mathcal{C} = (X, Y)$, X is the image of Y and the extent of $\mathcal{C}$. Correspondingly, $(\{e, f\})' = \{g_1, g_2, g_3, g_4\}$ is the image of $\{e, f\}$. We say that the support of the itemset $\{e, f\}$ is $\texttt{Supp}(\{e, f\}) = 4$ and its frequency is $\sigma(\{e, f\}) = \frac{4}{|G|} = 0.8$.

Definition 3.1. *An association rule between an itemset X and an itemset Y is denoted by $X \to Y$, where X is called the premise and Y is called the conclusion of the rule.*

Rule $X \to Y$ means that if the description of some objects from G contains X, then it contains Y. There are two measures attached to an association rule: support (or frequency) and confidence.

Definition 3.2. *The support of a rule $X \to Y$ is $\texttt{Supp}(X \cup Y)$ and frequency of the rule $X \to Y$ is $\sigma(X \cup Y)$.*

Definition 3.3. *The confidence of a rule $X \to Y$ is $\texttt{Conf}(X \to Y) = \frac{\texttt{Supp}(X \cup Y)}{\texttt{Supp}(X)}$.*

The support and the frequency of a rule show how often one can find the premise in the dataset, while a rule $X \to Y$ with a high confidence means that in most of the cases if an object description includes X it is likely to include Y. For example, in the dataset in Table 3.1 with the set of objects $G = \{g_1, g_2, g_3, g_4, g_5\}$ the confidence of the rule $\{e\} \to \{f\}$ is 1 because in every case when an object description contains e it contains also f, while $\texttt{Conf}(\{c\} \to \{e, f\}) = \frac{1}{2}$.

A common objective in data mining is search for interesting patterns, i.e. for interesting itemsets or rules, that are usually related to a task. Among those different tasks, there are classification, clustering and expert analysis of the result. Here we focus on searching for patterns that are likely to be interesting to an expert. In the next section we describe the existing approaches for mining interesting itemsets.

3.4 Itemset Interestingness Measures

For comparison stability and leverage measures are selected. Support is also included into the comparison as a baseline measure. Stability and its estimate have been discussed in the previous chapter. The upper bound of this estimate is based on the minimal difference between pattern support and support of its descendants in the lattice, i.e., Δ-measure corresponds to $\Delta_{\min}$ function in (Eq. 2.5):

$$\Delta(Y) = \min_{X \supset Y}(\texttt{Supp}(Y) - \texttt{Supp}(X)). \tag{3.1}$$

Δ-measure can be computed efficiently and the experiments show the interestingness of this measure. For example, Δ-measure of itemset $\{e, f\}$ is 3, because support of $\{e, f\}$ is 4 and any superset of $\{e, f\}$ has support at most 1. Similarly Δ-measure of $\{d, e, f\}$ is 1. For non-closed itemsets, Δ-measure is always zero.

The next measure that we evaluate is leverage for itemsets. For defining leverage we recall that a 2-partition of a set Y is a partition of Y in two subsets V and W and is denoted by $\texttt{Part}_2(Y) = (V|W)$. For example, the pair $(\{a, b, c\}, \{e, f\})$ is a 2-partition of the set $\{a, b, c, e, f\}$. Now we can define what the leverage of an itemset is.

Definition 3.4 (Webb (2010))**.** *The leverage of an itemset $Y \in 2^M$ is the difference between $\sigma(Y)$ and the maximal frequency that would be expected under assumption of independence of any subset of Y:*

$$\texttt{Lev}(Y) = \sigma(Y) - \underset{(V|W)=\texttt{Part}_2(Y)}{\texttt{argmax}} \sigma(V) \cdot \sigma(W), \tag{3.2}$$

where $\texttt{Part}_2(Y)$ is a 2-partition of Y.

According to the definition, leverage of an itemset can be applied to any itemset. If an itemset is non-closed then the leverage value is not zero and the next proposition holds.

Proposition 3.1. *The leverage of an itemset is not larger than the leverage of its closure,* $\texttt{Lev}(Y) \leqslant \texttt{Lev}(Y'')$.

Proof. Frequency can only decrease with addition of an attribute, i.e., $(\forall X \subseteq Y)\sigma(X) \geqslant \sigma(Y)$. Frequencies of an itemset and its closure are equal, $\sigma(Y) = \sigma(Y'')$. Given an itemset X, a 2-partition of its closure $\texttt{Part}_2(X'') = (V|W)$ induces the 2-partition of X, i.e., $(V \cap X|W \cap X)$ is a 2-partition of X. Then,

$$\begin{aligned}\texttt{Lev}(Y'') = \sigma(Y'') - \underset{(V|W)=\texttt{Part}_2(Y'')}{\texttt{argmax}} \sigma(V)\cdot\sigma(W) = \\ = \sigma(Y) - \underset{\substack{(V|W) = \texttt{Part}_2(Y) \\ (P|Q) = \texttt{Part}_2(Y''\backslash Y)}}{\texttt{argmax}} \sigma(V \cup P)\cdot\sigma(W \cup Q) \geqslant \\ \geqslant \sigma(Y) - \underset{(V|W)=\texttt{Part}_2(Y)}{\texttt{argmax}} \sigma(V)\cdot\sigma(W) = \texttt{Lev}(Y).\end{aligned}$$

Thus, leverage maximizes its value on closed itemsets, and, consequently, we can compute it only for closed itemsets. □

Let us consider an example. In order to compute leverage of the itemset $\{e, f\}$ we need to find all its 2-partitions. There is only one 2-partition $(\{e\}|\{f\})$. The frequencies are $\sigma(\{e, f\}) = 0.8$, $\sigma(\{e\}) = 0.8$, $\sigma(\{f\}) = 0.8$. Thus, $\texttt{Lev}(\{e, f\}) = 0.8 - 0.8^2 = 0.16$. For itemset $\{d, e, f\}$ we have three 2-partitions: $(\{e\}|\{d, f\})$, $(\{d\}|\{e, f\})$ and $(\{f\}|\{e, d\})$. The frequencies are $\sigma(\{d, e, f\}) = 0.2$, $\sigma(\{e\}) \cdot \sigma(\{d, f\}) = 0.8 \cdot 0.2 = 0.16$, $\sigma(\{d\}) \cdot \sigma(\{e, f\}) = 0.2 \cdot 0.8 = 0.16$, $\sigma(\{f\}) \cdot \sigma(\{d, f\}) = 0.8 \cdot 0.2 = 0.16$. Thus, $\texttt{Lev}(\{d, e, f\}) = 0.2 - 0.16 = 0.04$.

The leverage of an itemset is based on the notion of leverage of a rule. Hereafter, we use leverage of a rule in our comparison as a base line and, thus, we need to provide its definition.

Definition 3.5. *The leverage of a rule is defined as follows*

$$\texttt{Lev}(X \to Y) = \sigma(X \cup Y) - \sigma(X) \cdot \sigma(Y) \tag{3.3}$$

In this work rule leverage is applied to rules of the form $X \to \{\mathcal{C}\}$, where $\mathcal{C}$ is a class label in classification. Let us consider Table 3.1, where the target class is given by column "class". In order to define rule leverage of $\{e, f\} \to \{+\}$, first, we should find the frequencies: $\sigma(\{e, f, +\}) = 0.6$, $\sigma(\{e, f\}) = 0.8$, $\sigma(\{+\}) = 0.6$. Thus, $\texttt{Lev}(\{e, f\} \to \{+\}) = 0.6 - 0.8 \cdot 0.6 = 0.12$.

We are now ready to introduce our methodology for evaluating interestingness measures.

3.5 Evaluation Methodology

In this work the classification task is used to estimate the interestingness of measures for itemset selection w.r.t. expert interest, by measuring the precision and recall of classifiers built with these measures.

The intuition behind the usage of classification for evaluating measures is the following. If an itemset is of high interest for an expert, then it should reflect basic dependencies in a domain. Thus, the performance of this itemset in classification should be better than an arbitrary itemset. Consequently, systematic good performances may mean that a measure is more suitable to find itemsets of high interest. Accordingly, the evaluation methodology consists of the following steps:

1. A dataset $\mathcal{D}$ is selected.
2. The dataset $\mathcal{D}$ is divided into training and test sets by random sampling 100 times. A training set contains 90% of the objects with class labels (but at most 1000 objects which is a limit of `Magnum Opus` demo (Webb 2007) that is used for leverage computation). The test set contains the rest of the objects.
3. One target class label $\mathcal{C}$ is selected.
4. One target measure $\mathcal{M}$ is selected.
5. A training set built at step 2 is used to find itemsets and rank them w.r.t. the measure $\mathcal{M}$. However, during the search, class labels for objects are ignored.
6. Among the whole set of itemsets, the emerging patterns for class $\mathcal{C}$ are selected from the training set (Dong and Li 1999). An emerging pattern is an itemset that is a characteristic of one class, i.e., it covers objects mostly labelled with the same class, w.r.t. a threshold θ. These emerging patterns are assumed to be good for classification purposes. The idea of emerging patterns is borrowed from Kuznetsov (1996), where emerging patterns are called hypotheses. Let say that there are N emerging patterns.
7. From these N emerging patterns we form N classifiers based on the first k patterns (with $k \leqslant N$). Each classifier works in the following way. Given a set of patterns $\{p_1, \cdots, p_k\}$, the classifier attaches the label $\mathcal{C}$ to any object whose description contains p_i for $i \in [1, k]$.
8. We compute precision and recall for these N classifiers in the test set. Then we interpolate 21 points of the form (p, r) where p stands for precision and r stands for recall, where $r \in \{0, 0.05, \cdots, 0.95\}$. These 21 points yield a curve.
9. Steps 6–8 are repeated for every pair of training and test sets. An average curve is computed for all the curves based on the pairs of training and test sets.
10. The area under this averaged curve is computed providing a numerical quality of the measure $\mathcal{M}$ in dataset $\mathcal{D}$ w.r.t. class $\mathcal{C}$.
11. We repeat steps 3–10 for all classes in $\mathcal{D}$ and all measures.
12. We repeat steps 1–11 for all available datasets.

Thus, each measure is evaluated for every class label and for any division of a dataset. The precision and recall in step 8 are computed in a standard way, i.e., in terms of true/false positives/negatives where the precision is $\mathtt{Pr} = \frac{\mathtt{TP}}{\mathtt{TP}+\mathtt{FP}}$ and the recall is $\mathtt{R} = \frac{\mathtt{TP}}{\mathtt{TP}+\mathtt{FN}}$.

But how can one select the threshold θ? This is a tricky question. On the one hand, it is necessary to take the high θ in order to force a measure to select itemsets relevant for the classification. Thus, datasets where there are no patterns with high θ are not adapted for the methodology. On the other hand, it is necessary to have a sufficient number of emerging patterns to capture differences between measures. Here, we posed $\theta = 90\%$, i.e., at least 90% of objects in the image of a pattern are in the same class. However, the selection of an ideal θ is still an open question.

Finally, any emerging pattern X for class $\mathcal{C}$ can be written as an association rule $X \rightarrow \{\mathcal{C}\}$. In particular Azevedo and Jorge (2007) study different association rule measures by means of classification. Such kind of measures rely on class labeling and thus they are biased for classification task. In contrast in our work measures evaluate itemsets and after that a labeling is introduced. Thus, our approach appears to be closer to the expert interest. However, it is also possible to introduce the interestingness measures for rules in this framework as a baseline for evaluating interestingness measures of itemsets. We decided to add the leverage interestingness measure for rules, see (Eq. 3.3). The results for rule leverage measure are provided only as a baseline taken into account the above comment.

3.5.1 Application to the Running Example

Let us consider this methodology on the example in Table 3.1. We have a dataset containing 8 objects (step 1). This dataset is divided into training set, $Tr = \{g_1, g_2, g_3, g_4, g_5\}$, and test set $T = \{g_6, g_7, g_8\}$ (step 2). The target class label is $\mathcal{C} = +$ (step 3). The target measure is Δ-measure (step 4). In this example we consider an itemset to be an emerging pattern if 50% of objects in its image are labeled with the target class. Thus, we have five closed emerging patterns: $\{e, f\}$, $\{c, f\}$, $\{a, e, f\}$, $\{c, e, f\}$ and $\{b, c, f\}$ (step 6). The corresponding Δ-measures are 3, 1, 1, 1, 1. Thus, they are well sorted and we are ready to construct classifiers (step 7) and evaluate their performance (steps 8 and 9).

The first one is only based on $\{e, f\}$. This itemset is only included in the description of g_6, consequently only g_6 should be classified positively. The precision and recall of this classifier are 1 and 0.5. The next classifier is based on $\{e, f\}$ and $\{c, f\}$. The description of g_6 includes $\{e, f\}$ and, thus, it should be classified positively with the second classifier. The descriptions of g_7 and g_8 include $\{c, f\}$ and, thus, they should be also classified positively. The precision and recall of the second classifier is $\frac{2}{3}$ and 1. After repeating this with all emerging patterns we can interpolate the value of precision for every recall of the form $0.05 \cdot K$, where $K \in \{1, 2, \cdots, 20\}$. Doing this several time for every division of the dataset we can obtain the averaged precisions corresponding to these recalls. Finally, we can compute the area under the average curves providing a numerical quality of the measure on this dataset.

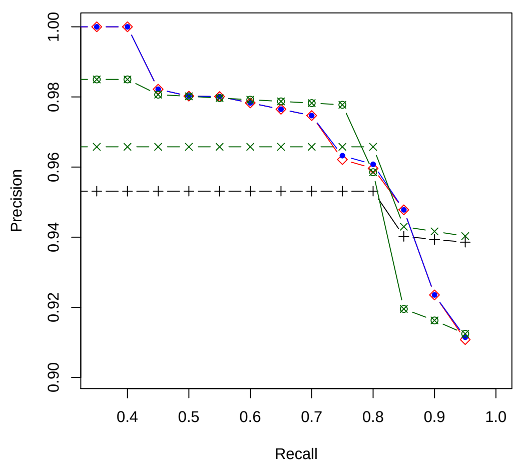
3.6 Experiment

The experiments are carried out with public available datasets from UCI (Frank and Asuncion 2010): `Mushroom`[5], `Congressional Voting Records`[6], `Nursery`[7] datasets. All datasets contain emerging patterns and thus we can apply our methodology. In the experiments we have compared 4 interestingness measures for itemset ranking, i.e., support, stability (Eq. 2.1), Δ-measure (Eq. 3.1) and leverage (Eq. 3.2), as well as a measure for association rule ranking, i.e., rule leverage (Eq. 3.3). In this chapter we do not discuss the computational efficiency of different measures. Thus, we only mention that computations take less than a minute per experiment in every case.

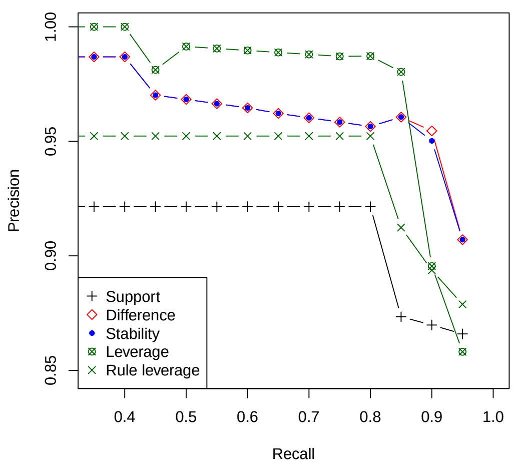
Let us consider one dataset deeper. Figure 3.1 shows the results of two experiments on `Mushroom` dataset. Figure 3.1a shows precision and recall for predicting the class of edible mushrooms, while Figure 3.1b corresponds to poisonous mushrooms. Every line in this figure corresponds to a measure. Every point corresponds to a precision-recall pair at the end of step 9 of the proposed methodology.

In this figure we can see that stability and Δ-measure have nearly the same behaviour. It is the case for every tested dataset. The second point is that the support measure is not the

[5]http://archive.ics.uci.edu/ml/datasets/Mushroom
[6]http://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records
[7]http://archive.ics.uci.edu/ml/datasets/Nursery



(a) Eatable mushrooms



(b) Poisonous mushrooms

Figure 3.1: Precision and Recall for mushroom dataset for classifiers built with different interestingness measures.

Table 3.2: The surface under the ROC-diagram for different datasets different target classes and different measures. The best measure in a row is bolded.

Dataset	Class	Support	Δ-measure	Stability	Itemset Lev.	Rule Lev.
Mushroom	Poisonous	0.890658	0.945881	0.945665	**0.956895**	0.919898
Mushroom	Eatable	0.927239	0.953793	**0.953941**	0.946683	0.938007
Vote	Democrat	0.865279	0.862507	0.8645	0.904433	**0.953708**
Vote	Republican	0.883406	**0.921093**	0.921004	0.884818	0.883406
Nursery	Not Recommended	**0.975**	**0.975**	**0.975**	**0.975**	**0.975**
Nursery	Priority	**0.78503**	0.743039	0.725221	0.605405	0.525
Nursery	Special Priority	**0.875556**	0.850174	0.851127	0.699788	0.639793

best one for pattern selection, which is not surprising. The unexpected result here is that the rule leverage does not perform well. Logically it should be the best one because it is the only tested measure that can access the label information in the dataset. One explanation can be that the statistical significance (at least of the rule leverage type) is not directly related to the interestingness of an itemset to real patterns.

In Table 3.2 the numerical qualities for every dataset and every class label is given. Every column corresponds to a measure and every line corresponds to a combination of a dataset and a class label. First, the Δ-measure and stability have a similar behaviour, and the numerical quality has nearly the same value in every experiment. Second, although there is no evident winner between stability and itemset leverage and they often have a comparable result, but on `Nursery` dataset stability has a significantly better result.

3.7 Conclusion

In this chapter we have discussed a methodology for evaluating interestingness measures for closed itemset selection. The proposed methodology has been applied to compare leverage, stability and

Δ-measure. Although stability has a slightly better behaviour than leverage we cannot conclude that one is better than the other. It is also shown that stability and Δ-measure have very similar behaviour, but Δ-measure is computed faster.

It should be noticed that stability and Δ-measure have an important property that they can be applied to any kind of datasets as soon as support can be computed, e.g., datasets of sequences or graphs. This is not, for example, the case for leverage. Since Δ-measure is faster to compute, we should conclude that Δ-measure is the most convenient measure providing the same quality as stability and leverage.

Thus, Δ-measure is a criteria for pattern selection in a large dataset. In the next chapter we apply this measure to a large dataset of mutagenic compounds. We find how well it reduce the pattern space and what kind of patterns it selects.

Chapter 4

Formal Concept Analysis for Big Datasets of Complex Data. *Discovering Structural Alerts for Mutagenicity using Stable Emerging Molecular Patterns.*

Contents

4.1 Introduction

In this chapter we consider how stability can be used to find relevant structural alerts in a big dataset on mutagenicity. In the pharmaceutical industry, it is widely recognized that early safety evaluation of candidate molecules is needed before making significant investments of time and resources (Muster et al. 2008; Pearl et al. 2001). To this aim, the notion of predictive toxicology, which includes the application of computer technologies to detect relationships that connect chemical structures and toxicological activities in large biological and chemical datasets, is very appealing. The advantages of *in silico* techniques in comparison with *in vitro* and *in* *vivo* techniques can be summarized by their higher throughput, their cost effectiveness, and their potential to reduce the use of animals. In a regulatory framework, the use of toxicity prediction tools is encouraged to improve prioritization of data requirements and risk assessment not only for pharmaceuticals (Kavlock et al. 2008; Kruhlak et al. 2007) but also for other chemical products like cosmetics and agrochemicals (Commission 2007; Rogers 2003). *In silico* prediction methods can roughly be classified into two categories: knowledge-based expert systems and data driven models. On the one hand, knowledge-based expert systems like Derek Nexus (Ridings et al. 1996; Sanderson and Earnshaw 1991), HazardExpert (Smithing and Darvas 1992) and OncoLogic (Lai and Woo 2005; Lai, Woo, et al. 1995) do not discover new associations between chemicals and toxicity but rather formalize the knowledge of human experts and the scientific literature. On the other hand, data driven models like MultiCASE (Klopman 1984), Topkat (*ADMET and Predictive Toxicology* n.d.), LAZAR (Helma 2006) and PASS (Lagunin et al. 2000) analyze existing data, identify chemical features that are relevant for the observed toxicological endpoints, and automatically build statistical models. The definition of structural alerts corresponds to one of the most interesting approach of predictive toxicology since it defines the key features of a molecule that are required to interact with a biological system and initiate a toxicology pathway. Its main advantage is the identification of chemicals with common mechanism of action. The Tennant and Ashby's set of structural alerts is a well-known example of such associations (Ashby and Tennant 1991). This set defines structural alerts for DNA reactivity based on the analysis of *in vitro* mutagenicity and *in vivo* carcinogenicity data. Other researchers largely extended this set of alerts and to date, one of the most advanced lists for evaluating the mutagenic and carcinogenic potential of chemicals has been proposed by Benigni and Bossa (2011). This list has been implemented as rules in knowledge-based expert systems like Toxtree (Benigni 2008) and the OECD QSAR Toolbox (K. v. Leeuwen et al. 2009). However, some limitations have been reported in the literature (Guzelian et al. 2005; Valerio 2009): i) the updating of the knowledge base is a very time consuming process since it requires strong investment of domain experts and a detailed analysis of the scientific literature, ii) the expert opinion can sometimes be prone to subjectivity leading to inaccuracies, and iii) a negative response cannot be interpreted as a lack of toxicity but simply as a lack of information with respect to the molecule of interest.

The evolution of artificial intelligence and data mining tools should answer some limitations mentioned above, and particularly the time and efforts needed to identify new structural alerts, sometimes beyond the limits of human perception. The computation of the frequency of a chemical substructure in a dataset is often at the core of the process for the definition of its toxicological relevance. The rationale for using a frequency constraint is that it is unlikely to generalize on a substructure which has been observed on a few chemicals. However, the algorithms that enumerate frequent substructures from a set of molecules, like Gaston (Nijssen and Kok 2005) or gSpan (Yan and Han 2002), often lead to the generation of too many such substructures.

To limit the number of generated substructures, methods have been developed in the recent years for finding representative and significant structural patterns. For example, from a mutagenicity dataset, Kazius, McGuire, et al. (2005) determined the statistical association of each proposed frequent substructure with mutagenicity, the association expressed as the p-value resulting from a statistical test. Even if it relies on manual annotations, this work has enabled the development of 29 approved toxicophores. Recently, Ahlberg et al. (2014) have proposed a framework that automatically derives potential structural alerts; it also relies on the p-value of a statistical test to select the significant substructures. Even if the computation does not exhaustively enumerate all the possible substructures, it constitutes a fast and automated way for deriving toxicophores. These two works do not directly compute significant molecular substructures: they compute significant atom signatures (Faulon, Churchwell, et al. 2003; Faulon, Visco, et al. 2003) from which the significant substructures are derived.

Helma et al. (2004) used MOLFEA, a molecular feature miner, to discover linear molecular fragments (chains) that occur with a higher frequency in mutagenic compounds than elsewhere. MOLFEA uses a level-wise algorithm (Mannila and H Toivonen 1997) enabling the extraction of linear substructures which are frequent in the set of the mutagens while infrequent outside of it. However, the restriction to linear substructures disables a direct extraction of the fragments containing a branching point or a cycle. This technical limitation has been overtaken thanks to the design of general frequent subgraph mining algorithms such as Gaston (Nijssen and Kok 2005) or gSpan (Yan and Han 2002). Kazius, Nijssen, et al. (2006) have applied this methodological advance to extract emerging fragments; their work has led to the discovery of six new structural alerts.

Emerging pattern mining is a contrast data mining technique (Dong and Bailey 2013) introduced by Dong and Li (1999). The emerging constraint captures differentiating characteristics between two classes of data and was first applied in chemoinformatics by Auer and Bajorath (2006). They introduced the notion of emerging chemical patterns (ECPs) as a novel approach to molecular classification. To describe the molecules, they did not use molecular graphs but a set of physicochemical and molecular properties. The *jumping emerging patterns* (JEPs) correspond to a subset of the emerging patterns: a JEP denominates a pattern that is sufficiently present in one class and absent from the other. Closed JEPs, called JSM-hypotheses, were used in predictive toxicology by Blinova et al. (2003): an itemset representation was used, with items staying for particular molecular fragments.

Sherhod, Gillet, et al. (2012) and Sherhod, Judson, et al. (2014) applied the notion of emerging patterns for the identification of structural features contrasting mutagenic with non-mutagenic compounds. An emerging pattern corresponds here to a conjunction of structural features, a structural feature being either a functional group, a ring fragment or an atom-pair (Carhart et al. 1985). The functional groups and the ring fragments are automatically computed from a molecular dataset by keeping only the most meaningful parts of the molecules. The method has also been successfully used to investigate clusters of mutagenic compounds and to implement new structural alerts in the knowledge base of the Derek Nexus expert system (Coquin et al. 2015).

In this chapter we introduce a method that computes the conjunctions of molecular fragments whose frequency of occurrences in a dataset is sufficiently discriminative between different subgroups of molecules (e.g., mutagens and non-mutagens) to be of interest. The method operates directly from the molecular graphs: it automatically enumerates the molecular fragments which are sufficiently frequent to be considered. It is based on a graph-based mining framework for extraction of emerging patterns from a dataset of molecules that has been recently introduced (Cuissart et al. 2013; Lozano et al. 2010; Poezevara et al. 2011). Although this framework is not originally described in terms of FCA, it is not hard to introduce it by means of FCA and we proceed in this way in the rest of this chapter. We apply this method to mine a mutagenicity dataset collected by Hansen et al. (2009). Our first concern would be a study on how stability could improve the aforementioned framework. Thus, we extract a so called stable emerging molecular patterns.

The computational method is detailed in Section 4.2. The main results of an expert analysis demonstrate the practical interest of the computational method: the extracted structural patterns constitute an efficient basis for a process of chemical knowledge discovery. The content of this chapter is based on Métivier et al. (2015).

4.2 Materials & Methods

4.2.1 Molecular patterns

Given a dataset of molecules where the structure of a molecule is given by its usual graph model, we proceed in a similar way as it was done in Chapter 1. In particular, we first find a set of frequent *molecular fragments*, i.e., a set of graphs that are subgraphs of a sufficient number of molecular graphs from the dataset. Second, a formal context (G, M, I) is built by putting the set of molecules as a set of objects G, the set of frequent molecular fragments as a set M, and a pair (g, m) belongs to a set I if and only if the molecule g includes the fragment m as a substructure (or a subgraph). Then, the corresponding concept lattice can be found. The intent of a concept is referred as a *molecular pattern*, while the extent of a molecular pattern is the extent of the corresponding concept; the *length* of a molecular pattern designates the number of fragments it contains. We say that a molecular pattern $p \subseteq M$ is *included* into a molecular pattern $q \subseteq M$, if $p \subseteq q$, while the pattern q *covers* the pattern p.

We should notice that any intent in this concept lattice is a closed molecular pattern. In particular it means that any molecular fragment in this pattern is a closed molecular pattern (Kuznetsov and Samokhin 2005). Moreover, if a molecular fragment is included into a molecular pattern, then any subgraph of this molecular fragment is also included in to the molecular pattern. Thus, any closed molecular pattern can pruned by removing any fragment that is included into another fragment from this molecular patter (Poezevara et al. 2011). For the sake of simplicity we work with pruned molecular patterns in the rest of the paper.

4.2.2 Emerging molecular patterns

Since the entire set of molecular fragments is very large, it leads to a huge number of molecular patterns. To select meaningful patterns, one may consider a pattern only if it occurs sufficiently often in the molecular dataset. However, a combination of frequent fragments does not necessarily lead to a relevant molecular pattern. For example, a molecular pattern made with the basic molecular fragment $C-C$ does not carry alone any significance for studying important properties such as mutagenicity or acute toxicity.

For automatically discovering structural alerts, it is highly appropriate to look for structural changes between different groups of molecules (e.g., between mutagens and non-mutagens). In particular, given a set of molecules, a molecular pattern which sufficiently occurs within the molecules of the given set and whose occurrences are significantly more frequent in the mutagens than in the non-mutagens stands as a potential structural alert related to the mutagenicity. The notion of a frequent emerging molecular pattern embodies this natural idea by using the growth-rate measure. When a chemical dataset is partitioned between targeted molecules and non-targeted ones (also named "classes"), the *growth-rate* of a pattern p, denoted $\rho(p)$, is defined as the ratio between the frequency of p in the targeted molecules over its frequency outside the targeted molecules (Dong and Li 1999). The growth-rate of a molecular pattern is obtained by dividing its frequency in the mutagens by its frequency in the non-mutagens. A *Jumping Emerging Pattern (JEP)* denotes a pattern which has the noticeable property to occur solely in molecules of the targeted class. By default, the growth-rate of a JEP is denoted by the infinity symbol, ∞. We remind that in Chapter 1 we worked with JEPs. A frequent emerging molecular pattern denotes a molecular pattern which fulfills two constraints: a frequency sufficiently high for founding a further inductive usage and a growth-rate value sufficiently high for indicating a potential structural alert. Thus, being a frequent emerging molecular pattern depends on the settings of both a minimum frequency threshold and a minimum growth-rate threshold.

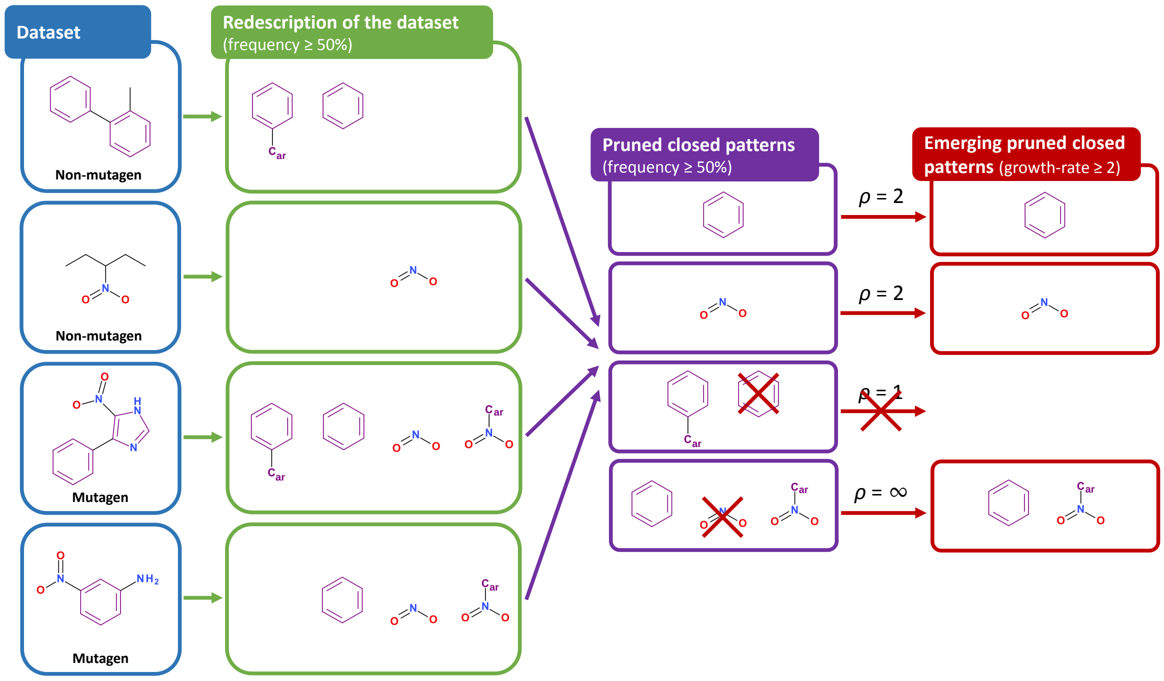

Dataset
Non-mutagen
Non-mutagen
Mutagen
Mutagen
Redescription of the dataset (frequency ≥ 50%)
Pruned closed patterns (frequency ≥ 50%)
Emerging pruned closed patterns (growth-rate ≥ 2)
ρ = 2
ρ = 2
ρ = 1
ρ = ∞

Figure 4.1: Mining workflow: from a molecule dataset to emerging molecular patterns

Illustration. Figure 4.1 illustrates the notion of a frequent emerging molecular pattern. As an input, this example considers the learning dataset of four molecules depicted on the left of Figure 4.1, a minimum frequency threshold set to 50% and a minimum growth-rate threshold is set to 2.

Since any frequent emerging molecular pattern is made up of frequent molecular fragments only(Poezevara et al. 2011), it is sufficient to describe the molecules using the frequent fragments they contain. In the example a fragment that occurs twice or more among the four molecules is a frequent fragment: it results in four frequent fragments. These frequent fragments constitute the set of attributes used in an intermediate description of the molecules: every molecule of the dataset is described by the frequent fragments it contains. From this binary description, one is able to generate every pattern which is frequent and a closed pattern. So as to retain only meaningful pieces of information, these patterns are pruned: any fragment is removed from a pattern as soon as it is contained into another fragment of the pattern. In the example, the pruning step leads to the removal of the phenyl group from the third pattern and to the removal of the nitro fragment from the fourth pattern.

In the example, the minimum growth-rate threshold being set to 2, a frequent pattern that is at least twice more frequent among the mutagens than among the non-mutagens is an emerging pattern. Among the four frequent patterns, three are considered as frequent emerging molecular patterns as their growth-rate, denoted by ρ, exceeds 2: each represents a conjunction of fragments which is frequent and whose occurrences in the learning set are discriminative w.r.t. the threshold.

4.2.3 Stable emerging patterns

The number of frequent emerging patterns is usually very high and many of them are not significant and may result from artefacts of the dataset. Here we use stability in order to select the most relevant molecular patterns (Kuznetsov 2007). Stability was discussed in Chapters 2 and 3.

We remind that stability of a pattern p measures a relative number of subsets of the extent of p, i.e., subsets of molecules where p occurs, such that p is closed in these subsets.

Let us return to the example in Figure 4.1. In this example, the phenyl group alone is a molecular pattern. To compute stability of this molecular pattern, one can reason as follows. From the initial dataset of 4 molecules, 16 different subsets can be generated, including the empty set and the whole initial set. Among these 16 subsets, the phenyl group is considered as a molecular pattern if the subset fulfills two conditions. First, the phenyl group has to be considered as a fragment: it has to appear in at least one molecule of the subset. This condition is not fulfilled by two subsets: the empty set and the singleton subset with the second molecule alone. Second, the first molecule and the fourth molecule must be elements of the subset otherwise the phenyl group is not closed anymore. Indeed, without the fourth molecule in a subset, the closed molecular fragment should be a phenyl group associated by a single bond to an aromatic carbon and not a phenyl group alone. Similarly, without the first molecule in a subset, the (closed) pattern should be a phenyl group together with the fragment cN(=O)O. This second condition is fulfilled by 4 subsets. Thus, since the phenyl group is considered as a molecular pattern in 4 subsets generated from the initial dataset, its stability is equal to $\frac{4}{16} = 0.25$.

4.2.4 The computational method

This section details computation of stable emerging patterns (SEPs) from a set of molecules partitioned into two subsets (e.g., the mutagens and the non-mutagens). The method is a straight extension of the work introduced by Poezevara et al. (2011). It relies on three main steps: the computation of the closed frequent fragments, then the computation of the frequent emerging pruned closed patterns, and finally the selection of the most stable emerging patterns.

First, the frequent molecular fragments are computation by mining the training set of molecules provided as the input; the operation relies on a minimum frequency threshold. `Gaston` is used to mine the chemical graphs [8]. The efficiency of `Gaston` mainly relies on the adoption of the *quick-start principle* (see Nijssen and Kok (2005) for more details). In our computation, *a fragment never contains an incomplete chemical ring.* To exclude molecular fragments containing incomplete chemical ring, we use a similar approach as Borgelt (Borgelt 2006). As a pretreatment, any edge of a molecule which is included in at least one ring is tagged as “in a ring”. As a filter, every frequent fragment output by `Gaston` is tested: when the fragment has at least one of its edges which is not in a ring and which is tagged as “in a ring” then the fragment is discarded from the list of the frequent molecular fragments.

Then, the emerging patterns are found by postfiltering of the set of frequent molecular patterns. Finally, from the set of emerging molecular patterns only stable patterns are selected. We remind that stability is hard to compute (Kuznetsov 1990), and, thus, we apply here the bounding estimate of stability discussed in Chapter 2. This estimate is polynomial in contrast to stability itself and hence it is able to process a huge set of emerging molecular patterns found in the previous step.

[8] http://www.liacs.nl/~snijssen/gaston/

4.3 Experiments and Discussion

4.3.1 Datasets

The Hansen Dataset

The mining process has been applied to a publicly available benchmark dataset reported by Hansen et al. (2009) The dataset consisted of 6 512 compounds resulting from the compilation of Ames mutagenicity data described in CCRIS (CancerInformatics.org.uk n.d.), Helma et al. (2004), Kazius, McGuire, et al. (2005), Feng et al. (2003), VITIC (Judson et al. 2005), and GeneTox databases (Fda.gov n.d.). To be classified as Ames positive, i.e mutagen, a compound had to significantly induce a revertant colony growth in at least one of the strains of *Salmonella typhimurium*. (Ames et al. 1973) Even if the dataset was already pretreated to remove duplicate structures and inorganic molecules, we cleaned the chemical data using Pipeline pilot (Accelrys Inc., San Diego, CA, USA) and ChemAxon (Chemaxon Ltd., Budapest, Hungary) components. The additional curation steps consisted in the normalization of specific chemotypes (e.g., nitro group, organophosphate moiety...), the conversion of the structures to their aromatic form, and the addition of hydrogens on the heteroatoms. It resulted in a well-balanced dataset containing 3503 mutagenic and 3009 non-mutagenic compounds.

The study of Hansen et al. (2009) uses a particular 5-fold cross-validation scheme. The authors of this paper have partitioned the dataset into six parts. The first part gathers all compounds of the dataset which are verifiable according to Derek Nexus (Ridings et al. 1996; Sanderson and Earnshaw 1991) or MultiCASE (Klopman 1984); the leftover compounds have been distributed into the five other parts (with the same ratio of mutagens and non-mutagens). Within each of the five folds of cross-validation, the training set corresponds to the union of the first part with four of the five other parts and the test set is constituted by the remaining part. This validation scheme differs from the usual 5-fold cross-validation because the first part of the partition is included into every training sets.

External test set

It is now widely accepted that an external test is required to assess the predictivity of a classification model (Martin et al. 2012; Tropsha and Golbraikh 2007). The constitution of a rigorous external test set, with no involvement in the model development, has been considered as a whole part of this study. We collected every molecule from LeadScope (Leadscope Inc, Dublin, OH) which is annotated with Ames mutagenicity data and which does not belong to the Hansen Dataset. We curated the LeadScope chemical structures in the same way as for the Hansen dataset and we omitted molecules when inconsistent mutagenicity data were observed. It results an external test set of 1 178 molecules to measure the classification accuracy of our rules on unseen data.

4.3.2 Quantitative assessments of the stable emerging molecular patterns

Throughout this section, key quantitative experimental facts are provided and discussed; they result from empirical investigations conducted on the dataset described in the previous section. The whole experiment has been performed thanks to the 5-fold cross-validation scheme introduced by Hansen et al. (2009). Every indicated result corresponds to an average calculated over the 5 folds, unless explicitly stipulated otherwise. In order to obtain potential structural alerts, we rely on the stability measure to select candidates among the frequent emerging molecular

Table 4.1: Best value of the stability thresholds and its related AUC.

Fold	*1*	*2*	*3*	*4*	*5*	*average*
stability threshold	0.92	0.99	0.93	0.98	0.96	**0.96**
AUC	0.7792	0.7697	0.7739	0.7811	0.7789	**0.7766**

Table 4.2: The reduction rate of stability constraint for frequent emerging molecular patterns.

Frequency threshold	*0.36%*	*1%*	*2%*	*5%*	*10%*
# of patterns	222 651.0	38 889.6	8 083.6	868.0	194.8
$\rho \geqslant 2$	12 968.6	2 217.4	534.8	75.8	41.4
$\rho \geqslant 5$	4 564.2	690.2	122.4	4.2	1.2
$\rho \geqslant 10$	1 499.8	189.0	22.8	0.0	0.0

(a) Numbers of emerging patterns.

Frequency threshold	*0.36%*	*1%*	*2%*	*5%*	*10%*
# of SEPs	14 943.0	9 641.8	4 387.8	792.4	183.2
$\rho \geqslant 2$	2 167.2	1 036.4	372.6	62.2	30.8
$\rho \geqslant 5$	616.8	261.0	71.8	3.8	0.8
$\rho \geqslant 10$	164.0	57.0	10.2	0.0	0.0

(b) Numbers of stable emerging patterns (SEPs).

patterns. This selection aims at providing a reasonable number of molecular patterns that are as independent as possible of the constitution of the training set.

The setting of the minimum stability threshold. The selection based on the stability measure relies on a minimum stability threshold. To automatically set this parameter to its best value, we performed a cross-validation on each of the cross-validation fold of the Hansen's dataset (described in the Additionnal Materials). To evaluate the impact of a value of the stability threshold, we measured this impact on the Area Under the Curve (AUC) of a ROC plot. A ROC plot is conceptually similar to an enrichment plot in that it shows the relationship between the true positive rate and the false positive rate (Hanley and McNeil 1982). The Area Under the Curve (AUC) of a ROC plot is common way to quantitavely summarize the overall quality of a ROC plot. By means of the AUC indicator, we aim at discarding as many frequent emerging molecular patterns as possible while conserving the ability to discriminate between the mutagens and non-mutagens.

Table 4.1 reports for each of the original Hansen's fold the best values for stability and its related AUC. It also reports the average on all folds of these thresholds; the mean value will be used in the following. To maximize the AUC stability has to be set to high values (over 0.90) on each of the cross-validation folds. On average, the value of the stability threshold is 0.96.

Stable emerging patterns. Table 4.2 compares the numbers of frequent emerging patterns and stable emerging patterns w.r.t. to different frequency thresholds and different growth-rate thresholds. A comparison of the results provided by 4.2a and by 4.2b indicates that the stability based selection is very efficient. When the frequency threshold is set to 0.36%, the number of patterns is reduced by a factor of 15. At the same time, the selection resulting from a stability threshold tends to keep patterns with a high value of growth-rate. For example, the frequent patterns having a growth-rate above 5 are only reduced by a factor 7.5. The selection based on

Table 4.3: Cover rates obtained with emerging and stable emerging patterns on a test set.

Growth-rate threshold	*Mutagen cover rate*	*Non mutagen cover rate*
0	0.9986	0.9936
1	0.9773	0.8863
2	0.9202	0.6369
3	0.8582	0.4841
4	0.7888	0.3827
5	0.7404	0.3196
6	0.6863	0.2604
7	0.6204	0.2195
8	0.5757	0.1943
9	0.5277	0.1724
10	0.4928	0.1486

(a) Emerging patterns

Growth-rate threshold	*Mutagen cover rate*	*Non mutagen cover rate*
0	0.9986	0.9936
1	0.9728	0.8703
2	0.9031	0.5909
3	0.8332	0.4187
4	0.7610	0.3196
5	0.6951	0.2467
6	0.6309	0.1957
7	0.5455	0.1514
8	0.4973	0.1325
9	0.4470	0.1096
10	0.4226	0.0916

(b) Stable emerging patterns

Table 4.4: Result table of the classification of Hansen's dataset with several classifiers.

Classifier	# of patterns	*Accuracy%*	*Precision%*	*Recall%*	AUC
Emerging Pruned Closed Patterns	222 651	71.73	73.39	80.85	0.777
SEPs	14 943	72.82	76.04	77.92	0.785

stability raises the portion of the strongly dicriminative molecular patterns among all frequent patterns.

The frequency threshold is now set at 0.36%. Table 4.3 provides the cover rates obtained with emerging patterns and with SEPs on a test set. For example, for SEPs with a growth-rate threshold set to 4, 76.10% of the mutagens of a test set and 31.96% of the non-mutagens contain at least one SEP. The comparison of SEPs and emerging patterns shows that the cover rate of mutagens slightly decreases when SEPs are used (the ratio varies between 1 and 85%), while the cover rate of non-mutagens decreases more significantly (the ratio varies from 1 to 60%). It follows that the set of SEPs better separate the mutagens and the non-mutagens, than do the whole set of the emerging pruned closed patterns, when the growth-rate threshold is set to a high value.

As a conclusion, the selection of the stable emerging patterns (SEPs) leads to a set of patterns which is more discriminative. Moreover, as such a selection noticeably decreases the number of patterns, it enables to focus on the strongest chemical patterns and, thus, it facilitates the examination of the selected patterns as potential structural alerts.

Contribution of stability. As seen previously, stability greatly reduces the number of molecular patterns without jeopardizing the cover rate of molecules. To assess the contribution of stability in term of discriminating power, molecular patterns and stable molecular patterns need to be compared in classification.

Molecular patterns can be used as association rules, where the premises are the presence of a pattern in a molecule and the conclusion is mutagenicity of a molecule with a confidence immediately correlated to the growth-rate of the pattern in the premise. Given a growth-rate threshold and a set of association rules, a naive classifier can be engineered to separate the molecules. Using a five-fold cross-validation, the growth-rate threshold is set to the value maximizing the accuracy (this value ranging from 3.39 to 4.05).

Table 4.4 reports the results in terms of accuracy, precision, recall, and AUC. The accuracy is a good prediction rate of a classifier, the precision is the number of mutagens among the predicted mutagen molecules (TP/TP+FP), and the recall corresponds to the number of mutagens predicated among the whole set of mutagens (TP/TP+FN). The first line reports results obtained with emerging pruned closed patterns in a naive classifier, and the second line with stable emerging patterns (SEPs).

The use of SEPs rather than emerging pruned closed patterns increases the accuracy of the naive classifier by more than one point. It also increases the precision by about three percents but at the cost of three percents of recall. Nevertheless, these results on the naive classifier show that using SEPs improves the overall quality of the classifier. It can be explained by the fact that non stable patterns do not generalize very well. Non stable patterns are sensible to their extension, removing few molecules may exclude them from the pattern set. This behavior can be related to labelling errors or statistical anomalies in the training set.

Seal et al. (2012) published classification results using four genuine classifiers on Hansen's dataset. Among these classifiers, there are a naïve bayes which achieves an accuracy of 63.28%, a sequential minimal optimizer achieving an accuracy of 66.43%, J48 which is a decision tree based classifier reaching an accuracy of 73.65%, and finally a random forest which reaches a high accuracy of 79.18%.

We can see that our results are competitive with the ones published by Seal et al. (2012). Indeed, they outperform the naïve bayes and the sequential minimal optimizer, they are similar to the results of J48, but they are behind the results of the random forest. However, it is important to note that results of Seal et al. (2012) does not use the same cross-validation. They use a five-fold cross-validation on all molecules, thus molecules from the static training set can be used in a test set. These molecules are easier to classify ((Hansen et al. 2009)), including them in the test set may boost the accuracy. Using the same type of cross-validation increases the accuracy of our approach to 74.57%, and ranks the SEPs as the second best classifier in terms of accuracy.

If we compare our results in terms of AUC, it is possible to complete our comparaison with the state of the art. Hansen et al. (2009) and Xu et al. (2012) used genuine classifiers (from k-NN to SVM) to separate mutagens from non-mutagens. They respectively reported as best results an AUC of 0.86 and 0.858. These results are better than the results returned by our naïve classifier. But nonetheless, our results in terms AUC indicate that the use of SEPs as a fingerprint in more sophisticated classification techniques is promising.

4.3.3 Expert analysis of the stable emerging patterns (SEPs)

The previous section practically indicates that the successive application of a constraint of frequency, a constraint of emergence and a constraint of stability leads to the automatic identification of promising molecular patterns. Nevertheless, in a process of chemical knowledge discovery, the emerging molecular patterns need a further manual examination by experts of the domain. The examination may produce definitions of new validated structural alerts, but it may also lead to a better understanding of the related activity (e.g., of the mutagenicity).

We focus on the stable emerging patterns denoted as SEPs. As an additional measure, we can evaluate the following ratio. Let p be a SEP and f the set of closed frequent fragments included in p:

$$\mathcal{H}(p) = \frac{\rho(p)}{argmax_{f_i \in f}\rho(f_i)} \tag{4.1}$$

When $\mathcal{H}$ is higher than 1, then the conjunction of fragments is more mutagenic than each of the individual fragments. Two hypotheses can explain this phenomenon. The first one is a conjunction of individually non-mutagenic fragments whose association leads to a mutagenic

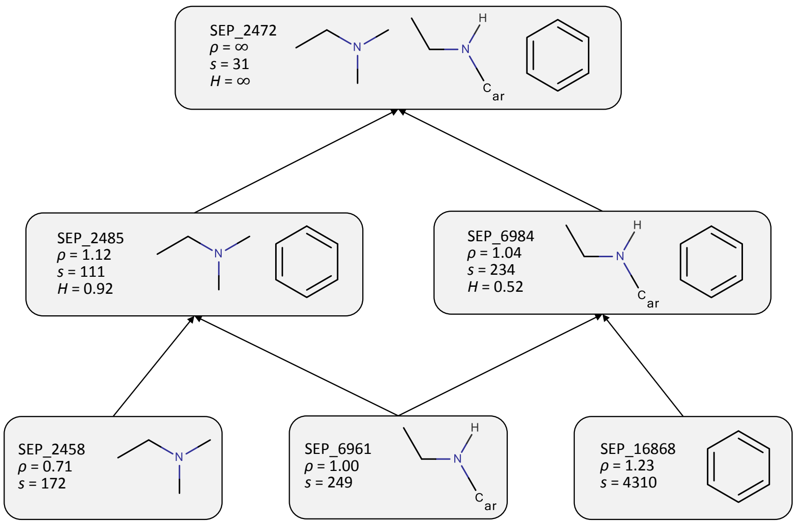


Figure 4.2: Example of a conjunction of SEPs leading to a JEP.

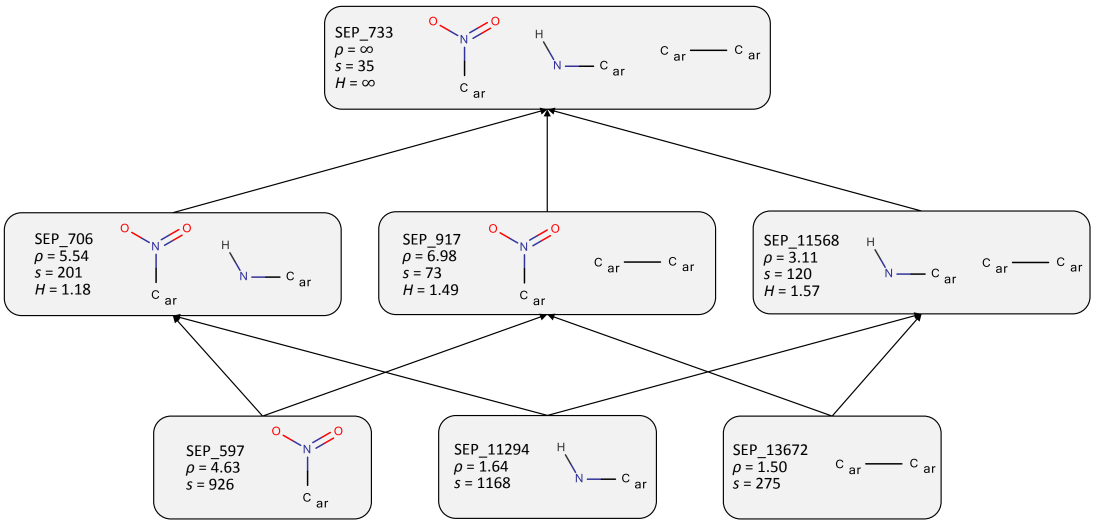
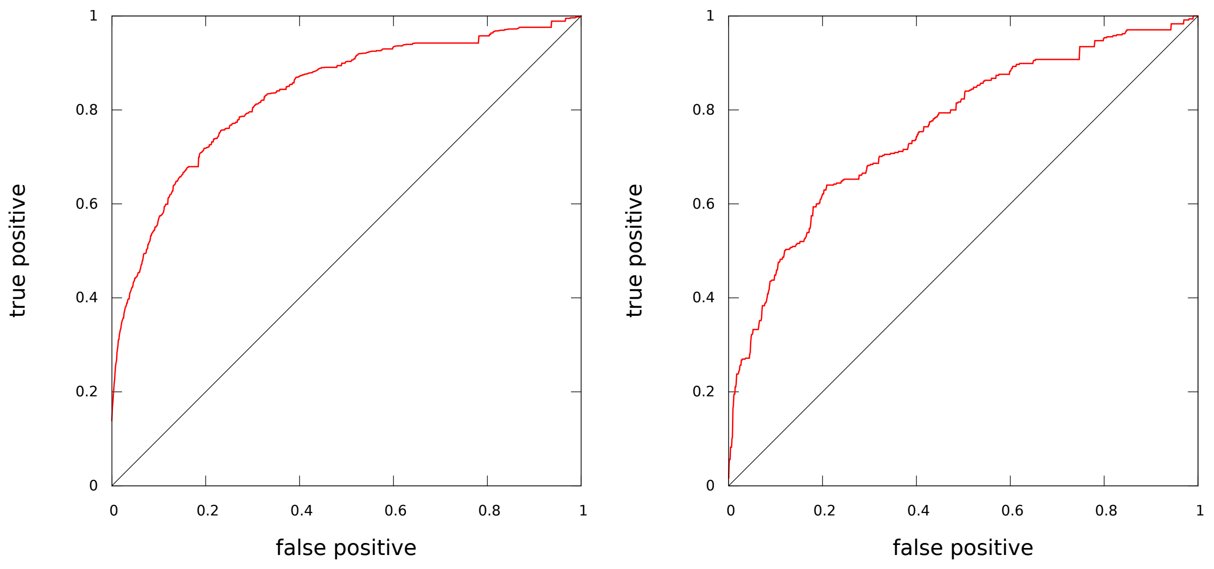


Figure 4.3: Example for the stimulation of a nitro aromatic group.



(a) ROC for training set (b) ROC for external test set

Figure 4.4: ROC plots for the training set (left) and the external validation set (right).

pattern. As an example (see Figure 4.2), let us consider the conjunction of a tertiary amine (SEP_2458), an anilino fragment (SEP_6961) and a phenyl group (SEP_16868). The associations between a tertiary amine and the phenyl group (SEP_2485) or the anilino fragment with the phenyl group (SEP_6984) do not lead to high growth-rates (1.12 and 1.04, resp.) but the three fragments together leads to a JEP (SEP_2472, $\rho = \infty$, $s = 31$, $\mathcal{H} = \infty$).

The second hypothesis is a stimulation (Bissell-Siders et al. 2010) associated to some fragments, leading to an increase of the overall mutagenic property. For example in Figure 4.3 the single nitro aromatic group already represents a contrasting molecular fragment in favor of mutagenicity (SEP_597, $\rho = 4.63$). The conjunction with a nitrogen (NH) connected to an aromatic group (SEP_11294) increases the growth-rate (SEP_706, $\rho = 5.54$). The addition of the third fragment corresponding to two aromatic rings connected by a single bond (SEP_13672) even leads to a JEP (SEP_733, $\rho = \infty$, $s = 35$, $\mathcal{H} = \infty$). For most of the cases, this notion of stimulation is clearly pointed out.

4.3.4 External test of the SEPs

An independent external test set was used to evaluate the generalization of the predicting rules in application. 1178 additional molecules were selected from LeadScope and the performances are shown in Figure 4.4. To be classified as a mutagen, a compound must exhibit at least one SEP. The area under the ROC plot and the maximum prediction rate were approximately 0.76 and 0.73, respectively. We observed a slight decrease of the performances in comparison with the training set (0.83 and 0.76, resp.). The maximum prediction rate was obtained with the molecular patterns displaying a growth-rate greater than 4. By using SEPs with lower growth-rate values we would detect a greater number of mutagens, but the number of false positives, i.e the non-mutagens classified as mutagens, would also increase. The implementation of the rules in more sophisticated classifiers, like k-nearest neighbors algorithm, will improve the performances as suggested by preliminary studies.

4.4 Conclusion

Stable emerging patterns (SEPs) were applied for discovering new relationships between molecular structural features and the toxicological behaviour of a molecule. The computation of these patterns from a molecular dataset has been full-filled by means of a sophisticated workflow that integrates a graph-mining tool together with a well-established measurement of the contrast between classes and with the stability of a pattern.

The methodology was practically applied to a well-known benchmark dataset in order to study mutagenicity. The extracted SEPs were assessed through both a quantitative examination and a chemical expertise. It results that these patterns generalize very efficiently: their quality is preserved from the training set to the test set. Moreover, the SEPs have covered a large scope of different relationships between a molecular structure and its mutagenicity. It follows that the SEPs, when they are used alone as association rules, reach a fair level of confidence on a test set. The chemical analysis has shown that SEPs demonstrate a high ability to express structural alerts.

To wrap up the first part of this manuscript, formal concept analysis allows one to process datasets of complex data. If a dataset is large, stability enables selecting important patterns among a huge set of possibilities. It selects important patterns as it is experimentally shown in this chapter. However, such kinds of approaches requires that a huge set of patterns is found before filtering it with stability. *How can we deal with this huge set of patterns?*. One way is to limit the set of mined pattern by a certain constraint, e.g., in this chapter we filtered out the molecular fragments that contain unclosed chemical ring. In the next part we consider pattern structures and projections (Ganter and Kuznetsov 2001) that enable removing some irrelevant pattern from the search procedure, i.e., they are not computed, and hence the computational time may significantly reduce. Another way is to directly mine stable patterns, i.e., without mining the whole set of patterns. In this way we proceed in the last part of this manuscript and show how to make the procedure polynomial and memory efficient.

Part II

Pattern Structures for Complex Data

Introduction

In this part of the manuscript we switch to a more comprehensive mathematical framework, called pattern structures (Ganter and Kuznetsov 2001). Pattern structures are an extension of FCA and allow one to deal with different types of descriptions, while FCA itself can be considered as a partial case of pattern structures when descriptions are sets of attributes.

Pattern structures are introduced in Chapter 5. This chapter is purely theoretical and is accompanied by only small artificial examples. In Chapter 5 we also extend the original framework of pattern structures introduced by Ganter and Kuznetsov (2001) and study the properties of our extension in order to process complex data more efficiently.

In the followed chapters of this part we provide several applications of pattern structures. In Chapter 6 we show how pattern structures can help to complete RDF data. Indeed, RDF data contain information from different sources that are incomplete and are not necessary known to one another. Thus, information from different sources can be combined and some missed pieces of information can be added to RDF data. The usage of pattern structures in this chapter is quite basic but allows dealing with heterogeneity of RDF data.

Later in Chapter 7 we apply pattern structures in order to deal with syntactic trees for extracting relations. The data consist of natural language sentences containing information about drugs. Apparently, every sentence could describe that drug A and drug B interact (or not) with each other. We show how pattern structures can be used for dealing with syntactic trees and how they can help to extract these drug-drug interaction form natural language texts.

Finally in Chapter 8 we apply pattern structures for dealing with complex sequences. Such kind of sequences naturally appears when working with hospitalization trajectories. Indeed, patient can be hospitalized in different hospitals during the treatment time forming a sequence of hospitalizations. Then, the information of every hospitalization is heterogeneous (medical procedures, location and type of hospital, *etc.*) and hence a good mathematical framework is needed to formalize this kind of data. Thus, we show that pattern structures are useful also in the case of these complex data.

Chapter 5

Pattern Structures and O-projections

Contents

5.1 Introduction

A significant part of recorded data represents phenomena in a structured way, e.g., a molecule is better represented as a labeled graph than as a set of attributes. Pattern structures are an extension of FCA for dealing with such kind of data (Ganter, Grigoriev, et al. 2004; Ganter and Kuznetsov 2001; Ganter and Wille 1999). Such a pattern structure is defined by a set of objects, a set of descriptions associated with the set of objects, and a similarity operation on descriptions, matching a pair of descriptions to their common part. For instance, the set of objects can contain molecule names, the set of descriptions contains fragments of molecules, and the similarity operation taking two sets of graphs to a set of maximal common subgraphs. The similarity operation is a semilattice operation on the set of descriptions. It allows one to deal with data (objects and their descriptions) in a similar way as one deals with objects and their intents in standard FCA. Such kind of formalization allows one to describe many types of data, however processing can be computationally very demanding. For example, pattern structures on sets of graphs (Ganter, Grigoriev, et al. 2004; Ganter and Kuznetsov 2001; Kuznetsov and Samokhin 2005) is based on the operation of finding maximal common subgraphs for a set of graphs, which is #P-hard.

To deal with this complexity and to have a possibility to process most of the data, projections of pattern structures were introduced (Ganter and Kuznetsov 2001). Projections are special mathematical functions on the set of descriptions that simplify the descriptions of objects. This approach reduces the number of concepts in the pattern lattice corresponding to a pattern structure. However, it does not impact the computational worst-case complexity of the similarity operation. Moreover, it cannot remove concepts of special kinds from the "middle" of the semilattice which can be important in some practical cases, e.g., concepts containing too small graphs can be considered useless but they cannot be removed with projections. For example, in Buzmakov et al. (2013b) concepts having intents that include short sequences of patient hospitalisations have little sense. Hence, short sequences could be "removed" from the intent, but the descriptions of objects, i.e., patients, usually include only one long sequence and should not be changed.

In this chapter we introduce *o-projections* of pattern structures, a generalization of projections of pattern structures, that allow one to reduce the computational complexity of similarity operations. They also allow one to remove certain kinds of descriptions in the "middle" of the semilattice while the descriptions of the objects can be preserved. By introducing o-projections of pattern structures, we correct also a formal problem of projections of pattern structures, which is discussed later.

The main difference between o-projections and projections is that in o-projected pattern structures we modify the semilattice of descriptions, while in the case of projected pattern structures we can modify only the descriptions of single objects. It should be noticed that most of the properties of projections are valid for o-projections. However, the relation between representation contexts, a reduction from pattern structures to FCA, and projections is quite different from the relation between representation contexts and o-projections. In addition we have discovered the fact that the set of o-projections of a pattern structure forms a semilattice. From a practical point of view it allows one to apply a set of independent o-projections, e.g., o-projections obtained from several experts, to a pattern structure.

We should notice that Pernelle et al. (2002) introduce an approach very similar to pattern structures. Moreover, they have also introduced the notion of projections that is free from the limitations of projections by Ganter and Kuznetsov (2001). However, they have not studied the relation between representation contexts and projections. The recent work of Soldano and Ventos (2011) has also discussed the lattice-order of projections. However below we discuss this order that helps us to prove the relation between the representation contexts and o-projections.

This chapter is based on Buzmakov et al. (2015c) and further develops the methodology introduced in Buzmakov et al. (2013b), where it was applied for the analysis of sequential pattern structures by introducing projections that remove irrelevant concepts.

The rest of the chapter is organized as follows. In Section 5.2 we introduce the definitions of a pattern structure, representation context of a pattern structure, and discuss how one can compute with pattern structures along the lines of FCA. Section 5.3 introduces projections and o-projections of a pattern structure, defines the partial order on o-projections and shows that this order is a semilattice. At the end of this section the relation between o-projections and representation contexts of o-projected pattern structure is discussed. Finally, we conclude the chapter and discuss furture work.

5.2 Pattern Structures

In FCA a formal context (G, M, I), where G is a set of objects, M is a set of attributes, and $I \subseteq G \times M$ is a binary relation between G and M, is taken to a concept lattice $\mathfrak{L}(G, M, I)$ (Ganter and Wille 1999). For non-binary data, such as sequences or graphs, lattices can be constructed in the same way using pattern structures (Ganter and Kuznetsov 2001).

Definition 5.1. *A pattern structure $\mathbb{P}$ is a triple $(G, (D, \sqcap), \delta)$, where G, D are sets, called the set of objects and the set of descriptions, and $\delta : G \to D$ maps an object to a description. Respectively, $(D, \sqcap)$ is a meet-semilattice on D w.r.t. $\sqcap$, called similarity operation such that $\delta(G) := \{\delta(g) \mid g \in G\}$ generates a complete subsemilattice $(D_\delta, \sqcap)$ of $(D, \sqcap)$.*

For illustration, let us represent standard FCA in terms of pattern structures. The set of objects G is preserved, the semilattice of descriptions is $(\wp(M), \cap)$, where $\wp(M)$ denotes the powerset of the set of attributes M, a description is a subset of attributes and $\cap$ is the set-theoretic intersection. If $x = \{a, b, c\}$ and $y = \{a, c, d\}$ then $x \sqcap y = x \cap y = \{a, c\}$, and $\delta : G \to \wp(M)$ is given by $\delta(g) = \{m \in M \mid (g, m) \in I\}$.

Note that Definition 5.1 has an important partial case where $(D, \sqcap)$ is a complete meet-semilattice. In this case the semilattice $(D_\delta, \sqcap)$ is necessarily complete. First, in practical applications one often needs finite lattices, which are always complete. Second, in many practical cases one can easily extend an incomplete semilattice to a complete one by introducing some extra elements. For example, given an incomplete semilattice w.r.t containment order on the interval (a, b), one can add a and b to obtain the interval $[a, b]$, which is a complete semilattice. Some of the statements hereafter hold only for the partial case of $(D, \sqcap)$ being a complete meet-semilattice.

The Galois connection for a pattern structure $(G, (D, \sqcap), \delta)$, relating sets of objects and descriptions, is defined as follows:

$$A^\diamond := \bigsqcap_{g \in A} \delta(g), \qquad \text{for } A \subseteq G$$

$$d^\diamond := \{g \in G \mid d \sqsubseteq \delta(g)\}, \qquad \text{for } d \in D$$

Given a subset of objects A, $A^\diamond$ returns the description which is common to all objects in A. Given a description d, $d^\diamond$ is the set of all objects whose description subsumes d. The natural partial order (or subsumption order between descriptions) $\sqsubseteq$ on D is defined w.r.t. the similarity operation $\sqcap$: $c \sqsubseteq d \Leftrightarrow c \sqcap d = c$ (in this case we say that c is subsumed by d). In the case of standard FCA the natural partial order corresponds to the set-theoretical inclusion order, i.e., for two sets of attributes x and y $x \sqsubseteq y \Leftrightarrow x \subseteq y$.

Definition 5.2. *A pattern concept of a pattern structure $(G, (D, \sqcap), \delta)$ is a pair (A, d), where $A \subseteq G$ and $d \in D$ such that $A^\diamond = d$ and $d^\diamond = A$; A is called the pattern extent and d is called the pattern intent.*

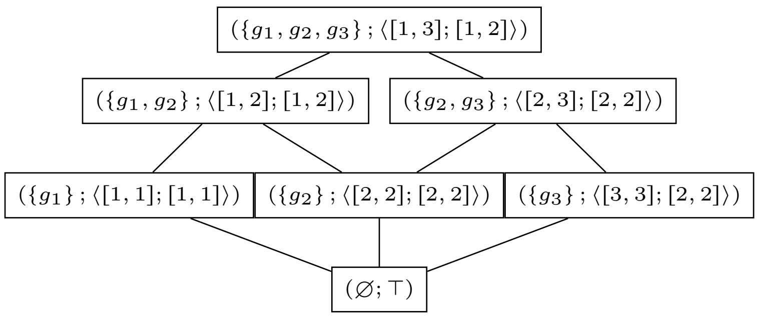
As in standard FCA, a pattern concept corresponds to the maximal set of objects A whose description subsumes the description d, where d is the maximal common description of objects in A. The set of all pattern concepts is partially ordered w.r.t. inclusion of extents or, dually, w.r.t. subsumption of pattern intents within a concept lattice, these two antiisomorphic orders making a lattice, called pattern lattice.

5.2.1 Running Example

Kaytoue, Kuznetsov, Napoli, and Duplessis (2011) have used interval pattern structures for gene expression analysis. Let us consider an example of such pattern structures. In Figure 5.1a an interval context is shown. It has three objects and two attributes. Every attribute shows the interval of values the attribute can have. If we have two objects, then a numerical attribute

	m_1	m_2
g_1	$[1,1]$	$[1,1]$
g_2	$[2,2]$	$[2,2]$
g_3	$[3,3]$	$[2,2]$

(a) An interval context.


$(\{g_1,g_2,g_3\}\,;\langle[1,3];[1,2]\rangle)$
$(\{g_1,g_2\}\,;\langle[1,2];[1,2]\rangle)$
$(\{g_2,g_3\}\,;\langle[2,3];[2,2]\rangle)$
$(\{g_1\}\,;\langle[1,1];[1,1]\rangle)$
$(\{g_2\}\,;\langle[2,2];[2,2]\rangle)$
$(\{g_3\}\,;\langle[3,3];[2,2]\rangle)$
$(\varnothing;\top)$

(b) An interval pattern lattice.

Figure 5.1: An interval pattern structure and the corresponding lattice.

can have all values from the interval of this attribute in the first object and from the interval of this attribute of the second object. Consequently, the similarity between two intervals can be defined as a convex hull of the intervals, i.e. $[a,b] \sqcap [c,d] = [\min(a,c), \max(b,d)]$. Then, given two tuples of intervals, the similarity between these tuples is computed as a component-wise similarity between intervals.

In this example, we have the pattern structure $(G,(D,\sqcap),\delta)$, where $G = \{g_1, g_2, g_3\}$, the set D is the set of all possible interval pairs with the similarity operation described above, and δ is given by the context in Figure 5.1a, i.e., $\delta(g_1) = \langle[1,1];[1,1]\rangle$ and $\delta(g_1) \sqcap \delta(g_2) = \langle[1,2];[1,2]\rangle$.

Figure 5.1b shows the pattern lattice of the interval context in Figure 5.1a. One can check that the extents and the intents in this lattice are connected by means of the Galois connection given above. The partial order in the semilattice of intervals is given by "the smaller the interval, the larger the description with this interval", i.e., the former description gives more certainty about the values than the latter.

5.2.2 Representation Context of a Pattern Structure

Note that any pattern structure can be represented by a formal context with the concept lattice isomorphic to the lattice of the pattern structure. Below we introduce a representation context of a pattern structure and its properties in the line of Ganter and Kuznetsov (2001).

Given a pattern structure $(G,(D,\sqcap),\delta)$, we denote by $D_\delta \subseteq D$ the set of all intents of the concept lattice, i.e., $D_\delta = \left\{ d \in D \mid (\exists X \subseteq G) \bigsqcap_{g \in X} \delta(g) = d \right\}$. Since $(D_\delta, \sqcap)$ is a complete subsemilattice of $(D,\sqcap)$, for $X \subseteq D$ a join operation $\sqcup$ can be defined as follows:

$$\bigsqcup X = \bigsqcap \{d \in D_\delta \mid (\forall x \in X) x \sqsubseteq d\}\,.$$

Given this join operation, $(D_\delta, \sqcap, \sqcup)$ is a complete lattice. We say that a set $M \subseteq D$ is $\sqcup$-dense for $(D_\delta, \sqcap)$ if every element in D_δ is of the form $\sqcup X$ for some $X \subseteq M$. For example, $M = D_\delta$ is always $\sqcup$-dense for D_δ.

Definition 5.3. *Given a pattern structure $\mathbb{P} = (G,(D,\sqcap),\delta)$ and a set $M \subseteq D$ $\sqcup$-dense in D_δ, a formal context (G,M,I) is called the representation context of $\mathbb{P}$, if I is given by $I = \{(g,m) \in G \times M \mid m \sqsubseteq \delta(g)\}$. The representation context of $\mathbb{P}$ is denoted by $\mathbb{R}(\mathbb{P})$.*

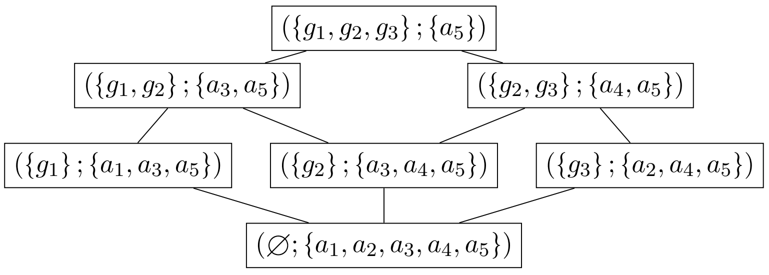
The next theorem establishes a bijection between the pattern concepts in the lattice of pattern structure $\mathbb{P}$ and the concepts in the lattice of the representation context $\mathbb{R}(\mathbb{P})$. Here, the ideal of element $d \in D$ is denoted by $\downarrow d = \{e \in D \mid e \sqsubseteq d\}$.

	$\langle[3,+\infty];[-\infty,+\infty]\rangle$	$\langle[2,+\infty];[-\infty,+\infty]\rangle$	$\langle[-\infty,1];[-\infty,+\infty]\rangle$	$\langle[-\infty,2];[-\infty,+\infty]\rangle$	$\langle[-\infty,+\infty];[2,\infty]\rangle$	$\langle[-\infty,+\infty];[-\infty,1]\rangle$	$\langle[1,3];[1,2]\rangle$
	$m_1 \geqslant 3$	$m_1 \geqslant 2$	$m_1 \leqslant 1$	$m_1 \leqslant 2$	$m_2 \geqslant 2$	$m_2 \leqslant 1$	
g_1			x	x		x	x
g_2		x		x	x		x
g_3	x	x			x		x

(a) Representation context corresponding to interordinal scaling.

	$\langle[1,1];[1,1]\rangle$	$\langle[3,3];[2,2]\rangle$	$\langle[1,2];[1,2]\rangle$	$\langle[2,3];[2,2]\rangle$	$\langle[1,3];[1,2]\rangle$
	a_1	a_2	a_3	a_4	a_5
g_1	x		x		x
g_2			x	x	x
g_3		x		x	x

(b) Another possible representation context.


$(\{g_1,g_2,g_3\};\{a_5\})$
$(\{g_1,g_2\};\{a_3,a_5\})$
$(\{g_2,g_3\};\{a_4,a_5\})$
$(\{g_1\};\{a_1,a_3,a_5\})$
$(\{g_2\};\{a_3,a_4,a_5\})$
$(\{g_3\};\{a_2,a_4,a_5\})$
$(\varnothing;\{a_1,a_2,a_3,a_4,a_5\})$

(c) A concept lattice for the context if Figure 5.2b.

Figure 5.2: Possible representation contexts for the pattern structure in Figure 5.1 and the concept lattice for the context in Figure 5.2b.

Theorem 5.1 (Theorem 1 from Ganter and Kuznetsov (2001)). *Let $\mathbb{P} = (G, (D, \sqcap), \delta)$ be a pattern structure and let $\mathbb{R}(\mathbb{P}) = (G, M, I)$ be a representation context of $\mathbb{P}$. Then for any $A \subseteq G$, $B \subseteq M$, and $d \in D$ the following conditions are equivalent:*

1. *(A, d) is a pattern concept of $\mathbb{P}$ and $B = \downarrow d \cap M$.*

2. *(A, B) is a formal concept of $\mathbb{R}(\mathbb{P})$ and $d = \bigsqcup B$.*

Example 5.1. *A representation context for the pattern structure given in Figure 5.1 can be given by the set M where every element $m \in M$ is of the form $\langle[-\infty, a]; [-\infty, +\infty]\rangle$ or $\langle[-\infty, +\infty]; [b, +\infty]\rangle$, and $a, b \in \{1, 2, 3\}$.*

In fact, the element $\langle[-\infty, +\infty]; [a, +\infty]\rangle$ corresponds to the attribute '$m_2 \geqslant a$' in the case of the interordinal scaling (Ganter and Wille 1999) of numerical data. Another representation context can be constructed from the intents of join-irreducible concepts of the lattice in Figure 5.1b. These two representation contexts of the pattern structure related to Figure 5.1 are shown in Figures 5.2a and 5.2b. It can be seen that the resulting lattices, e.g., the lattice in Figure 5.2c, are isomorphic to the lattice in Figure 5.1b.

It should be noticed that in some cases the representation context is hard to compute. For example, in case of numerical data with the set of all values W, to construct representation context, one needs to create $2|W| + 1$ binary attributes, which can be much more than the number of original real-valued attributes. Kaytoue, Kuznetsov, Napoli, and Duplessis (2011) have shown that pattern structures provide more efficient computations than the equivalent

```
Function CloseByOne($Ext$, $Int$)
    Data: $\mathbb{P} = (G, (D, \sqcap), \delta)$, the extent $Ext$ and the intent $Int$ of a concept.
    Result: All canonical ancestors of $(Ext, Int)$ in the concept lattice.
    foreach $S \subseteq G$, $S \succ Ext$ do
        $NewInt \longleftarrow \bigsqcap_{g \in S} \delta(g)$ ;                    /* $\sqcap$ - the similarity */
        $NewExt \longleftarrow \{g \in G \mid NewInt \sqsubseteq \delta(g)\}$ ;   /* $\sqsubseteq$ - the subsumption */
        if IsCanonicExtension($Ext$, $NewExt$) then
            SaveConcept(($NewExt, NewInt$));
            CloseByOne($NewExt$,$NewInt$);
/* Looking for all concepts of the concept lattice                     */
CloseByOne($\varnothing$, $\top$);
```

Algorithm 3: The version of the Close-by-One algorithm computing the pattern lattice of a pattern structure $\mathbb{P}$.

approach based on FCA and scaling, which can be considered as a way to build representation context of interval pattern structures, e.g., see Figure 5.2a.

In case of graph data the set of attributes of the representation context consists of all subgraphs of the original graph descriptions, which is hard to compute (Kuznetsov and Samokhin 2005).

5.2.3 Computation of Pattern Lattices

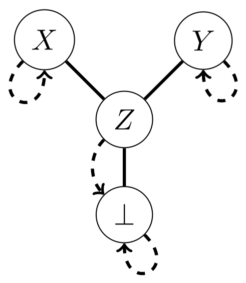
Nearly any algorithm for computing concept lattices from contexts can be adapted to compute pattern lattices from pattern structures. To adapt an algorithm, every set intersection operation on attributes is replaced by the semilattice operation $\sqcap$ on corresponding patterns, and every subset checking is replaced by the semilattice order $\sqsubseteq$ checking, in particular, all $(\cdot)'$ operations are replaced by $(\cdot)^{\diamond}$. For example, let us consider a modified version of Close-by-One (`CbO`) algorithm (Kuznetsov 1993).

Algorithm 3 shows the listing of the modified part of `CbO`. Here the canonical extension `IsCanonicExtension` and canonical order $\succ$ are defined on the set of objects and hence are the same as in Kuznetsov (1993). We can see that only lines 3 and 4 are modified. In these lines the set intersection operation and the subset relation checking are replaced by the corresponding operators of a pattern structure.

5.3 Revised Projections of Pattern Structures

Pattern structures are hard to process due to the large number of pattern concepts in the pattern lattice and the algorithmic complexity of the similarity operation $\sqcap$. Projections of pattern structures “simplify” to some degree the computation and allow one to work with “simpler" descriptions. In fact, a projection can be considered as a mapping for pruning descriptions with certain mathematical properties. These properties ensure that a projection of a semilattice is a semilattice and that the concepts of a projected[9] pattern structure are related to the concepts of the original pattern structure (Ganter and Kuznetsov 2001).

[9] We use the expression “a projected pattern structure” instead of “a projection of a pattern structure” to distinguish between projection as an operator ψ and as the result of applying the operator to a pattern structure.



$$D = \{x, y, z, \bot\}$$

$$\psi : x \mapsto x, y \mapsto y,$$
$$z \mapsto \bot, \bot \mapsto \bot$$

$$\psi(x \sqcap y) = \psi(z) = \bot \neq$$
$$\neq z = \psi(x) \sqcap \psi(y)$$

Figure 5.3: Contrexample to Proposition 1 from (Ganter and Kuznetsov 2001).

In this section we introduce o-projected pattern structures (“o" coming from “order"), i.e. a revision of projected pattern structures in accordance with (Buzmakov et al. 2013b). We discuss the properties of o-projected pattern structures and relate them to the projected pattern structures from (Ganter and Kuznetsov 2001). The notion of (o-)projected pattern structure is based on a kernel operator (a projection).

Definition 5.4 ((Ganter and Kuznetsov 2001))**.** *A projection $\psi : D \to D$ is a kernel (interior) operator on the partial order $(D, \sqsubseteq)$, i.e. it is (1) monotone ($x \sqsubseteq y \Rightarrow \psi(x) \sqsubseteq \psi(y)$), (2) contractive ($\psi(x) \sqsubseteq x$) and (3) idempotent ($\psi(\psi(x)) = \psi(x)$).*

Given a projection ψ we say that the fixed point of ψ is the set of all elements from D such that they are mapped to themselves by ψ. The fixed point of ψ is denoted by $\psi(D) = \{d \in D \mid \psi(d) = d\}$. Note that, if $\psi(d) \neq d$, then there is no other $\tilde{d}$ such that $\psi(\tilde{d}) = d$ because of idempotency. Hence, any element outside the fixed point of the projection ψ is pruned.

5.3.1 Definition of Projected Pattern Structures

Let us first consider the projected pattern structure w.r.t. a projection ψ according to Ganter and Kuznetsov (2001). Given a pattern structure $\mathbb{P} = (G, (D, \sqcap), \delta)$ and a projection ψ on D, the projected pattern structure is defined as $(G, (D, \sqcap), \psi \circ \delta)$. As we can see, a projection only changes the descriptions of the objects but not the underlying semilattice $(D, \sqcap)$. There are two problems with this definition of the projected pattern structures. First, it is necessary to restrict the class of projections given by Definition 5.4 in order to ensure the property $\psi(x \sqcap y) = \psi(x) \sqcap \psi(y)$. Second, the complexity of computing $\sqcap$ can be very high, but with this kind of projected patten structures we cannot decrease the algorithmic complexity. Below we discuss these two points.

In Proposition 1 of (Ganter and Kuznetsov 2001) the following property of the projection operator is discussed: given a semilattice $(D, \sqcap)$ and a projection ψ on D, for any two elements x and y from D one has $\psi(x \sqcap y) = \psi(x) \sqcap \psi(y)$. Let us consider the example in Figure 5.3 with the meet-semilattice $D = \{x, y, z, \bot\}$ given by its diagram and the projection ψ given by the dotted lines. It is easy to see that $\psi(x \sqcap y) = \bot \neq z = \psi(x) \sqcap \psi(y)$. One way of solving this problem is to give additional conditions on projection ψ that would imply the required property. An important example is the following condition: for all $x, y \in D$ if $x < y$ and $\psi(y) = y$, then $\psi(x) = x$. This kind of solution respects the intuition behind the definition of the projected pattern structure in Ganter and Kuznetsov (2001), according to which the initial descriptions of objects are changed, but the similarity operation $\sqcap$ is not changed.

Another way of solving the problem above is to generalize the definition of the projected pattern structure, and we proceed in this way in the next section, by allowing to modify the similarity operation on descriptions.

5.3.2 Definition of o-projected Pattern Structures

Below we propose a definition of o-projected pattern structures by means of a kernel operator ψ. The definition takes into account the problems discussed above. In the o-projected pattern structure we substitute the semilattice of descriptions by its suborder (the letter "o" comes from "order") with another similarity operation, which can be different from the initial one.

Let us first note that, given a meet-semilattice D and a kernel operator ψ, the fixed point $\psi(D)$ is a semilattice w.r.t. the natural order on D.

Theorem 5.2. *Given a semilattice $(D, \sqcap)$ and a kernel operator ψ, the fixed point $(\psi(D), \sqcap_\psi)$ is a semilattice w.r.t. the natural order on $(D, \sqcap)$, i.e., $d_1 \sqsubseteq d_2 \Leftrightarrow d_1 \sqcap d_2 = d_1$. If $\bigsqcap X$ exists for a set $X \subseteq D$, then $\bigsqcap_{\psi\, x \in X} \psi(x)$ exists and is given by*

$$\bigsqcap_{\psi\, x\in X} \psi(x) = \psi(\bigsqcap_{x\in X} x) \tag{5.1}$$

Proof. Let us denote $d = \bigsqcap_{x\in X} x$. Since $(\forall x \in X) d \sqsubseteq x$, one has $(\forall x \in X)\psi(d) \sqsubseteq \psi(x)$. Let us show that for any $p \in \psi(D)$, i.e. $\psi(p) = p$ such that $(\forall x \in X)p \sqsubseteq \psi(x)$, we have $p \sqsubseteq \psi(d)$, i.e., that $\psi(d) = \bigsqcap_{\psi\, x\in X} \psi(x)$.

Since $(\forall x \in X)p \sqsubseteq \psi(x)$ then $(\forall x \in X)p \sqsubseteq x$. Since $d = \bigsqcap_{x\in X} x$, one has $p \sqsubseteq d$. Thus, $p = \psi(p) \sqsubseteq \psi(d)$ and $\psi(d)$ is the minimum of the set $\psi(X)$, i.e. $\psi(D)$ is a semilattice and the Eq. (5.1) holds. □

Corollary 5.1. *Given a complete subsemilattice $\tilde{D}$ of $(D, \sqcap)$ and a kernel operator ψ on D, the image of $\tilde{D}$ is a complete subsemilattice $\psi(\tilde{D})$ of the fixed point $(\psi(D), \sqcap_\psi)$.*

Since according to Theorem 5.2 $\psi(D)$ is a semilattice and according to Corollary 5.1 $\psi(D_\delta)$ is a complete semilattice, we can define an o-projected pattern structure as a pattern structure with $\psi(D)$ as a semilattice.

Definition 5.5. *Given a pattern structure $\mathbb{P} = (G, (D, \sqcap), \delta)$ and a kernel operator ψ on D, the o-projected pattern structure $\psi(\mathbb{P})$ is a pattern structure $(G, (\psi(D), \sqcap_\psi), \psi \circ \delta)$, where $\psi(D) = \{d \in D \mid \psi(d) = d\}$ and $\forall x, y \in D, x \sqcap_\psi y := \psi(x \sqcap y)$.*

In the o-projected pattern structure the kernel operator ψ modifies not only the descriptions of the objects, but also the semilattice operation, i.e., the semilattice $(\psi(D), \sqcap_\psi)$ is not necessarily a subsemilattice of $(D, \sqcap)$ and so it is not always true that $x \sqcap y = x \sqcap_\psi y$ in D.

Example 5.2. *Let us define an o-projection for the interval pattern structure from Subsection 5.2.1. Let us suppose that the aggregated size of a pattern, i.e., the sum of the lengths of the intervals in the pattern, should be less than 2. First, we should define the corresponding kernel operator $\psi : D \to D$. Thus, if an aggregated length of a pattern p is less than 2, then $\psi(p) := p$, otherwise $\psi(p) := \bot = \langle[-\infty, +\infty]; [-\infty, +\infty]\rangle$. For instance, $\psi(\langle[1, 1]; [1, 1]\rangle) = \langle[1, 1]; [1, 1]\rangle$, while $\psi(\langle[1, 2]; [1, 2]\rangle) = \langle[-\infty, +\infty]; [-\infty, +\infty]\rangle$, because it has two intervals of length 1, i.e., the aggregated size is equal to 2.*

Let us consider the o-projected interval pattern structure $(G, (\psi(D), \sqcap_\psi), \psi \circ \delta)$. It is clear that $\psi \circ \delta = \delta$, thus this o-projected interval pattern structure cannot be expressed as a projected pattern structure.

The concepts of a pattern structure and a projected pattern structure are connected through Proposition 5.1. This proposition can be found in Ganter and Kuznetsov (2001), but thanks to Theorem 5.2, it is also valid in our case.

Proposition 5.1. *Given a pattern structure $\mathbb{P} = (G, (D, \sqcap), \delta)$ and a kernel operator ψ on D:*

1. *if A is an extent in $\psi(\mathbb{P})$, then A is also an extent in $\mathbb{P}$.*
2. *if d is an intent in $\mathbb{P}$, then $\psi(d)$ is also an intent in $\psi(\mathbb{P})$.*

It is easy to see that the other propositions from Ganter and Kuznetsov (2001) concerning projected pattern structures hold for the o-projected pattern structures as well. Below we cite Proposition 3 from Ganter and Kuznetsov (2001) that relates implications in a pattern structure and those in an o-projected pattern structure. We skip the propositions related to supervised classification with projected pattern structures by means of hypotheses, since it is out of the scope of the current work. However, they are valid in the case of o-projected pattern structures and can be proven with the help of Theorem 5.2.

Proposition 5.2 (Proposition 3 from Ganter and Kuznetsov (2001))**.** *Let $a, b \in D$. If $\psi(a) \to \psi(b)$ and $\psi(b) = b$ then $a \to b$, where $x \to y \Leftrightarrow$ for all $g \in G$ ($x \sqsubseteq \delta(g)$ implies $y \sqsubseteq \delta(g)$)*

5.3.3 Order of Projections

In this subsection we limit ourselves to the practically important case when a set of descriptions is a complete semilattice. We can consider projections as a means of description pruning in $(D, \sqcap)$. Indeed, given a complete semilattice $(D, \sqcap)$ and a projection ψ on this semilattice, the set D can be divided into two sets $D = \{d \in D \mid \psi(d) = d\} \cup \{d \in D \mid \psi(d) \neq d\}$, i.e., the fixed point of ψ and the rest. It can be seen that the intents of the o-projected pattern structure $\psi((G, (D, \sqcap), \delta))$ are in the fixed point of ψ, i.e., all elements of the form $\psi(d) \neq d$ are discarded. We recall that by $\psi(D) = \{d \in D \mid \psi(d) = d)\}$ we denote the fixed point of ψ. *But under which condition do we have that for any $D_1 \subset D_2$ there is a projection ψ of D_2 such that $\psi(D_2) = D_1$?* The following theorem gives necessary and sufficient conditions for such a property.

Theorem 5.3. *Given a complete semilattice $(D, \wedge)$, with the natural order $\leqslant$, and $D_s \subseteq D$, there is a projection $\psi : D \to D$ such that $\psi(D) = D_s$, if and only if $\bot \in D_s$ and for any $X \subseteq D_s \subseteq D$, one has $\bigvee X \in D_s$, where $\bot := \bigwedge D$ and $\bigvee X = \bigwedge \{d \in D \mid (\forall x \in X) d \geqslant x\}$.*

Proof. 1. Given a projection ψ such that $\psi(D) = D_s$, $\bot \in D_s$ because of contractivity of ψ, i.e., $\psi(\bot) = \bot$. Let us suppose that there is a set $X \subseteq D_s$, i.e., $(\forall x \in X)\psi(x) = x$ such that $\psi(\bigvee X) \neq \bigvee X$. Then, $(\forall x \in X)(x < \bigvee X \underset{\text{monotonicity}}{\Rightarrow} x \leqslant \psi(\bigvee X) \underset{\text{contractivity}}{<} \bigvee X)$. It is a contradiction, since $\bigvee X$ is the supremum of X. Hence for any $X \subseteq D_s$ we have $\psi(\bigvee X) = \bigvee X$.

2. Given $D_s \subseteq D$ such that $\bot \in D_s$ and for any $X \subseteq D_s$, one has $\bigvee X \in D_s$, let us construct the corresponding projection ψ. First, $\psi(d \in D_s) := d$ and for all $d \in D \backslash D_s$ we should have $\psi(d) \neq d$. For an element $d \in D \backslash D_s$, let us consider the set $S_d = \{x \in D_s \mid x < d\}$, which is

not an empty set since $\bot \in D_s$. We know that $\bigvee S_d \in D_s$ and by definition of $\bigvee$ we have $\bigvee S_d < d$. Then we set $\psi(d) := \bigvee S_d$.

Let us show that the function ψ is a projection of D. Idempotency and contractivity are satisfied by the construction of ψ. Let us check monotonicity. Let us take any $a, b \in D$ such that $a > b$. Then, if $\psi(a) = a$, then $\psi(a) = a > b \geqslant \psi(b)$, i.e., the monotonicity holds. If $\psi(a) \neq a$, then $\psi(a) = \bigvee S_a$ by construction. Hence, if $\psi(b) = b$, then $b \in S_a$, i.e., $\psi(a) \geqslant \psi(b)$. Finally, if $\psi(b) \neq b$, then $S_b \subseteq S_a$, because if $d \in S_b$, i.e., $d < b$, then $d < b < a$, i.e. $d \in S_a$. In this case, $\psi(a) = \bigvee S_a \geqslant \bigvee S_b = \psi(b)$.

□

Corollary 5.2. *Given a complete semilattice $(D, \wedge)$, with the natural order $\leqslant$, and a subset $D_s \subseteq D$ such that $\bot \in D_s$ and for any $X \subseteq D_s$, one has $\bigvee X \in D_s$, the poset $(D_s, \leqslant)$ is a complete semilattice.*

Proof. According to Theorem 5.3 there is a projection $\psi : D \to D$ such that $\psi(D) = D_s$. Then, according to Theorem 5.2 D_s is a semilattice. □

Since a projection of D can be considered as a mapping with the fixed point $\psi(D)$, we can introduce an order w.r.t. this fixed point.

Definition 5.6. *Given a complete semilattice $(D, \sqcap)$ and two projections ψ_1 and ψ_2 in D, we say that $\psi_1 \leqslant \psi_2$ if $\psi_1(D) \subseteq \psi_2(D)$.*

However in some cases, it is more convenient to order projections w.r.t. a superposition of projections or their "generality".

Definition 5.7. *Given a complete semilattice $(D, \sqcap)$ and two projections ψ_1 and ψ_2 in D, we say that $\psi_1 \leqslant \psi_2$ if there is a projection $\psi : \psi_2(D) \to \psi_2(D)$ such that $\psi_1 = \psi \circ \psi_2$.*

It can be seen that these two definitions yield the same ordering.

Proposition 5.3. *Definitions 5.6 and 5.7 are equivalent.*

Proof.
1. Let $\psi_1 = \psi \circ \psi_2$. Since ψ is a projection in $\psi_2(D)$, then $\psi_1(D) = \psi(\psi_2(D)) \subseteq \psi_2(D)$.

2. Let $\psi_1(D) \subseteq \psi_2(D)$. Let us denote by $(\cdot)_1$ and $(\cdot)_2$ the operations in $(\psi_1(D), \sqcap_{\psi_1})$ and $(\psi_2(D), \sqcap_{\psi_2})$, respectively, and let us denote $D_i = \psi_i(D)$ the fixed points of ψ_i, where $i \in \{1, 2\}$.

   Let us build $\psi : D_2 \to D_1$ equal to ψ_1 in D_2, i.e., for all $d \in D_2$ we set $\psi(d) := \psi_1(d)$. Since ψ_1 is a projection in D, ψ is a projection in D_2 (the natural order is the same). Since D_1 is the fixed point of ψ_1 then $\psi_1(D_2) \subseteq D_1$. However, since $D_1 \subseteq D_2$ and $\psi_1(D_1) = D_1$ then $\psi_1(D_2) = D_1$, i.e., there is a projection ψ such that $\psi_1 = \psi \circ \psi_2$.

□

Example 5.3. *Let us return to Example 5.2. We change the threshold for the aggregated size. In Example 5.2 it was set to 2 ($\psi_{al=2}$), but we can change it to 5 ($\psi_{al=5}$) or 10 ($\psi_{al=10}$). The higher the threshold, the more possible descriptions are projected to themselves, i.e., belong to the fixed point of the projection. Thus, we have $\psi_{al=2} \leqslant \psi_{al=5} \leqslant \psi_{al=10}$.*

Thanks to Proposition 5.1 it can be seen that, given a pattern structure $\mathbb{P}$, if we have two projections $\psi_1 \leqslant \psi_2$, then the set of pattern extents of $\psi_1(\mathbb{P})$ is a subset of the set of pattern extents of $\psi_2(\mathbb{P})$, i.e., the smaller the projection, the smaller the number of concepts in the corresponding projected pattern structure.

Now it can be seen that projections actually form a semilattice with respect to the previously defined order.

Proposition 5.4. *Projections of a complete semilattice $(D, \sqcap)$ ordered by Definition 5.6 or 5.7 form a semilattice $(\mathbb{F}, \wedge)$, where the semilattice operation between $\psi_1, \psi_2 \in \mathbb{F}$ is given by $\psi_1 \wedge \psi_2 = \psi_3$ iff $\psi_3(D) = \psi_1(D) \cap \psi_2(D)$.*

Proof. It follows from the definitions that if for any ψ_1 and ψ_2 the projection ψ_3 exists, then projections of D form a semilattice. Let us describe the corresponding ψ_3.

Let us denote $D_1 = \psi_1(D)$ and $D_2 = \psi_2(D)$ and $D_3 = D_1 \cap D_2$. Let us suppose that there exist $x, y \in D_3$ such that $x \sqcup y \notin D_3$. But as $D_3 \subseteq D_1$ and $D_3 \subseteq D_2$, then, since ψ_1 is a projection of D and ψ_2 is a projection of D, we have $x \sqcup y \in D_1$ and $x \sqcup y \in D_2$, i.e., $x \sqcup y \in D_1 \cap D_2 = D_3$. Thus, $(\forall x, y \in D_3) x \sqcup y \in D_3$. Then, according to Theorem 5.3 there is a projection ψ_3 such that $\psi_3(D) = D_3$. □

5.3.4 Analogue of Theorem II for Revised Projections

An important question is *how a projection changes the representation context of a pattern structure?* We limit the discussion of this question for the case when a set of description D is a complete semilattice. Ganter and Kuznetsov (2001) describe this change by means of Theorem 2. The formulation of this theorem was corrected by Kaiser and Schmidt (2011). Below we give the corrected version of the theorem.

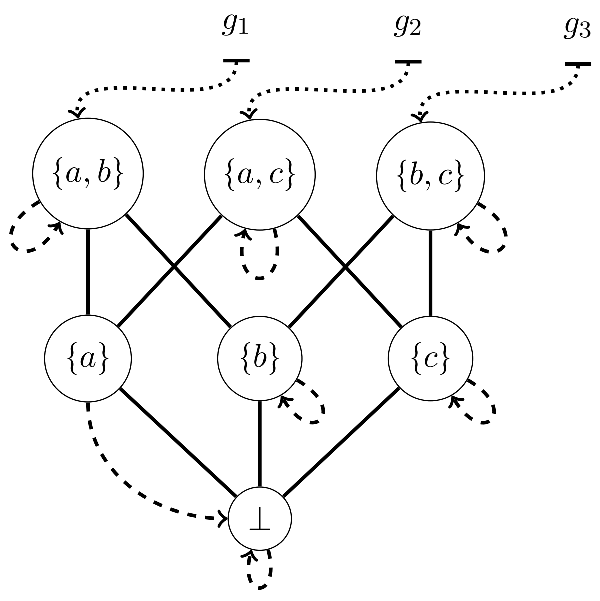
Theorem 5.4 (Theorem 2 from Ganter and Kuznetsov (2001))**.** *For pattern structures $(G, (D, \sqcap), \delta_1)$ and $(G, (D, \sqcap), \delta_2)$ the following statements are equivalent:*

1. *$\delta_2 = \psi \circ \delta_1$ for some ψ on $(D, \sqcap)$.*
2. *$(\forall g \in G)(\delta_2(g) \sqsubseteq \delta_1(g))$ and there is a representation context (G, M, I) of $(G, (D, \sqcap), \delta_1)$ and some $N \subseteq M$ such that $(G, N, I \cap (G \times N))$ is a representation context of $(G, (D, \sqcap), \delta_2)$.*

In Theorem 5.4 one compares two pattern structures that differ in mapping functions. However, in the o-projected pattern structures we can modify the lattice structure itself. *How can we adjust the formulation of Theorem 5.4 in such a way that it can be applied to revised projections?* First, we should notice that in a pattern structure and in an o-projected pattern structure the set of objects is preserved. Second, the minimal representation context of a pattern structure can have less attributes than the minimal representation context of an o-projected pattern structure, as shown in Example 5.4.

Example 5.4. *Let $M = \{a, b, c\}$ and the description semilattice be $D = (2^M, \cap)$. Let $\psi : 2^M \to 2^M$ be the following mapping: $\psi(\{a\}) = \varnothing$ and for any $A \neq \{a\}$ we put $\psi(A) = A$. This projection is visualised in Figure 5.4a by dashed arrows. Let us consider the following pattern structure $(\{g_1, g_2, g_3\}, (2^M, \cap), \{g_1 \mapsto \{a, b\}, g_2 \mapsto \{a, c\}, g_3 \mapsto \{b, c\}\}$.*

The minimal representation context of this pattern structure contains 3 attributes $M = \{a, b, c\}$, while the minimal representation context of the o-projected pattern structure contains 4 attributes $M_\psi = \{b, c, ab, ac\}$. The corresponding contexts are shown in Figures 5.4b and 5.4c.



(a) A semilattice D and its projection ψ.

	a	b	c
g_1	x	x	
g_2	x		x
g_3		x	x

(b) Representation context of the pattern structure

	ab	ac	b	c
g_1	x		x	
g_2		x		x
g_3			x	x

(c) Representation context of the projected pattern structure

Figure 5.4: An example of a projection that can increase the number of attributes in the minimal represenation context.

We can see that to introduce the "revised Theorem 2" from (Ganter and Kuznetsov 2001) we have to define a special relation between contexts.

Definition 5.8. *Given two contexts $\mathbb{K}_1 = (G, M_1, I_1)$ and $\mathbb{K}_2 = (G, M_2, I_2)$, $\mathbb{K}_1$ is said to be simpler than $\mathbb{K}_2$, denoted by $\mathbb{K}_1 \leqslant_S \mathbb{K}_2$, if for any $m_{1,i} \in M_1$ there is a set $B_2 \subseteq M_2$ such that $(\{m_{1,i}\})^1 = (B_2)^2$. Here by $(\cdot)^1$ and $(\cdot)^2$ we denote the derivation operators in the contexts $\mathbb{K}_1$ and $\mathbb{K}_2$, respectively.*

Example 5.5. *The context in Figure 5.4c is simplier w.r.t. Definition 5.8 than the context in Figure 5.4b because every column of the context in Figure 5.4c is the intersection of a subset of columns of the context in Figure 5.4b.*

This relation between contexts is a preorder. Indeed, it is reflexive, transitive, but not necessarily antisymmetric: given two contexts $\mathbb{K}_1$ and $\mathbb{K}_2$, if $\mathbb{K}_1$ and $\mathbb{K}_2$ have the same closure system of attributes, i.e., the same set of intents in the concept lattice, then according to the definition $\mathbb{K}_1 \leqslant_S \mathbb{K}_2$ and $\mathbb{K}_1 \geqslant_S \mathbb{K}_2$. However, we can consider only the context with the minimal number of attributes in the class of equivalence, i.e., the attribute-reduced context. For simplicity in the rest of the chapter we consider only attribute-reduced contexts.

This definition of the simplicity order on contexts can be related to context bonds (Ganter and Wille 1999) in the following way. Three formal contexts $\mathbb{K}_i = (G_i, M_i, I_i)$ form a bond if $\mathbb{K}_1 \leqslant_S \mathbb{K}_2$ and $\mathbb{K}_2^T \leqslant_S \mathbb{K}_3^T$, where $\mathbb{K}^T = (M, G, I^T)$. Simplicity order can also be considered as a generalization of "closed-relation-of" order between contexts:

Definition 5.9 (Definition 50 from Ganter and Wille (1999))**.** *A binary relation $J \subseteq I$ is called a **closed relation** of the context (G, M, I) if every concept of the context (G, M, J) is also a concept of (G, M, I).*

From Definitions 5.8 and 5.9 it can be seen that if J is a closed relation of (G, M, I), then $(G, M, J) \leqslant_S (G, M, I)$, but not always in the other direction. The following theorem gives a relation between kernel operators of D and the change in the representation context of o-projected pattern structures.

Theorem 5.5. *Given a pattern structure $\mathbb{P} = (G, (D, \sqcap), \delta)$ such that $(D, \sqcap)$ is a complete semilattice the following holds:*

1. *for any projection ψ of D we have $\mathbb{R}(\psi(\mathbb{P})) \leqslant_S \mathbb{R}(\mathbb{P})$.*
2. *for any context $\mathbb{K} = (G, M, I)$ such that $\mathbb{K} \leqslant_S \mathbb{R}(\mathbb{P})$, there is a projection ψ of D such that $\mathbb{K}$ is a representation context of $\psi(\mathbb{P})$.*

Proof. 1. The first statement follows from the fact that any extent of $\psi(\mathbb{P})$ is an extent of $\mathbb{P}$ (Proposition 5.1).

2. Given a pattern structure $\mathbb{P}$ and a context $\mathbb{K}$ such that $\mathbb{K} \leqslant_S \mathbb{R}(\mathbb{P})$, let us define the set $D_M = \{d \in D \mid (\exists m \in M)(m')^\diamond = d\}$ (notice that for $\mathbb{K}$ and $\mathbb{P}$ there is the same set G, thus, given $A \subseteq G$, both A' and $A^\diamond$ are defined in $\mathbb{K}$ and $\mathbb{P}$ correspondingly). Since $\mathbb{K} \leqslant_S \mathbb{R}(\mathbb{P})$, m' is an extent of $\mathbb{P}$. Thus, we can see that there is a bijection between D_M and M given by $m' = d^\diamond$. We denote this bijection by $f(m) = d$, i.e. $f(m) = d \Leftrightarrow m' = d^\diamond$. Correspondingly, given a subset $N \subseteq M$, we denote by $f(N) = \{d \in D_M \mid f^{-1}(d) \in N\}$, i.e., $f(M) = D_M$.

   Let us define $D_\psi = \{d \in D \mid (\exists X \subseteq D_M) \bigsqcup X = d\}$. According to Theorem 5.3 there is a projection ψ such that $D_\psi = \psi(D)$.

   Let us consider the o-projected pattern structure $\psi(\mathbb{P})$. The set D_M is $\sqcup$-dense for $\psi(D)$, i.e., the context (G, D_M, I_{D_M}), where $(g, d) \in I_{D_M} \Leftrightarrow \psi \circ \delta(g) \sqsupseteq d$, is a representation context of $\psi(\mathbb{P})$. There is the bijection between D_M and M. Let us show that the relation I is similar to the relation I_M, i.e., $(g, m) \in I \Leftrightarrow (g, f(m)) \in I_{D_M}$.

   It can be seen that for all $g \in G$ and all $d \in f(g')$, we get $\psi \circ \delta(g) \sqsupseteq d$, because for any $d \in f(g')$ we have $g \in d^\diamond$. Moreover, for any $\tilde{d} \in D \backslash f(g')$ we have $d \not\sqsupseteq \psi \circ \delta(g)$. Thus, the context $\mathbb{K}$ and the context (G, D_M, I_{D_M}) are similar, and hence for any context $\mathbb{K} \leqslant_S \mathbb{R}(\mathbb{P})$ there is a projection such that $\mathbb{K}$ is a representation context of $\psi(\mathbb{P})$.

□

5.4 Conclusion

In this chapter we have introduced o-projections of pattern structures that are based on kernel operators $\psi : D \to D$. O-projections are a generalization of projections of pattern structures and allow one to change the semilattice of descriptions in o-projected pattern structures. Thus, the complexity of similarity (semilattice) operation can be reduced. Moreover, O-projections also correct a formal problem of projections.

We have shown that o-projections form a semilattice. This can be important when several independent o-projections are applied to a pattern structure. For example, if projections are discussed with several experts it may happen that several types of projections should be combined. In the case of several independent projections we know that there is the only one o-projection w.r.t. the semilattice of o-projections that is a combination of these projections.

Finally, we have shown that the representation context of an o-projected pattern structure can have more attributes than the representation context of the pattern structure itself. To describe this change in the representation context after o-projection we have introduced a new order on contexts, with the use of which we have described the way the representation context can change.

An open direction of the future work is to formalize *transformations* of pattern structures, i.e., special homomorphisms between the semilattice of descriptions D and a different semilattice D_1. In particular, it allows one to formalize the mappings of the form $\psi : D \to \mathbb{R}$, an instance of which are kernel functions used in Support Vector Machines (SVM).

Let us now exemplify the pattern structure mathematical framework with some applications and in the next chapter we consider applying pattern structures for completion of RDF data.

Chapter 6

Mining definitions from RDF annotations using Formal Concept Analysis

Contents

6.1 Introduction

World Wide Web has tried to overcome the barrier of data sharing by converging data publication into Linked Open Data (LOD) (Bizer et al. 2009). The LOD cloud stores data in the form of *subject-predicate-object* triples based on the RDF language[10], a standard formalism for information description of web resources. In this context, DBpedia is the largest reservoir of linked data in the world currently containing more than 4 million triples. All of the information stored in DBpedia is obtained by parsing Wikipedia, the largest open Encyclopedia created by the collaborative effort of thousands of people with different levels of knowledge in several and diverse domains.

More specifically, DBpedia content is obtained from semi-structured sources of information in Wikipedia, namely *infoboxes* and *categories*. Infoboxes are used to standardize entries of a given type in Wikipedia. For example, the infobox for "automobile" has entries for an image depicting the car, the name of the car, the manufacturer, the engine, etc. These *attributes* are mapped by the DBpedia parser to a set of "properties" defined in an emerging ontology[11] (Benz et al. 2010)

[10]Resource Description Framework - http://www.w3.org/RDF/

[11]Emerging in the sense of "dynamic" or "in progress".

(infobox dataset) or mapped through a hand-crafted lookup table to what is called the DBPedia Ontology (mapped-based ontology). Categories are another important tool in Wikipedia used to organize information. Users can freely assign a category name to an article relating it to other articles in the same category. Example of categories for cars are “Category:2010s automobiles”, “Category:Sports cars” or “Category:Flagship vehicles”. While we can see categories in Wikipedia as an emerging “folksonomy”, the fact that they are curated and “edited” make them closer to a controlled vocabulary. DBpedia exploits the Wikipedia category system to “annotate”[12] objects using a taxonomy-like notation. Thus, it is possible to query DBpedia by using *annotations* (e.g., all cars annotated as “Sport cars”). While categorical information in DBpedia is very valuable, it is not possible to use a category as one could expect, i.e., as a definition of a class of elements that are instances of the class or, alternatively, that are “described” by the category. In this sense, such a category violates the actual spirit of semantic Web.

Let us explain this with an example. The Web site of DBpedia in its section of “Online access” contains some query examples using the SPARQL query language. The first query has the description “People who were born in Berlin before 1900” which actually translates into a graph-based search of entities of the type “Person”, which have the property “birthPlace” pointing to the entity representing the “city of Berlin” and another property named “birthDate” with a value less than 1900. We can see here linked data working at “its purest”, i.e., the form of the query provides the right-hand side of a definition for “People who were born in Berlin before 1900”. Nevertheless, the fourth query named “French films” does not work in the same way. While we could expect also a graph-based search of objects of the type “Film” with maybe a property called “hasCountry” pointing to the entity representing “France”, we have a much rougher approach. The actual SPARQL query asks for objects (of any type) annotated as “French films”.

In general, categorization systems express “information needs” allowing human entities to quickly access data. French films are annotated as such because there is a need to find them by these keywords. However, for a machine agent this information need is better expressed through a *definition*, like that provided for the first query (i.e., “People who were born in Berlin before 1900”). Currently, DBPedia mixes these two paradigms of data access in an effort to profit from the structured nature of categories, nevertheless further steps have to be developed to ensure coherence and completeness in data.

Accordingly, in this chapter we describe an approach to bridge the gap between the current syntactic nature of categorical annotations with their semantic correspondent in the form of a concept definition. We achieve this by mining patterns derived from entities annotated by a given category, e.g., All entities annotated as “Lamborghini cars” are of “type automobile” and “manufactured by Lamborghini”, or all entities annotated as “French films” are of “type film” and of “French nationality”. We describe how these category-pattern equivalences can be described as “definitions” according to *implication rules* among attributes which can be mined using FCA (Ganter and Wille 1999). The method considers the analysis of heterogeneous complex data (not necessarily binary data) through the use of “pattern structures” (Ganter and Kuznetsov 2001). A concept lattice can be built from the data and then used for discovering *implication rules* (i.e., association rules whose confidence is 100%) which provide a basis for “subject definition” in terms of necessary and sufficient conditions.

This chapter is based on Alam et al. (2015) and is structured as follows: Section 6.2 gives a brief introduction to the theoretical background necessary to sustain the rest of the chapter. Section 6.3 describes the approach used for data completion in the DBpedia knowledge base.

[12]Notice that in DBPedia the property used to link entities and categories is called “subject”. We use “annotation” instead of “subject” to avoid confusions with the “subject” in an RDF triple.

Section 6.4 provides experimental results on four datasets created from DBpedia and a brief discussion over our findings. Finally, Section 6.5 concludes the chapter offering some perspectives over our approach.

6.2 Preliminaries

Linked Open Data (LOD) (Bizer et al. 2009) is a formalism for publishing structured data on-line using the resource description framework (RDF). RDF stores data in the form of RDF triples represented as $\langle subject, predicate, object\rangle$. The profile of an RDF triple $\langle s, p, o\rangle$ is given by $(U \cup B) \times (U \cup B) \times (U \cup B \cup L)$ where a set of RDF triples is an RDF graph, denoted by $\mathcal{G}$. Here, U denotes a set of URI references, B refers to the blank node and L to literals. For the sake of simplicity, in the current study we do no take into account blank nodes (B). An RDF triple is represented as $U \times U \times (U \cup L)$. For convenience, in the following we denote the set of predicate names as P and the set of object names as O. LOD can then be queried and accessed through SPARQL[13], which is a standard query language for RDF data. SPARQL is based on matching graph patterns (present in the *WHERE* clause of a query) against RDF graphs. For example, let us consider the SPARQL query given in Listing 6.1, for all the entities of type Automobile manufactured by *Lamborghini*, annotated as "Sport_cars" and as "Lamborghini_vehicles",

```
SELECT ?s WHERE {
  ?s dc:subject dbpc:Sports_cars .
  ?s dc:subject dbpc:Lamborghini_vehicles .
  ?s rdf:type dbo:Automobile .
  ?s dbo:manufacturer dbp:Lamborghini }
```

Listing 6.1: SPARQL for the formal context in Figure 6.1. Prefixes are defined in Table 6.1.

Predicates		Objects	
Index	URI	Index	URI
A	dc:subject	a	dbpc:Sport_Cars
		b	dbpc:Lamborghini_vehicles
B	dbp:manufacturer	c	dbp:Lamborghini
C	rdf:type	d	dbo:Automobile
D	dbp:assembly	e	dbp:Italy
E	dbo:layout	f	dbp:Four-wheel_drive
		g	dbp:Front-engine

Namespaces:	
dc:	http://purl.org/dc/terms/
dbo:	http://dbpedia.org/ontology/
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
dbp:	http://dbpedia.org/resource/
dbpc:	http://dbpedia.org/resource/Category:

Table 6.1: Index of pairs predicate-object and namespaces.

Formal Concept Analysis (FCA) works with formal contexts (G, M, I) and G is called a set of objects. However in this chapter we sligty modify this terminology and say that G is a set of *entities* in order to avoid confusions with "objects" as defined for RDF triples.

In Figure 6.1 a formal context corresponding to a small LOD is shown, where $G = U$, $M = (P \times O)$ and $(u, (p, o)) \in I \iff \langle u, p, o\rangle \in \mathcal{G}$, i.e., $\langle u, p, o\rangle$ is a triple built from different

[13]http://www.w3.org/TR/rdf-sparql-query/

	A		B	C	D	E	
	a	b	c	d	e	f	g
Reventon	×	×	×	×	×	×	
Countach	×	×	×	×	×		
350GT	×	×	×	×			×
400GT	×	×	×	×			
Islero	×	×	×	×			
Veneno	×	×					
Aventador Roadster	×	×					
Estoque			×	×		×	×
Gallardo			×	×	×		

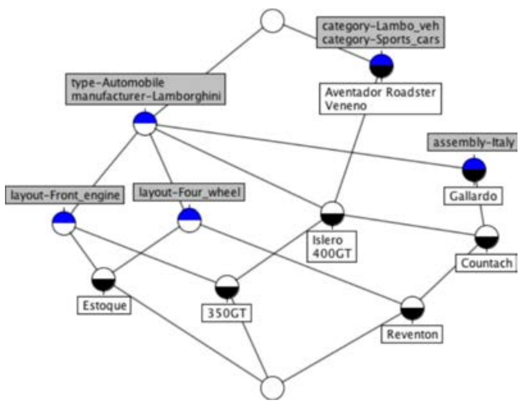


Figure 6.1: The formal context shown on the left is built after scaling from DBpedia data given in Table 6.1. Each cross (×) corresponds to a triple subject-predicate-object. On the right the corresponding concept lattice is shown.

triples manually extracted from DBpedia about nine different Lamborghini cars (35 RDF triples in total). Given a subject-predicate-object triple, the formal context contains subjects in rows, the pairs predicate-object in columns and a cross in the cell where the triple subject in row and predicate-object in column exists.

Figure 6.1 depicts the concept lattice in reduced notation calculated for this formal context and contains 12 formal concepts. Consider the first five cars (*subjects*) in the table for which the maximal set of attributes they share is given by the first four *predicate-object* pairs.

Given a concept lattice, rules can be extracted from the intents of concepts which are comparable. For two sets $X, Y \subseteq M$, a rule has the form $X \implies Y$ where X and Y are subsets of attributes. A rule $X \implies Y$ has a *support* given by the proportion of entities having the set of attributes in X and Y (i.e., $|X' \cap Y'|$[14]) w.r.t. the whole set of entities, and a *confidence* which is the proportion of entities having the (same) set of attributes in X and Y, w.r.t. X (i.e., $|X' \cap Y'|/|X'|$). For instance, consider the association rule $e \implies a, b, c, d$. Its support is given by $|\{a, b, c, d\}' \cap \{e\}'|$ which is the same as $|\{Reventon, Countach\}| = 2$. Since $|\{e\}'| = 3$, we have that the confidence of this association rule is $2/3 \approx 0.67$. A rule $X \implies Y$ of confidence 1 (when $|X' \cap Y'| = |X'|$ or $X' \subseteq Y'$) is called an *implication*. Otherwise, the confidence is less than 1 and the rule is called an *association rule*. When $X \implies Y$ and $Y \implies X$ are implications, we say that $X \iff Y$ is an equivalence or a *definition*. This can happen when two attributes have the same "attribute concept", e.g., *type-Automobile* and *manufacturer-Lamborghini* in the concept lattice of Figure 6.1.

6.3 Improving DBpedia with FCA

6.3.1 Problem context

Consider the following fictional scenario. You are a bookkeeper in a library of books written in a language you do not understand. A customer arrives and asks you for a book about "Cars". Since you do not know what the books are about (because you cannot read them), you ask the customer to browse the collection on his own. After he finds a book he is interested to read, you will mark the symbol ⋆ on that book for future references. Then, in an empty page you will write

[14] $|\cdot|$ denotes set cardinality.

($\star$ - *Cars*). After several cases like this, you will probably end up with a page full of symbols representing different topics or categories of your books, among them ($\ominus$ - *Sports*), ($\diamond$ - *Football*) and ($\circ$ - *History*). Now you can even combine symbols when customers ask you for “Sport Cars” which you translate into $\star\ominus$. Actually, the demand for books about “Sport Cars” is so high that you create a new symbol † just for it. So doing, you have created your own categorization system of a collection of books you do not understand.

In general, given a topic, you are able to retrieve books without many troubles, however since you do not understand the books, you are restricted to the set of symbols you have for doing this. Furthermore, if you are not careful some problems start to arise, such as books marked with $\diamond$ and without $\ominus$. Finally, people do not get books marked with † when they look for “Cars”, since they only search for the symbol $\ominus$.

It is easy to stablish an analogy on how DBpedia profits from Wikipedia’s categorization system and the above scenario. DBpedia is able to retrieve entities when queried with an annotation (as the example of “French films”), however any information need not initially provided as a category is unavailable for retrieval (such as “French films about the Art Nouveau era”). Incoherences in categorical annotations are quite frequent in DBpedia, for example there are over 200 entities annotated as “French films” which are not typed as “Films”. Finally, DBpedia is not able to provide inferencing. For example, in Figure 6.1, the entities Veneno and Aventador, even though they are annotated as “Lamborghini vehicles”, cannot be retrieved when queried simply by “vehicles”. In such a way, it is exactly as if they were marked with a symbol such as †.

6.3.2 The completion of DBpedia data

Our main concern in this case lies in two aspects. Firstly, are we able to complete data using logical inferences? For example, can we *complete* the information in the dataset by indicating that the entities “Estoque” and “Gallardo” should be categorized as “Lamborghini vehicles” and “Sport cars”? Secondly, are we able to *complete* the descriptions of a given type? For example, DBpedia does not specify that an “Automobile” should have a “manufacturer”. In the following, we try to answer these two questions using implications and association rules.

Rule	Confidence	Support	Meaning
d $\Longrightarrow$ c	100%	7	Every automobile is manufactured by Lamborghini.
c $\Longrightarrow$ d	100%	7	Everything manufactured by Lamborghini is an automobile.
e $\Longrightarrow$ c,d	100%	3	All the entities assembled in Italy are Lamborghini automobiles.
c,d $\Longrightarrow$ a,b	71%	7	71% of the Lamborghini automobiles are categorized as “sport cars” and “Lamborghini vehicles”

Table 6.2: Association rules extracted from formal context in Figure 6.1.

Consider rules provided in Table 6.2. Of course, the first three implications are only true in our dataset. This is due to the fact that we use the “closed world” assumption, meaning that our rules only apply in “our world of data” where all cars are of “Lamborghini” brand, i.e., all other information about cars that we do not know can be assumed as false (Fürber and Hepp 2011). While these implications are trivial, they provide a good insight of the capabilities of our model. For instance, including a larger number of triples in our dataset would allow discovering that, while not all automobiles are manufactured by Lamborghini, they are manufactured by either a Company, an Organization or an Agent. These three *classes*[15] are types of the entity Lamborghini in DBpedia. Such a rule would allow providing a *domain* characterization to the otherwise empty description of the predicate "dbo:manufacturer" in the DBpedia schema.

The association rule given in the fourth row in Table 6.2 shows the fact that 29% of the subjects of type "Automobile" and manufactured by "Lamborghini" should be categorized by "Sports cars" and "Lamborghini vehicles" to complete the data. This actually corresponds to the entities "Estoque" and "Gallardo" in Figure 6.1. Based on this fact, we can use association rules also to create new triples that allow the completion of the information included in DBpedia.

6.3.3 Pattern structures for the completion process

The aforementioned models to support linked data using FCA are adequate for small datasets as the example provided. Actually, LOD do not always consists of triples of resources (identified by their URIs) but contains a diversity of *data types* and structures including dates, numbers, collections, strings and others making the process of data processing much more complex. This calls for a formalism able to deal with this diversity of complex and heterogeneous data, i.e., pattern structures.

For linked data, here, we propose to use the approach called "heterogeneous pattern structure" framework introduced in Codocedo and Napoli (2014) as a way to describe objects in a heterogeneous space, i.e., where there are relational, multi-valued and binary attributes. It is easy to observe that this is actually the case for linked data where the set of literals L greatly varies in nature depending on the predicate. For the sake of simplicity we provide only the most important details of the model used for working with linked data.

When the range of a predicate (hereafter referred to as "relation") $p \in P$ is such that $range(p) \subseteq U$, we call p an "object relation". Analogously, when the range is such that $range(p) \subseteq L$, p is a "literal relation". For any given relation p (object or literal), we define the pattern structure $\mathbb{K}_p = (G, (D_p, \sqcap), \delta_p)$, where $(D_p, \sqsubseteq)$ is an ordered set of descriptions defined for the elements in $range(p)$, and δ_p maps entities $g \in G$ to their descriptions in D_p. Based on that, the triple (G, H, Δ) is called a "heterogeneous pattern structure", where $H = \bigtimes_{p \in P} D_p$ is the Cartesian product of all the descriptions sets D_p, and Δ maps an entity $g \in G$ to a tuple where each component corresponds to a description in a set D_p.

For an "object relation", the order in $(D_p, \sqsubseteq)$ is given by standard set inclusion and thus, the pattern structure $\mathbb{K}_p$ is just a formal context. Regarding "literal relations", such as numerical properties, the pattern structure may vary according to what is more appropriate to deal with that specific kind of data. For example, considering the predicate *dbo:productionStartYear* discussed in the previous section, $\mathbb{K}_{\text{dbo:productionStartYear}}$ should be modelled as an interval pattern structure. For the running example, the heterogeneous pattern structure is presented in Table 6.3. Cells in grey mark a *heterogeneous pattern concept* the extent of which contains cars "350GT, 400GT, Islero". The intent of this heterogeneous pattern concept is given by the tuple $(\{a, b\}, \{c\}, \{d\}, \langle[1963, 1967]\rangle)$, i.e., *"Automobiles manufactured by Lamborghini between 1963 and 1967"*. The model of heterogeneous pattern structures is the basis of the experiments which are presented in the next section.

[15] *In the OWL language sense.*

	$\mathbb{K}_A$		$\mathbb{K}_B$	$\mathbb{K}_C$	$\mathbb{K}_D$	$\mathbb{K}_E$		$\mathbb{K}_{\text{dbo:productionStartYear}}$
	a	b	c	d	e	f	g	
Reventon	×	×	×	×	×	×		$\langle[2008, 2008]\rangle$
Countach	×	×	×	×	×			$\langle[1974, 1974]\rangle$
350GT	×	×	×	×			×	$\langle[1963, 1963]\rangle$
400GT	×	×	×	×				$\langle[1965, 1965]\rangle$
Islero	×	×	×	×				$\langle[1967, 1967]\rangle$
Veneno	×	×						$\langle[2012, 2012]\rangle$
Aventador Roadster	×	×						-
Estoque			×	×		×	×	-
Gallardo			×	×	×			-

Table 6.3: Heterogeneous pattern structure for the running example. Indexes for properties are shown in Table 6.1.

6.4 Experimentation

To evaluate our model, four datasets were created from DBpedia, namely "Cars", "Videogames", "Smartphones" and "Countries" (see characteristics of the datasets in Table 6.4). Each dataset was created using a single SPARQL query with a unique restriction (either a fixed subject or a fixed type). A dataset consists of a set of triples whose predicate is given by the properties in Table 6.4. The heterogeneous aspect of data is illustrated by the fact that in two of the four datasets there are properties with numerical ranges.

Dataset	**Cars**	**Videogames**	**Smartphones**	**Countries**
Dataset building conditions				
Restriction	dc:subject dbpc:Sports_cars	dc:subject dbpc:FPS†	dc:subject dbpc:Smartphones	rdf:type Country
Predicates	rdf:type dc:subject bodyStyle transmission assembly designer layout	rdf:type dc:subject cp‡ developer requirement genre releaseDate	rdf:type dc:subject manufacturer operativeSystem developer cpu	rdf:type dc:subject language govenmentType leaderType foundingDate gdpPppRank
Dataset Characteristics				
# Subjects	529	655	363	3,153
# Objects	1,291	3,265	495	8,315
# Triples	12,519	20,146	4,710	50,000
# Concepts	14,657	31,031	1,232	13,754
Exec. time [s]	17.32	17.14	0.7	59.82
Results				
Rules Eval.	19	46	47	50
P@20	0.85	0.7	0.79	0.9

† Front_Person_Shooters
‡ computerPlatform

Table 6.4: Summary table of experimental procedures. Upper table shows the predicates used to construct each datasets. Properties without a prefix have the default namespace "dbo:". Underlined properties have numerical ranges. Middle table show each dataset characteristics. Lower table shows experimental results. P@20 stands for "Precision at the first 20 implication".

For each dataset we calculated the set of all implications derived from the heterogeneous pattern concept lattice. Each rule of the form $X \implies Y$ was ranked according to the confidence

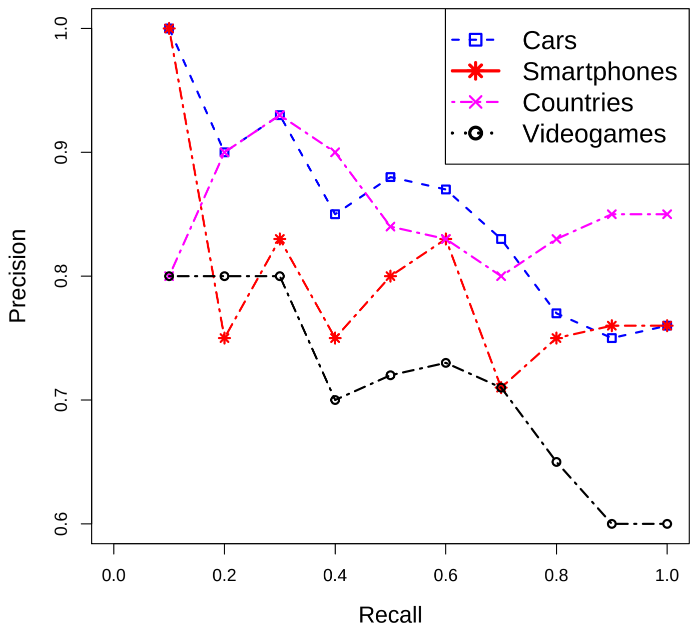


Figure 6.2: Precision at 11-points for each dataset (P@11p)

of the rule $Y \implies X$ (the latter is referred as the “inverted rule” of the former). Thus, given that implications have always a confidence of 100%, the confidence of the inverted rule tells us how close we are from a definition, i.e., $X \iff Y$ or both rules are implications. Having an implication or not leads to the decision whether a set of RDF triples should be completed or not. For example, the following implication from the Cars dataset has an inverted rule of 92% of confidence:

$$\textit{rdf:type-dbo:MaseratiVehicles} \implies \textit{dbo:manufacturer-dbp:Maserati}$$

Accordingly, we can make of this implication a definition stating that the remainder 8% of the entities manufactured by *Maserati* should also be “typed” as *MaseratiVehicles* (recall in here that we have constructed our “world of data” by taking all “Sport Cars” from DBpedia, thus things built by Maserati which are not vehicles do not belong in our data). Of course, there are cases in which this will not be true. For example with a 90% of confidence in the opposite direction we have the implication:

$$\textit{dbo:layout-dbp:Quattro} \implies \textit{dbo:manufacturer-dbp:Audi}$$

The creation of a definition from this rule (i.e., making the remainder 10% of the cars manufactured by Audi have a layout 4x4) would be wrong. While we expect that the high confidence of the opposite association rule distinguish the case when a definition should be made, a human ruling to include background information will always be needed.

Considering that there is no ground truth for any of the datasets, the reported results are given for assessing the feasibility of our approach. For each of the ranked basis of implications in the experimentation we performed a human evaluation. With the help of DBpedia, it was evaluated if an implication was likely to become a definition. The answer provided for each of the rule was binary (yes or no). For instance, in the previous examples the first implication would render a “yes” answer, while the second, a “no”. Afterwards, we measured the precision for the first 20 ranked implications (P@20) as the proportion of the rules that were likely to become a definition (those evaluated as yes) over 20 (the total number of rules taken). Actually, the precision value works as an indicator of how likely implications are useful for RDF data completion (see Table 6.4).

By contrast, since we do not have a ground truth of all the triples that should be added to each dataset, we are not able to report on recall. Nevertheless, to complement this evaluation, we provide the values of the precision at 11 points (P@11p) (Christopher D. Manning et al. 2008). We consider each human evaluation as the ground truth for its respective dataset and thus, each list of implications has a 100% recall. Precision at 11 points provides a notion on how well distributed are the answers in the ranking. Figure 6.2 contains the curves for each of the datasets. Values for precision at 20 points are high for all datasets and particularly for the dataset “Countries”. This may be due to the fact that Countries was the only dataset built for resources with a fixed type.

A precision of 0.9 indicates that 9 out of 10 implications can be transformed into definitions by creating RDF triples that would complete the entities descriptions. Precision at 11-points shows that confidence is a good indicator on the usefulness of implications for data completion. For example, regarding the worst result i.e., the Videogames dataset, when the evaluator provides the last “yes” answer for an implication, he/she has also given a “yes” to 6 out 10 (from a total of 46). For our best result (Countries dataset) it is over 8 out of 10. Results show that confidence is a good indicator for the selection of implications in terms of data completion.

Further experimentation should be performed to assess if the triples being created are “correct” or not. As already mentioned, we assume that resources being completed are correctly linked to the implication. While this may not always be true, our approach is still useful under those circumstances given that it would allow discovering such “incorrectly” annotated entities. Finally, regarding execution times, Table 6.4 shows that even for the larger dataset, the execution time is less than a minute, and this time is perfectly acceptable for the analysis of implications.

6.5 Related work, discussion and conclusion

Paulheim and Bizer (2013) use an inference mechanism which considers the links between instances to obtain their class types, assuming that some relations occur only with certain classes. Moreover, there are some studies which focus on the correction of numerical data present in DBpedia using *outlier detection method*, which identify those facts which deviate from other members of the sample (Wienand and Paulheim 2014). By contrast, the presented approach focuses on completing RDF data with the help of association rule mining. Zaveri et al. (2013) propose a manual and semi-automatic methodology for evaluating the quality of LOD resources w.r.t. a taxonomy of “quality problems”. Quality assessment is based on user inputs (crowd-sourcing) and measures the correctness of schema axioms in DBpedia. Y. Yu and Heflin (2011a,b) try to detect triples which are regarded as erroneous w.r.t. similar triples. The detection is based on probabilistic rule learning and on the discovery of generalized functional dependencies that are used to characterize the abnormality of the considered triples.

Different interesting perspectives are opened following this work. As we have discussed, categories represent some pre-loaded information needs in Wikipedia, i.e., a pre-answered questions whose answer is relevant for a group of people. Thus, an interesting application would be to translate these information needs into description logics definitions, instead of attributes. It is possible to think that, instead of annotating each French film with a “FrenchFilm” tag, we could define the category as

$$\texttt{FrenchFilm} \equiv \texttt{Film} \sqcap \texttt{hasCountry}.\{\texttt{FRANCE}\}$$

Given that these definitions are more restrictive than typing (*rdf:type*), our approach should be adapted to deal with "near-definitions" in which both directions ($X \implies Y$ and $Y \implies X$) are association rules with high confidence. While we have presented our approach applied to DBpedia, its applicability is more general than this specific knowledge resource. In fact, using category taxonomies to annotate resources in LOD is a common practice for several knowledge bases, to the extent that a meta-model for vocabularies (Simple Knowledge Organization System - SKOS) has been proposed by the World Wide Web Consortium [16]. SKOS organizes *"Concepts"* in a hierarchical taxonomy, while resources are said to be *"subjects"* of these concepts.

To conclude, in the current chaptre we introduce a mechanism based on association rule mining for the completion of the RDF dataset. Moreover, we use heterogeneous pattern structures to deal with heterogeneity in LOD. Several experiments have been conducted over four datasets and an evaluation was conducted for each of the experiments. This chapter shows the capabilities of FCA for completing complex RDF structures and the importance of pattern structure formalism for dealing with LOD. Let us now switch to analysis of syntax trees and drug-drug interactions by means of pattern structures.

[16] http://www.w3.org/2004/02/skos/

Chapter 7

Exploring Pattern Structures of Syntactic Trees for Relation Extraction

7.1 Introduction

When a doctor wants to prescribe a drug to a patient, he/she would like to know when this drug interacts with other drugs that the patient may already take. A lot of research has been done on each drug, resulting in a lot of articles (often more than 1000 articles per drug). It would not be feasible for a human agent to read all these articles. For this reason it could be interesting to automatically find which drugs are interacting in these articles. Accordingly, in the extraction of drug-drug interactions –DDIs in the following– the task is to find pairs of drugs that are described as interacting in a sentence or a text.

In 2011, for the first time, a challenge on this task was initiated (Segura-Bedmar et al. 2011). Several methods were proposed to perform this task (Björne et al. 2011; F. M. Chowdhury et al. 2011; M. F. M. Chowdhury and Lavelli 2011; Garcia-Blasco et al. 2011; Minard et al. 2011; P. Thomas et al. 2011). The best performing system, i.e., the system with the highest F_1-measure on the given test set, combined several different subsystems in which information from different feature spaces was exploited (P. Thomas et al. 2011). Their highest F_1 on the test set was 65.7, and their F_1 for a document-wise 10-fold cross validation on the training data was 60.6. Linguistics features were used, such as part-of-speech, together with different tree kernels of the dependency parses, i.e., trees describing the grammatical dependencies between words, of the sentences. Another system that was successful in the challenge was based on a union of two different machine learning techniques (F. M. Chowdhury et al. 2011). The first machine learning technique is a feature-based SVM using different words, morphosyntactic features (internal structural features of words, like number and case) and contextual features (words in between the two considered drugs). The second machine learning technique is a kernel-based method combining three different kernels, namely "shallow linguistic information" (like part-of-speech and word-inflection information), "mildly extended dependency trees" and "phrase structure".

It appears that the most successful systems combine both deep linguistic information, such as dependency trees or phrase structures, and shallow linguistic features, such as word features and morphological information. Thus, we propose to apply a symbolic method, based on pattern structures (Ganter and Kuznetsov 2001) to deal with the phrase structure, i.e., the syntactic level, in a different way. A pattern structure can manage a complex data type, such as a tree, and allows one to build a hierarchy of elements of this data type, in the present case a hierarchy of trees.

Such a pattern structure comes with a classification technique, called “Lazy Pattern Structure Classification” (LPSC) by Kuznetsov (2013), which classifies the syntactic trees containing drug-drug interactions. This is one original application of pattern structures to syntactic trees and to the task of text-mining (here the mining of DDIs). The method is novel and deserves more research work but we already obtained substantial results showing that the current approach is suitable and valuable.

This chapter is based on Leeuwenberg et al. (2015) and organized as follows. Firstly we explain the pipeline on which relies the proposed approach. Then we define the pattern structure for syntactic trees, namely STPS, and as well Lazy Pattern Structure Classification (LPSC). After that, we introduce a projection related to STPS and a set of tree-simplification operations to reduce computational time. Finally we evaluate the method on the corpus of the DDI challenge 2011 (Segura-Bedmar et al. 2011).

7.2 The Data and the Pipeline

Our data consists of medical texts containing potential drug-drug interactions, i.e., the training corpus of the DDI extraction challenge 2011 (Segura-Bedmar et al. 2011). This corpus consists of around 4200 sentences containing around 23000 potential interactions of which a small portion ($\sim$10%) is annotated as positive and the rest as negative. In these data drugs in the sentences are already tagged (see Example 7.1).

If we take a sentence from the data containing n drugs, there are $\binom{n}{2}$ pairs of drugs in the sentence that can potentially interact. Each such pair is represented by a separate sentence, where the two potentially interacting drugs in the sentence are replaced with a `drug_tag_r` tag, and all other drugs by a `drug_tag` tag (see Examples 7.2 and 7.3 where the corresponding tags are following the name of the tagged drug).

Example 7.1. *Antihistamines (`drug`) may enhance the effects of tricyclic_ antidepressants (`drug`), barbiturates (`drug`), alcohol (`drug`), and other CNS_ depressants (`drug`).*

Example 7.2. *`drug_tag_r` may enhance the effects of `drug_tag`, `drug_tag`, `drug_tag`, and other `drug_tag_r`.*

Example 7.3. *`drug_tag` may enhance the effects of `drug_tag`, `drug_tag_r`, `drug_tag`, and other `drug_tag_r`.*

Each such tagged sentence, representing a possible drug-drug interaction, is parsed by the Stanford constituency parser v3.4 (Klein and Christopher D Manning 2003; Socher et al. 2013). The resulting trees are simplified by means of operations that preserve the parts of the tree describing the potential interaction as much as possible. Trees representing drug-drug pairs can be considered as positive or negative. Trees are “positive” when an interaction is described between the two drugs replaced by the `drug_tag_r` tag (Example 7.2). Trees are “negative” when no such interaction is present (Example 7.3). The positive simplified tree of Example 7.2 is shown in Figure 7.1.

A pattern structure is defined on such syntactic trees, whose similarity operator is based on unordered rooted tree intersection. The trees are interpreted as unordered w.r.t the constituent order in the sentence in order to be able to generalize over some grammatical structures (eg. conjunctions or enumerations) without losing important grammatical relations (eg. verb argument relations, prepositions) as they are also encoded in the hierarchy of the tree. To improve

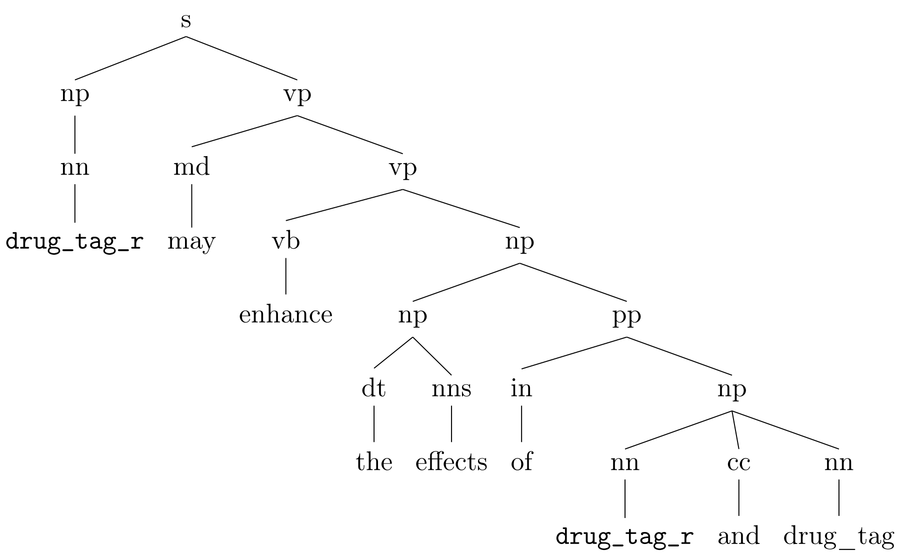


Figure 7.1: The simplified syntax tree from Example 7.2.

the computational time of similarity, a projection is introduced. Pattern structures, similarity and projections are discussed here after.

The set of trees, obtained from parsing the tagged sentences, is split into a "training set" and a "testing set" and LPSC is used to classify the trees. In the experiments, different settings based on tree simplifications are evaluated. Finally, a schematic view of the pipeline, starting from DDIs and going to LPSC classification, is shown in Figure 7.2.

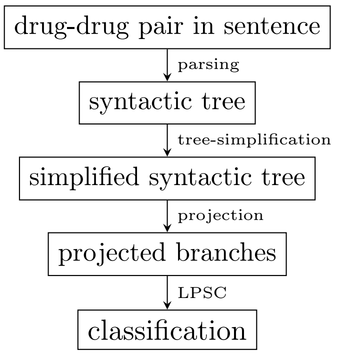


Figure 7.2: A schematic view of the pipeline.

7.3 A Pattern Structure for Syntactic Trees

7.3.1 Objects and Object Descriptions

In the current case, the set of objects G in the considered pattern structure $(G, (D, \sqcap), \delta)$ consists of drug-drug pairs, i.e., DDIs, extracted from the collection of sentences. Then the set of object descriptions D is composed of "unordered labeled trees". The resulting pattern structure will be called "Syntactic Tree Pattern Structure" or STPS for short.

Definition 7.1. *An* unordered labeled rooted tree *t is a simple connected graph $t = \langle N, E \rangle$,* *where N is a set of nodes, and E a set of ordered pairs from $N \times N$, called edges. It should satisfy two conditions:*

- *t does not contain any cycle (it is a tree)*
- *t has one distinguished node $r \in N$, called the root node, that is an ancestor of every node $n \in N$.*

In unordered labeled rooted trees, nodes carry a label while there exists no order between the children of each node. This means that the trees in Figure 7.3 are considered to be equivalent.

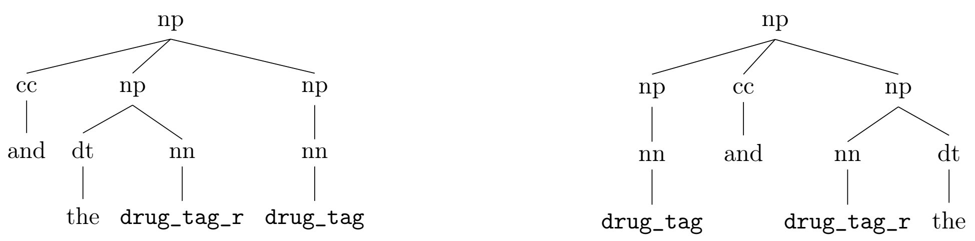


Figure 7.3: Two equivalent unordered labeled rooted trees.

The mapping δ gives for each potential drug-drug pair the corresponding unordered syntactic tree of the sentence in which it occurs, where the drugs are replaced by the tags. Intuitively, one could think of δ as the function that parses the sentence and simplifies the resulting tree.

7.3.2 Similarity Operators

A similarity operator $\sqcap_t$ is defined on the set of object descriptions D. This operator is based on rooted tree intersection. Balcázar et al. (2006) define rooted tree intersection for unordered unlabeled trees and gives a corresponding algorithm. Our definition and implementation follow those by Balcázar et al. (2006), except that we consider trees with labeled nodes. To define our rooted tree intersection for unordered labeled trees we need to define the notion of rooted subtree first.

Definition 7.2. *Rooted tree $t_1 = \langle N_1, E_1 \rangle$ is a* rooted subtree *of rooted tree $t_2 = \langle N_2, E_2 \rangle$ (from now written as $t_1 \subseteq_t t_2$) iff the following conditions hold:*

- *$N_1 \subseteq N_2$*
- *$E_1 \subseteq E_2$*
- *t_1 and t_2 have the same root.*

Using this notion of subtree, we can define a rooted intersection operator on trees.

Definition 7.3. *The* rooted tree intersection *between tree t_1 and t_2, from now written as $t_1 \cap_t t_2$, is the set containing all maximal trees*[17] *from $\{t \mid t \subseteq_t t_1\} \cap \{t \mid t \subseteq_t t_2\}$, i.e., the intersection between all subtrees of t_1 and all subtrees of t_2.*

An example of such intersection is shown in Figure 7.4. With the notion of rooted tree intersection we can define the similarity operator of our pattern structure.

[17]The maximal trees from a set X are all trees of X that are not a rooted subtree of another tree in X.

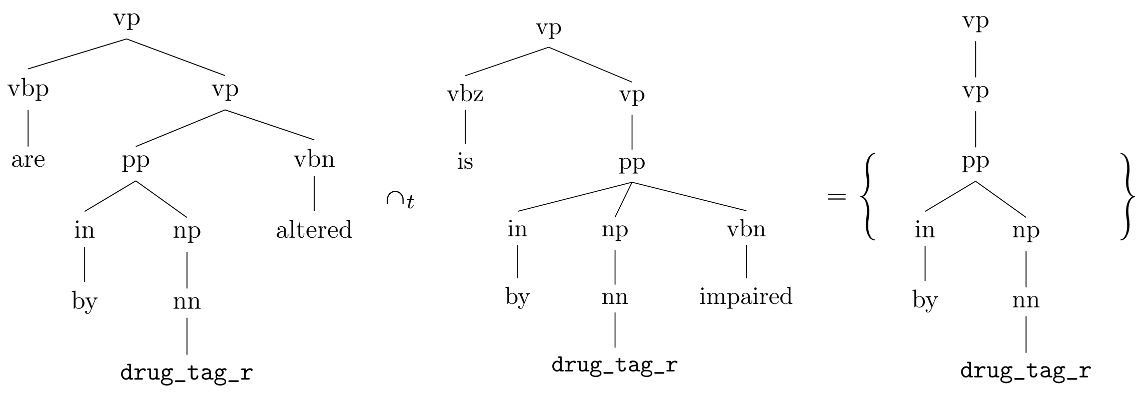


Figure 7.4: An example of rooted unordered tree intersection ($\cap_t$) of two syntactic tree fragments. The tree on the right side is the maximal rooted subtree of both trees on the left side.

Definition 7.4. *The* similarity *between a set of trees A and a set of trees B, written as $A \sqcap_t B$, is the subset of maximal trees from $\bigcup_{(a,b) \in A \times B} a \cap_t b$*

The corresponding subsumption operator is defined as mentioned previously $A \sqsubseteq_t B \Leftrightarrow A \sqcap_t B = A$. In fact, this definition corresponds to pattern structures of structured attributes by Ganter and Kuznetsov (2001). Indeed, the trees ordered by the subtree relation can be considered as a structured (ordered) set of attributes. And the corresponding similarity operation is exactly the one given by Definition 7.4.

7.3.3 The Projections for the Syntactic Tree Pattern Structure

Here we propose a projection that maps each tree description onto the set of its maximal branches. On the one hand, this projection does not loose a lot of information and, on the other hand, it significantly reduce the computational efforts.

Definition 7.5. *Rooted tree $t_1 = \langle N_1, E_1 \rangle$ is a* branch *of rooted tree $t_2 = \langle N_2, E_2 \rangle$ iff the following conditions hold: (a) $t_1 \subseteq_t t_2$ and (b) each node $n_1 \in N_1$ has at most one outgoing edge.*

Definition 7.6. *The branch projection of a set of rooted trees T, from now written as $\psi_b(T)$, is the set of maximal trees from $\bigcup_{t \in T} \{b \mid b \text{ is a branch of } t\}$*

Thus, a tree t defined by a root with n leaves will be projected to a set of size n, containing its branches (see Figure 7.5).

7.4 Classification based on Lazy Hypothesis Evaluation

In fact, the concept lattice resulting from the pattern structure which is defined above has not to be built. Instead, we follow a (kind of) supervised classification method for determining objects whose description includes a syntactic tree effectively representing a DDI, i.e., the drugs lying in the syntactic tree and marked with `drug_tag_r` tags are interacting. We follow a "Lazy Pattern Structure Classification" (LPSC) approach introduced by Kuznetsov (2013). LPSC can classify objects from a given pattern structure in polynomial time w.r.t the cardinality of the set of

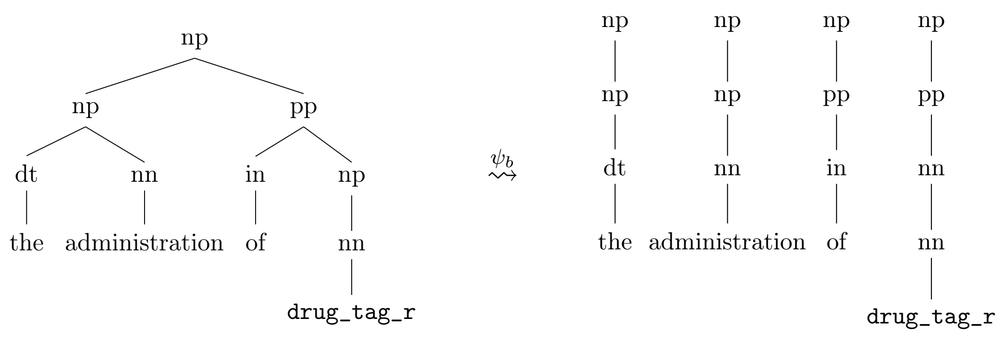


Figure 7.5: A tree and its maximal branches.

objects G considered as training data (modulo basic operation, e.g., description intersection and subsumption verification). It is based on a set of positive examples G_+ and a set of negative examples G_-. In the current experiment, positive examples are sentences including interacting drug-drug pairs while negative examples are sentences which do not include interacting drug-drug pairs.

Kuznetsov (2013) performs the classification of a new object o_n w.r.t. two questions:

(1.) Is there a "positive hypothesis" for o_n?

(2.) Is there a "negative hypothesis" for o_n?

A positive hypothesis is defined as a pattern intent in the pattern structure $(G_+, (D, \sqcap), \delta)$ that does not subsume any pattern from $\delta(G_-)$, i.e., does not subsume any negative example. A positive hypothesis for o_n is found iff: $\exists g_+ \in G_+ \ \forall g_- \in G_- : (o_n \sqcap g_+^\diamond) \nsubseteq g_-^\diamond$. In other words, a positive hypothesis for o_n is found if and only if o_n is similar to a positive example g_+, i.e., the potential positive hypothesis, and o_n does not share this similarity with any negative example g_-. A negative hypothesis for o_n is defined symmetrically, by switching the negative and positive examples. How an object is classified depends on the answers for the questions **(1.)** and **(2.)**, as shown in Table 7.1.

Table 7.1: Criteria in Lazy Pattern Structure Classification according to (Kuznetsov 2013) are displayed on the left, and criteria in LPSC restricted to only positive hypotheses evaluation –used in our experiments– are displayed on the right.

1.	**2.**	**Classification**
yes	yes	undefined
yes	no	positive
no	yes	negative
no	no	undefined

1.	**2.**	**Classification**
yes	yes	positive
yes	no	positive
no	yes	negative
no	no	negative

Our classification criteria differ from that in Kuznetsov (2013) as we are only looking for positive hypotheses and not for negative hypotheses. The underlying idea is that we assume that typical syntactic trees containing a DDI have some characteristic structures, while trees that do not contain any DDI do not have such characteristic structures. Thus, we discriminate positive and negative hypotheses w.r.t. the classification criteria. In our experiment, an object is classified as positive when the first question is answered with "yes", and by complementarity, an object is classified as negative when this first question is answered with "no". This kind of classification was exclusively used in our experiments and is termed as "*Lazy Positive Hypothesis Classification*" (LPHC) (see Table 7.1).

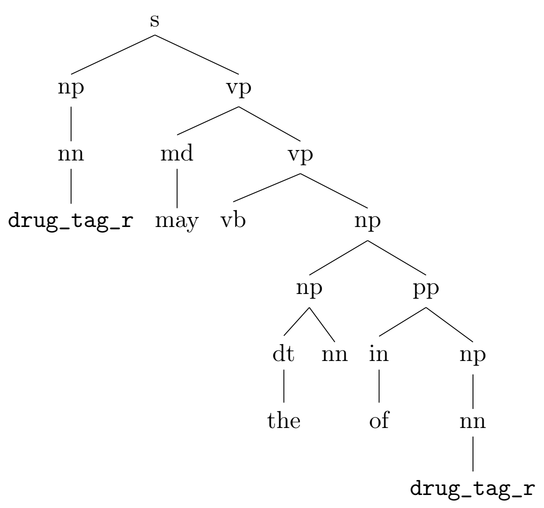


Figure 7.6: A positive pattern found in the experiments, created from the two sentences in Example 7.4 and Example 7.5. It should be noticed that, for the sake of clarity, this pattern is represented as a tree respecting the word ordering, but actually it is an unordered set of branches.

An example of a positive hypothesis that was found in the experiments with LPHC is shown in Figure 7.6. This positive hypothesis was created when classifying the tree corresponding to the potential DDI described in Example 7.4. Moreover, the positive example from the training set is the tree corresponding to Example 7.5.

Example 7.4. *Antihistamines (drug_tag_r) may partially counteract the anticoagulation effects of heparin (drug_tag_r) or warfarin (drug_tag).*

Example 7.5. *Tricyclic_ antidepressants (drug_tag_r) may block the antihypertensive (drug_tag) action of guanethidine (drug_tag_r) and similarly acting compounds.*

The tree in Figure 7.6 materializes the similarity between Example 7.4 and Example 7.5, and is not subsumed by any negative example in the training data. For this reason it is classified as a positive hypothesis for Example 7.4.

7.5 The Simplification of Syntactic Trees

When we looked manually at the sentences in the dataset, we remarked that not all parts of some sentences seem to contain useful information about the described DDIs. When a syntactic tree is large, it often takes more time to compute similarity with other trees. Therefore, it is interesting to remove parts of the sentence that are not required to find a DDI. Accordingly, we introduce "tree simplification operations" which are described below.

Constituent simplification.

By means of manually checking the trees, we noticed that some of the constituents are not very informative for describing a DDI in a sentence. In Example 7.6, it can be seen that an interaction is described between two `drug_tag_r` tags.

Example 7.6. *In diabetic patients, the metabolic effects of* ***drug_ tag_r*** *may decrease blood glucose and therefore* ***drug_ tag_r*** *requirements.*

However, it can be seen in Example 7.7 that some parts of the sentence can be removed without altering the description of the interaction.

Example 7.7. *The effects of* ***drug_ tag_r*** *may decrease blood glucose and* ***drug_ tag_r*** *requirements*

Usually, we can remove the constituents when the tree corresponding to the constituent does not contain any of the possibly interacting drugs, i.e., any of the two `drug_tag_r` nodes.

The candidate constituent to be removed that we considered are: (i) adjectives (JJ), (ii) prepositional phrases (PP), (iii) declarative clauses and clauses introduced by a subordinate conjunction such as relative clauses (S, SBAR), (v) adverbal phrases (ADVP) and (vi) parenthetical expressions (PRN). Subtrees of all these six categories that do not contain any of the `drug_tag_r` nodes are removed from the initial tree. The simplification of the tree corresponding to Example 7.6 is given in Figure 7.7.

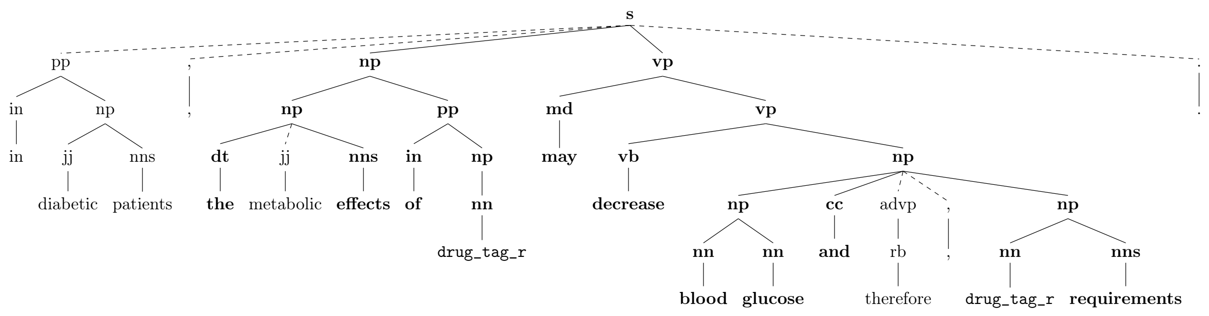


Figure 7.7: The original syntactic tree associated with the sentence "In diabetic patients, the metabolic effects of `drug_tag_r` may decrease blood glucose and therefore `drug_tag_r` requirements." The subtrees that will fall off after simplification are indicated with dashed lines.

NEGVP renaming.

To deal on a simple level with negation, each VP-node, i.e., representing a verb phrase, that directly contains a negating expression (not/no) is renamed as a NEGVP node. In this way a normal VP will not be matched with, or considered similar to, a negated VP.

Lowest-S simplification.

Because relations can sometimes be described very deep in a subordinate clause, only the deepest S-node (i.e., declarative clause) containing both `drug_tag_r` tags is considered, as shown in Figure 7.8. This makes sure that deeply nested interaction descriptions can be compared in an easier way to surface interaction descriptions. This way the lowest-S constituent in Example 7.8 (i.e., in the inner brackets) can be compared to the sentence in Example 7.9.

Example 7.8. *[$_S$* ***drug_ tag:*** *Clinical studies, as well as post marketing observations, have shown that [$_S$* ***drug_ tag_r*** *can reduce the* ***drug_ tag*** *effect of* ***drug_ tag_r*** *and* ***drug_ tag*** *in some patients].]*

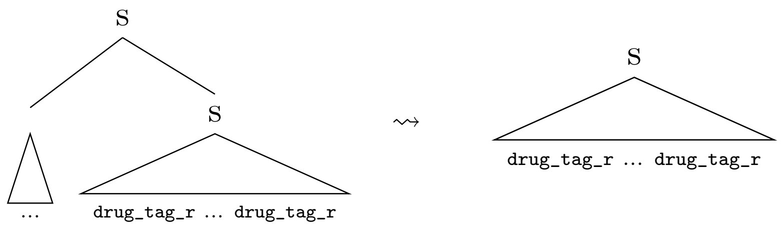


Figure 7.8: Schematic view of lowest-S simplification.

Example 7.9. *[$_S$ **drug_tag_r** agents reduce the renal clearance of **drug_tag_r** and add a high risk of **drug_tag** toxicity.]*

However, this rule does not always preserve all crucial information about the potential DDI. In some cases important information can be described at a meta level.

Example 7.10. *[$_S$ It is not known if [$_S$ **drug_tag_r** differ in their effectiveness when used with **drug_tag_r**].]*

In Example 7.10, both `drug_tag_r` tags occur in the S-constituent indicated by the inner brackets. Thus, when using lowest-S simplification, only the expression in the inner brackets is considered. However, the expression outside of the brackets, i.e., "It is not known if..." contains important information about the DDI description inside. It weakens or even nullifies the interaction that is described inside. For now, we do not have any clear solution to deal with such cases and we ignored them.

Link contraction.

After applying the constituent simplification operation, a resulting tree might contain branches that link nodes holding the same label with only one child. Such cases can be considered as redundant and can be simplified by removing the redundant non-branching duplicate nodes and linking the contracted new node with its single child node. An example is given in Figure 7.9.

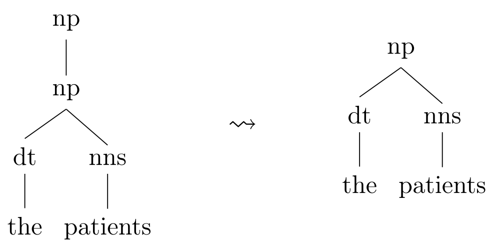


Figure 7.9: A tree and its contracted version.

If we apply all these tree simplifications on the trees obtained after parsing the experiment dataset, the average number of nodes in each tree drops from 130 to 41 and the maximum number of nodes from 311 to 138. This shows that the application of these simplification operations have a substantial impact on the set of resulting syntactic trees.

Table 7.2: Results from 10-fold cross validation on the DDI 2011 data set. Performance is measured in precision (P), recall (R) and F$_1$-measure (F$_1$). In all conditions constituent simplification is applied.

Simplifications	**P**	**R**	**F$_1$**
1. NEGVP, lowest-S, contraction	0.29786	0.48900	0.37022
2. NEGVP, contraction	0.32261	0.39044	0.35330
3. lowest-S, contraction	0.27073	**0.49450**	0.34990
4. NEGVP, lowest-S	0.33598	0.44712	0.38367
5. NEGVP, lowest-S, vp-map	0.35216	0.44585	0.39350
6. NEGVP, lowest-S, vp-map, prep-map	**0.38556**	0.41328	**0.39894**

7.6 Experiments and Discussion

7.6.1 The Experiment

In this experiment, different settings were evaluated. Each system classifies the potential DDIs by means of lazy pattern structure classification (actually positive hypothesis classification or LPHC). The underlying pattern structure is the one which is described in Section 7.3, using the branch projection. The settings are differing only in the tree simplifications that were applied.

For each setting, a 10-fold cross validation was performed on the data set. The corpus that is used is the training corpus of the DDI extraction challenge 2011 (Segura-Bedmar et al. 2011). In this corpus, the drugs are annotated and the interactions are build using the DrugBank, and then manually checked by a domain specialist.

We ran the experiments on a laptop with an i7 Intel processor (using 4 of its 8 virtual cores). The algorithm was implemented in Python. On average, each object classification took around 2 seconds. This long duration is primarily due to the search for positive hypotheses for each classification. It could be also possible to extract these positive hypotheses on a training set offline. Then they could be used as features in a different classification paradigm, maybe more optimized for a particular task. Here we did not do this as we were mostly interested in increasing the quality of the patterns.

The results from six settings we tested in the experiment are shown in Table 7.2. When we look at condition 1 and 2 in Table 7.2, we can see that applying lowest-S simplification strongly increases the recall, by 9.9%, but also reduces precision by 2.5%. Overall, F$_1$ increased by 1.69%. The reduction in precision, is probably due to some cases where the interaction is not fully described in the lowest declarative clause (lowest S-node). The increase in recall is probably due to the fact that surface clauses can now be compared better to deeper ones.

Applying the link contraction seems to have a weaker but similar effect. However, it decreases the F$_1$-measure. This can be noticed if we compare setting 1 and 4. After applying link contraction, the precision reduces with 3.8%, while the recall increases with 4,2% and the F$_1$-measure decreases with 1.4%. It appears that even if trees are non-branching, the hierarchy and its depth are important. Furthermore, if we compare setting 1 and 3, we can see that the NEGVP renaming has a positive effect on precision and only a minor negative effect on recall. It increases the F$_1$-measure with 2%.

Settings 5 and 6 are discussed below, in the error analysis.

7.6.2 Error Analysis

We manually looked both at false positives (i.e., negative trees classified as positive) and false negatives (i.e., positive trees classified as negative). False positives can be analyzed very precisely, because for each positively classified tree, the positive hypothesis from the positive training examples can be examined as well. A few non-mutually exclusive error categories that we found are the following.

1. *Insufficient similarity:* Sometimes, the similarity between the to be classified drug-drug pair and the positive hypothesis is too small to make a proper classification. This can be due to data sparseness or lack of information in the trees. Often in these cases the similarity between the to be classified tree and its corresponding positive hypothesis does not even contain a verb phrase node. Another frequent case is that the prepositions in the to be classified tree and its positive hypothesis do not match. An example of such poor similarity is given in Figure 7.10.

2. *Non-sentences:* Some mistakes seem to occur in phrases that are not full sentences or that are not parsed as such. Often the parser considers these phrases as noun phrases or as “fragments” (i.e., the root node is NP or FRAG). A reason for errors to occur in this category can be that there is not enough training data for these cases, or the parser made a mistake. Again the pattern in Figure 7.10 is an example of a non-sentence (an NP).

3. *Mistakes in annotation:* In some cases, a misclassification is due to errors in the drug annotations or in the interaction annotations. Examples of such cases can be found in P. Thomas et al. (2011).

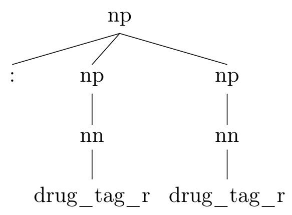


Figure 7.10: An example for error category 1 and 2. This pattern is clearly not sufficient for classification. This is due to the lack of a negative example in the training data that subsumes this pattern.

It can be noticed that some patterns may cause false positives, but can at the same time be responsible for a lot of true positives. In our experiments, we did not do any filtering directly based on performance. When the interest is in pure performance, it could be interesting to filter patterns that do not cause any true positives or those that cause more false positives then true positives.

7.6.3 Similarity Mappings

In error category 1, the similarity between the to be classified tree and its positive hypothesis was too small to make a proper classification. To prevent insufficient similarity, one could manually introduce some linguistically based constraints on the hypotheses and exclude hypotheses that do not satisfy them. We do this by mapping outputs of the similarity operator that do not fulfill the constraints to the empty set, and therefore have no potential for being a hypotheses. Based on the found errors, we introduce two types of similarity mappings: (i) *VP-mapping*, which maps outputs of the similarity operator that do not contain either a VP-node or a NEGVP-node to the empty set, (ii) *Prep-mapping*, which maps outputs of the similarity operator that do contain a prepositional phrase (PP-node) but not the exact preposition to the empty set.

Their result on performances can be found in Table 7.2. When we compare settings 4 and 5, the "vp-mapping" seems to have a small positive effect on precision (+ 1.6%), and hardly any effect on recall. When we compare settings 5 and 6, the "prep-mapping" also seems to have a positive effect on precision (+ 3.34%). However, the recall seems to decrease as well (- 3.3%). A reason for this could be that a side effect of the "prep-mapping" is that if two trees share a PP-node, but do not share the same preposition, this is considered the same as no PP-node match at all.

7.7 Conclusions and Future Work

In this chapter we presented a new way of analyzing drug-drug interactions in sentences based on pattern structures. We defined a pattern structure for syntactic trees and introduced a corresponding projection. Lazy pattern structure classification was also used to discover informative syntactic patterns, i.e., including DDIs. Furthermore we introduced a set of tree-simplification operations to reduce the size of the syntactic trees. The whole method was evaluated on the training corpus of the DDI extraction challenge 2011.

At present, it can be concluded that in terms of performance the system in its current state does not achieve very high performance. This is probably due to the rigid way the system deals with the found patterns. Furthermore, it should be noticed that this is a single system, using only phrase structure information.

However, from a qualitative point of view, many extracted syntactic patterns seem quite promising. For example, it would be interesting to use these extracted patterns as features in other classification paradigms and this could be included in future research. Another important direction could be to apply parse thickets (Galitsky, Ilvovsky, et al. 2014) for the task of DDI detection. A parse thicket is a graph built from the set of syntactic trees of a paragraph. This graph is enriched with the semantic links such as pronoun redirections. The work of Galitsky, Ilvovsky, et al. (2014) is based on pattern structures and, hence, can be adapted to our framework. Finally, other possible future research work could include the search for negative hypotheses, and to enrich the syntactic trees with semantic or morphological features.

The last (but not least) application of pattern structures is discussed in the next chapter where we deal with complex sequences formalizing the heterogeneity of hospitalization trajectories.

Chapter 8

On Mining Complex Sequential Data by means of Pattern Structures

Contents

8.1 Introduction

Sequence data is present and used in many applications. Mining sequential patterns from sequence data has become an important data mining task. In the last two decades, the main emphasis has been on developing efficient mining algorithms and effective pattern representations (Ding et al. 2009; Han, Pei, et al. 2000; Pei, Han, Mortazavi-Asl, H. Pinto, et al. 2001; Raïssi et al. 2008; Yan, Han, and Afshar 2003). However, one problem with traditional sequential pattern mining algorithms (and generally with all pattern enumeration algorithms) is that they generate a large number of frequent sequences while a few of them are truly relevant. To tackle this challenge, recent studies try to enumerate patterns using some alternative interestingness measures or by sampling representative patterns. A general idea in finding *statistically significant patterns* is to extract patterns whose characteristics for a given measure, such as frequency, strongly deviates from its expected value under a null model, i.e. the value expected by the distribution of all data. In this work, we focus on complementing the statistical approaches with a sound algebraic approach trying to answer the following question: *can we develop a framework for enumerating only relevant patterns based on data lattices and its associated measures?*

Table 8.1: Toy sequential data on patient medical trajectories.

Patient	Trajectory
p^1	$\langle [H_1, \{a\}]; [H_1, \{c, d\}]; [H_1, \{a, b\}]; [H_1, \{d\}] \rangle$
p^2	$\langle [H_2, \{c, d\}]; [H_3, \{b, d\}]; [H_3, \{a, d\}] \rangle$
p^3	$\langle [H_4, \{c, d\}]; [H_4, \{b\}]; [H_4, \{a\}]; [H_4, \{a, d\}] \rangle$

The above question can be answered by addressing the problem of analyzing sequential data using the framework of FCA and pattern structures (Ganter and Kuznetsov 2001). To analyze a dataset of "complex" sequences while avoiding the classical efficiency bottlenecks, we explain the usage of projections of pattern structures. Projections for sequences allow one to reduce the computational costs and the volume of enumerated patterns, avoiding the infamous "pattern flooding".

In this chapter, we develop a novel, rigorous and efficient approach for working with sequential pattern structures.

This chapter is an extension of the work presented at CLA'14 conference (Buzmakov et al. 2013b) and is based on the accepted paper to International Journal of General Systems (Buzmakov et al. 2015a).

The rest of this chapter is organized as follows. The specification of pattern structures for the case of sequences is presented in Section 8.2. Section 8.3 describes projections of sequential pattern structures followed in Section 8.4 by the evaluation and experimentations.

8.2 Sequential pattern structures

8.2.1 An example of sequential data

Imagine that we have medical trajectories of patients, i.e. sequences of hospitalizations, where every hospitalization is described by a hospital name and a set of procedures. An example of sequential data on medical trajectories with three patients is given in Table 8.1. We have a set of procedures $P = \{a, b, c, d\}$, a set of hospital names $T_H = \{H_1, H_2, H_3, H_4, CL, CH, *\}$, where hospital names are hierarchically organized (by level of generality). H_1 and H_2 are central hospitals (CH), H_3 and H_4 are clinics (CL), and $*$ denotes the root of this hierarchy. The least common ancestor in this hierarchy is denoted by $h_1 \sqcap h_2$, for any $h_1, h_2 \in T_H$, i.e. $H_1 \sqcap H_2 = CH$. Every hospitalization is described by one hospital name and may contain several procedures. The procedure order in each hospitalization is not important in our case. For example, the first hospitalization $[H_2, \{c, d\}]$ for the second patient (p^2) was a stay in hospital H_2 and during this hospitalization the patient underwent procedures c and d. An important task is to find the "characteristic" sequences of procedures and associated hospitals in order to improve hospitalization planning, optimize clinical processes or detect anomalies.

We approach the search for characteristic sequences by finding the most stable concepts in the lattice corresponding to a sequential pattern structure. For the simplification of calculations, subsequences are considered without "gaps", i.e the order of non consequent elements is not taken into account. This is reasonable in this task because experts are interested in regular consecutive events in healthcare trajectories. A sequential pattern structure is a set of sequences and is based on the set of maximal common subsequences (without gaps) between two sequences. Next subsections define partial order on sequences and the corresponding pattern structures.

8.2.2 Partial order on complex sequences

A sequence is constituted of elements from an alphabet. The classical subsequence matching task requires no special properties of the alphabet. Several generalizations of the classical case were made by introducing a subsequence relation based on an itemset alphabet (Agrawal and Srikant 1995) or on a multidimensional and multilevel alphabet (Plantevit, Laurent, et al. 2010). Here, we generalize the previous cases, requiring for an alphabet to form a semilattice $(E, \sqcap_E)$[18]. Thanks to the formalism of pattern structures we are able to process in a unified way all types of sequential datasets with poset-shaped alphabet (it is mentioned above that any partial order can be transformed into a semilattice). However, some sequential data can have connections between elements, e.g. (Adda et al. 2010), and, thus, cannot be straightforwardly processed by our approach.

Definition 8.1. *Given a semilattice $(E, \sqcap_E)$, also called an alphabet, a sequence is an ordered list of elements from E. We denote it by $\langle e_1; e_2; \cdots; e_n \rangle$ where $e_i \in E$.*

In this alphabet semilattice $(E, \sqcap_E)$ there is a bottom element $\bot_E$ that can be matched with any other element. Formally, $\forall e \in E, \bot_E = \bot_E \sqcap_E e$. This element is required by the lattice structure, but provides no useful information. Thus, it should be excluded from sequences. The bottom element of E corresponds to the empty set in sequential mining (Agrawal and Srikant 1995), and the empty set is always ignored in this domain.

Definition 8.2. *A valid sequence $\langle e_1; \cdots; e_n \rangle$ is a sequence where $\forall i \in \{1, \cdots, n\} e_i \neq \bot_E$.*

Definition 8.3. *Given an alphabet $(E, \sqcap_E)$ and two sequences $t = \langle t_1; ...; t_k \rangle$ and $s = \langle s_1; ...; s_n \rangle$ based on E ($t_q, s_p \in E$), the sequence t is a subsequence of s, denoted $t \leqslant s$, iff $k \leqslant n$ and there exist $j_1, ..j_k$ such that $1 \leqslant j_1 < j_2 < ... < j_k \leqslant n$ and for all $i \in \{1, 2, ..., k\}$, $t_i \sqsubseteq_E s_{j_i}$, i.e. $t_i \sqcap_E s_{j_i} = t_i$.*

Example 8.1. *In the running example (Section 8.2.1), the alphabet is $E = T_H \times \wp(P)$ with the similarity operation $(h_1, P_1) \sqcap (h_2, P_2) = (h_1 \sqcap h_2, P_1 \cap P_2)$, where $h_1, h_2 \in T_H$ are hospitals and $P_1, P_2 \in \wp(P)$ are sets of procedures. Thus, the sequence $ss^1 = \langle [CH, \{c, d\}]; [H_1, \{b\}]; [*, \{d\}] \rangle$ is a subsequence of $p^1 = \langle [H_1, \{a\}]; [H_1, \{c, d\}]; [H_1, \{a, b\}]; [H_1, \{d\}] \rangle$ because if we set $j_i = i + 1$ (Definition 8.3) then $ss^1_1 \sqsubseteq p^1_{j_1}$ ('CH' is more general than H_1 and $\{c, d\} \subseteq \{c, d\}$), $ss^1_2 \sqsubseteq p^1_{j_2}$ (the same hospital and $\{b\} \subseteq \{b, a\}$) and $ss^1_3 \sqsubseteq p^1_{j_3}$ ('*' is more general than H_1 and $\{d\} \subseteq \{d\}$).*

With complex sequences and this kind of subsequence relation the computation can be hard. Thus, for the sake of simplification, only "contiguous" subsequences are considered, where only the order of consequent elements is taken into account, i.e. given j_1 in Definition 8.3, $j_i = j_{i-1} + 1$ for all $i \in \{2, 3, ..., k\}$. Since experts are interested in regular consecutive events in healthcare trajectories, such a restriction does make sens for our data. It helps to connect only related hospitalizations. The next section introduces pattern structures that are based on complex sequences with a general subsequence relation, while the experiments are provided for a "contiguous" subsequence relation.

8.2.3 Sequential meet-semilattice

Based on the previous definitions, we can define the sequential pattern structure used for representing and managing sequences. For that, we make an analogy with the pattern structures

[18] We should note that in this chapter we consider two semilattices, the first one is related to the characters of the alphabet, $(E, \sqcap_E)$, and the second one is related to pattern structures, $(D, \sqcap)$.

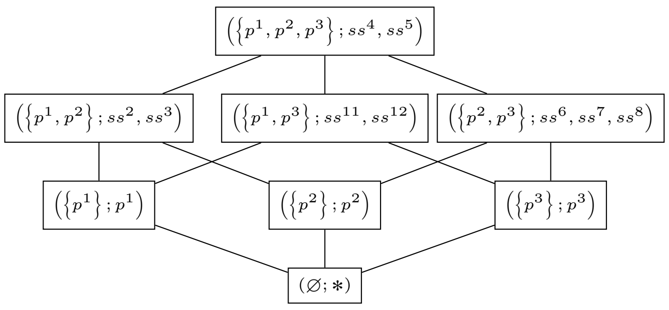

$(\{p^1, p^2, p^3\}; ss^4, ss^5)$
$(\{p^1, p^2\}; ss^2, ss^3)$
$(\{p^1, p^3\}; ss^{11}, ss^{12})$
$(\{p^2, p^3\}; ss^6, ss^7, ss^8)$
$(\{p^1\}; p^1)$
$(\{p^2\}; p^2)$
$(\{p^3\}; p^3)$
$(\varnothing; *)$

Figure 8.1: The concept lattice for the pattern structure given by Table 8.1. Concept intents reference to sequences in Tables 8.1 and 8.2.

for graphs (Kuznetsov 1999) where the meet-semilattice operation $\sqcap$ respects subgraph isomorphism. Thus, we introduce a sequential meet-semilattice respecting subsequence relation. Given an alphabet lattice $(E, \sqcap_E)$, $\mathfrak{S}$ is the set of all valid sequences based on $(E, \sqcap_E)$. $\mathfrak{S}$ is partially ordered w.r.t. Definition 8.3. $(D, \sqcap)$ is a semilattice on $\mathfrak{S}$, where $D \subseteq \wp(\mathfrak{S})$ such that, if $d \in D$ contains a sequence s, then all subsequences of s should be included into d, $\forall s \in d, \nexists \tilde{s} \leqslant s : \tilde{s} \notin d$, and the similarity operation is the set intersection for two sets of sequences. Given two patterns $d_1, d_2 \in D$, the set intersection operation ensures that if a sequence s belongs to $d_1 \sqcap d_2$ then any subsequence of s belongs to $d_1 \sqcap d_2$ and thus $d_1 \sqcap d_2 \in D$. As the set intersection operation is idempotent, commutative and associative, $(D, \sqcap)$ is a semilattice.

Example 8.2. *If pattern $d_1 \in D$ includes sequence $ss^4 = \langle[*, \{c, d\}]; [*, \{b\}]\rangle$ (see Table 8.2), then it should include also $\langle[*, \{d\}]; [*, \{b\}]\rangle$, $\langle[*, \{c, d\}]\rangle$, $\langle[*, \{d\}]\rangle$ and others. If pattern $d_2 \in D$ includes $ss^{12} = \langle[*, \{a\}]; [*, \{d\}]\rangle$, then it should include $\langle[*, \{a\}]\rangle$, $\langle[*, \{d\}]\rangle$ and $\langle\rangle$. Thus the intersection of two sets d_1 and d_2 is equal to the set $\{\langle[*, \{d\}]\rangle, \langle\rangle\}$.*

The next proposition stems from the aforementioned and will be used in the proofs in the next section.

Proposition 8.1. *Given $(G, (D, \sqcap), \delta)$ and $x, y \in D$, $x \sqsubseteq y$ if and only if $\forall s^x \in x$ there is a sequence $s^y \in y$, such that $s^x \leqslant s^y$.*

The set of all possible subsequences for a given sequence can be large. Thus, it is more efficient to consider a pattern $d \in D$ as a set of only maximal sequences $\tilde{d}$, $\tilde{d} = \{s \in d \mid \nexists s^* \in d : s^* \geqslant s\}$. Furthermore, every pattern will be given only by the set of all maximal sequences. For example, $\{p^2\} \sqcap \{p^3\} = \{ss^6, ss^7, ss^8\}$ (see Tables 8.1 and 8.2), i.e. $\{ss^6, ss^7, ss^8\}$ is the set of all maximal sequences specifying the intersection of p^2 and p^3. Similarly we have $\{ss^6, ss^7, ss^8\} \sqcap \{p^1\} = \{ss^4, ss^5\}$. Note that representing a pattern by the set of all maximal sequences allows for an efficient implementation of the intersection "$\sqcap$" of two patterns (in Section 8.4.1 we give more details on similarity operation w.r.t. a contiguous subsequence relation).

Example 8.3. *The sequential pattern structure for our example (Subsection 8.2.1) is $(G, (D, \sqcap), \delta)$, where $G = \{p^1, p^2, p^3\}$, $(D, \sqcap)$ is the semilattice of sequential descriptions, and δ is the mapping associating an object in G to a description in D shown in Table 8.1. Figure 8.1 shows the resulting lattice of sequential pattern concepts for this particular pattern structure $(G, (D, \sqcap), \delta)$.*

Table 8.2: Subsequences of patient sequences in Table 8.1.

	Subsequences
ss^1	$\langle [CH, \{c, d\}]; [H_1, \{b\}]; [*, \{d\}] \rangle$
ss^2	$\langle [CH, \{c, d\}]; [*, \{b\}]; [*, \{d\}] \rangle$
ss^3	$\langle [CH, \{\}]; [*, \{d\}]; [*, \{a\}] \rangle$
ss^4	$\langle [*, \{c, d\}]; [*, \{b\}] \rangle$
ss^5	$\langle [*, \{a\}] \rangle$
ss^6	$\langle [*, \{c, d\}]; [CL, \{b\}]; [CL, \{a\}] \rangle$
ss^7	$\langle [CL, \{d\}]; [CL, \{\}] \rangle$
ss^8	$\langle [CL, \{\}]; [CL, \{a, d\}] \rangle$
ss^9	$\langle [CH, \{c, d\}] \rangle$
ss^{10}	$\langle [CL, \{b\}]; [CL, \{a\}] \rangle$
ss^{11}	$\langle [*, \{c, d\}]; [*, \{b\}] \rangle$
ss^{12}	$\langle [*, \{a\}]; [*, \{d\}] \rangle$

8.3 Projections of sequential pattern structures

Pattern structures are hard to process due to the large number of concepts in the concept lattice, the complexity of the involved descriptions and the similarity operation. Moreover, a given pattern structure can produce a lattice with a lot of patterns which are not interesting for an expert. *Can we save computational time by avoiding to compute "useless" patterns?* Projections of pattern structures "simplify" to some degree the computation and allow one to work with a reduced description. Moreover, the stability measure of projected concepts never decreases w.r.t the original concepts.

Proposition 8.2. *Given a pattern structure $(G, (D, \sqcap), \delta)$, its concept c and a projected pattern structure $(G, (D_\psi, \sqcap_\psi), \psi \circ \delta)$, and the projected concept $\tilde{c}$, if the concept extents are equal ($\texttt{Ext}(c) = \texttt{Ext}(\tilde{c})$) then $\texttt{Stab}(c) \leqslant \texttt{Stab}(\tilde{c})$.*

Proof. Concepts c and $\tilde{c}$ have the same extent. Thus, according to Definition 2.1, in order to prove the proposition, it is enough to prove that for any subset $A \subseteq \texttt{Ext}(c)$, if $A^\diamond = \texttt{Int}(c)$ in the original pattern structure, then $A^\diamond = \texttt{Int}(\tilde{c})$ in the projected one.

Suppose that $\exists A \subset \texttt{Ext}(c)$ such that $A^\diamond = \texttt{Int}(c)$ in the original pattern structure and $A^\diamond \neq \texttt{Int}(\tilde{c})$ in the projected one. Then there is a descendant concept $\tilde{d}$ of $\tilde{c}$ in the projected pattern structure such that $A^\diamond = \texttt{Int}(\tilde{d})$ in the projected lattice. Then there is an original concept d for the projected concept $\tilde{d}$ with the same extent $\texttt{Ext}(d)$. Then $A^\diamond \sqsupseteq \texttt{Int}(d) \sqsupset \texttt{Int}(c)$ and, so, $A^\diamond$ cannot be equal to $\texttt{Int}(c)$ in the original lattice. Contradiction. □

Now we are going to present two projections of sequential pattern structures. The first projection comes from the following observation. In many cases it may be more interesting to analyze quite long subsequences rather than short ones. This kind of projections is called *Minimal Length Projection* (MLP) and it depends on the minimal length parameter ℓ for the sequences in a pattern. The corresponding function ψ maps a pattern without short sequences to itself, and a sequence with short sequences to the pattern containing only long sequences w.r.t. a given length threshold. Later, propositions 8.1 and 8.3 state that MLP is coherent with Definition 5.4.

Definition 8.4. *The function $\psi_{MLP} : D \to D$ of minimal length ℓ is defined as*

$$\psi_{MLP}(d) = \{s \in d \mid length(s) \geqslant \ell\}$$

Example 8.4. *If we prefer common subsequences of length $\ell \geqslant 3$, then between p^2 and p^3 in Table 8.1 there is only one maximal common subsequence, ss^6 in Table 8.2, while ss^7 and ss^8 are too short to be considered. Figure 8.2a shows the lattice of the projected pattern structure (Table 8.1) with patterns of length greater or equal to 3.*

Proposition 8.3. *The function ψ_{MLP} is a monotone, contractive and idempotent function on the semilattice $(D, \sqcap)$.*

Proof. The contractivity and idempotency are quite clear from the definition. It remains to prove the monotonicity.

If $X \sqsubseteq Y$, where X and Y are sets of sequences, then for every sequence $x \in X$ there is a sequence $y \in Y$ such that $x \leqslant y$ (Proposition 8.1). We should show that $\psi(X) \sqsubseteq \psi(Y)$, or in other words for every sequence $x \in \psi(X)$ there is a sequence $y \in \psi(Y)$, such that $x \leqslant y$. Given $x \in \psi(X)$, since $\psi(X)$ is a subset of X and $X \sqsubseteq Y$, there is a sequence $y \in Y$ such that $x \leqslant y$, with $|y| \geqslant |x| \geqslant \ell$ (ℓ is a parameter of MLP), and thus, $y \in \psi(Y)$. □

Another important type of projections is related to a variation of the lattice alphabet $(E, \sqcap_E)$. One possible variation of the alphabet is to ignore certain fields in the elements. For example, if a hospitalization is described by a hospital name and a set of procedures, then either hospital or procedures can be ignored in similarity computation. For that, in any element the set of procedures should be substituted by $\varnothing$, or the hospital by $*$ ("arbitrary hospital") which is the most general element of the taxonomy of hospitals.

Another variation of the alphabet is to require that some field(s) should not be empty. For example, we want to find patterns with non-empty set of procedures or the element $*$ of the hospital taxonomy is not allowed in elements of a sequence. Such variations are easy to realize within our approach. For this, when computing the similarity operation between elements of the alphabet, one should check if the result contains empty fields and, if yes, should substitute the result by $\perp$. This variation is useful, as it is shown in the experimental section, but is rather difficult to define within more classical frequent sequence mining approaches, which will be discussed later.

Example 8.5. *An expert is interested in finding sequential patterns describing how a patient changes hospitals, but with little interest in procedures. Thus, any element of the alphabet lattice, containing a hospital and a non-empty set of procedures can be projected to an element with the same hospital, but with an empty set of procedures.*

Example 8.6. *An expert is interested in finding sequential patterns containing some information about the hospital in every hospitalization, and the corresponding procedures, i.e. hospital field in the patterns cannot be equal to $*$, e.g., ss^5 is an invalid pattern, while ss^6 is a valid pattern in Table 8.2. Thus, any element of the alphabet semilattice with $*$ in the hospital field can be projected to the $\perp_E$. Figure 8.2b shows the lattice corresponding to the projected pattern structure (Table 8.1) defined by a projection of the alphabet semilattice.*

Below we formally define how the alphabet projection of a sequential pattern structure should be processed. Intuitively, every sequence in a pattern should be substituted with another sequence, by applying the alphabet projection to all its elements. However, the result can be an incorrect sequence, because $\perp_E$ cannot belong to a valid sequence. Thus, sequences in a pattern should be "developed" w.r.t. $\perp_E$, as it is explained below.

Definition 8.5. *Given an alphabet $(E, \sqcap_E)$, a projection of the alphabet ψ and a sequence $s = \langle s_1, \cdots, s_n \rangle$ based on E, the projection $\psi(s)$ is the sequence $\tilde{s} = \langle \tilde{s}_1, \cdots, \tilde{s}_n \rangle$, such that $\tilde{s}_i = \psi(s_i)$.*

Here, it should be noticed that $\tilde{s}$ is not necessarily a valid sequence (see Definition 8.2), since it can include $\perp_E$ as an element. However, in sequential pattern structures, elements should include only valid sequences (see Section 8.2.3).

Definition 8.6. *Given an alphabet $(E, \sqcap_E)$, a projection of the alphabet ψ_E, an alphabet projection for the sequential pattern structure $\psi(d)$ is the set of valid sequences smaller than the projected sequences from d:*

$$\psi(d) = \{s \in \mathfrak{S} | (\exists t \in d) s \leqslant \psi_E(t)\},$$

where $\mathfrak{S}$ is the set of all valid sequences based on $(E, \sqcap_E)$.

Example 8.7. *$\{ss^6\} = \{\langle [*, \{c, d\}]; [CL, \{b\}]; [CL, \{a\}] \rangle\}$ is an alphabet-projected pattern for the pattern $\{ss^{10}\} = \{\langle [CL, \{b\}]; [CL, \{a\}] \rangle\}$, where the alphabet lattice projection is given in Example 8.6.*

In the case of contiguous subsequences, $\{\langle [CH, \{c, d\}] \rangle\}$ is an alphabet-projected pattern for the pattern $\{ss^2\} = \{\langle [CH, \{c, d\}]; [*, \{b\}]; [*, \{d\}] \rangle\}$, where the alphabet lattice projection is given by projecting every element with medical procedure b to the element with the same hospital and with the same set of procedures excluding b. The projection of sequence ss^2 is $\langle [CH, \{c, d\}]; [*, \{\}]; [*, \{d\}] \rangle$, but $[*, \{\}] = \perp_E$, and, thus, in order to project the pattern $\{ss^2\}$ the projected sequence is substituted by its maximal subsequences, i.e. $\psi(\{\langle [CH, \{c, d\}]; [*, \{b\}]; [*, \{d\}] \rangle\}) = \{\langle [CH, \{c, d\}] \rangle\}$.

Proposition 8.4. *Considering an alphabet $(E, \sqcap_E)$, a projection of the alphabet ψ, a sequential pattern structure $(G, (D, \sqcap), \delta)$, the alphabet projection (see Definition 8.6) is monotone, contractive and idempotent.*

Proof. This projection is idempotent, since the projection of the alphabet is idempotent and only the projection of the alphabet can change the elements appearing in sequences.

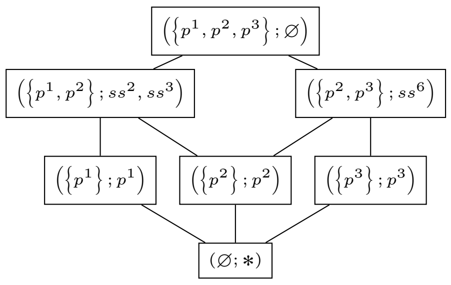
It is contractive because for any pattern $d \in D$ and any sequences $s \in d$, a projection of the sequence $\tilde{s} = \psi(s)$ is a subsequence of s. In Definition 8.6 the projected sequences should be substituted by their subsequences in order to avoid $\perp_E$, building the sets $\{\tilde{s}^i\}$. Thus, s is a supersequence for any $\tilde{s}^i$, and, so, the projected pattern $\tilde{d} = \psi(d)$ is subsumed by the pattern d.

Finally, we should show monotonicity. Given two patterns $x, y \in D$, such that $x \sqsubseteq y$, i.e. $\forall s^x \in x, \exists s^y \in y : s^x \leqslant s^y$, consider the projected sequence of s^x, $\psi(s^x)$. As $s^x \leqslant s^y$ for some s^y then for some $j_0 < \cdots < j_{|s^x|}$ (see Definition 8.3) $s^x_i \sqsubseteq_E s^y_{j_i}$ $(i \in 1, 2, ..., |s^x|)$, then $\psi(s^x_i) \sqsubseteq_E \psi(s^y_{j_i})$ (by the monotonicity of the alphabet projection), i.e. the projected sequence preserves the subsequence relation. Thus, the set of allowed subsequences of s^x is a subset of the set of allowed subsequences of s^y. Hence, the alphabet projection of the pattern preserves pattern subsumption relation, $\psi(x) \leqslant \psi(y)$ (Proposition 8.1), i.e. the alphabet projection is monotone. □

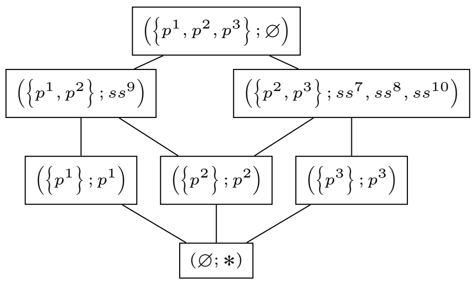
8.4 Sequential pattern structure evaluation

8.4.1 Implementation

Nearly any state-of-the-art FCA algorithm can be adapted to process pattern structures. We adapted the `AddIntent` algorithm (Merwe et al. 2004), as the lattice structure is important for us



(a) MLP projection, $l = 3$



(b) Projection removing '*' in the hospital field

Figure 8.2: The projected concept lattices for the pattern structure given by Table 8.1. Concept intents refer to the sequences in Tables 8.1 and 8.2.

to calculate stability (see the algorithm for calculating stability by Roth et al. (2008)). To adapt the algorithm to our needs, every set intersection operation on attributes is substituted with the semilattice operation $\sqcap$ on corresponding patterns, while every subset checking operation is substituted with the semilattice order checking $\sqsubseteq$, in particular all $(\cdot)'$ are substituted with $(\cdot)^\diamond$.

The next question is how the semilattice operation $\sqcap$ and subsumption relation $\sqsubseteq$ can be implemented for contiguous sequences. Given two sets of sequences $S = \{s^1, ...s^n\}$ and $T = \{t^1, ..., t^m\}$, the similarity of these sets $S \sqcap T$, is calculated according to Section 8.2.3, i.e. maximal sequences among all common subsequences for any pair of sequences s^i and t^j.

To find all common subsequences of two sequences, the following observations can be useful. If $ss = \langle ss_1; ...; ss_l\rangle$ is a subsequence of $s = \langle s_1; ...; s_n\rangle$ with $j_i^s = k^s + i$, i.e. $ss_i \sqsubseteq_E s_{k^s+i}$ (Definition 8.3: k^s is the index difference from which ss is a contiguous subsequence of s) and a subsequence of $t = \langle t_1; ...; t_m\rangle$ with $j_i^t = k^t + i$, i.e. $ss_i \sqsubseteq_E t_{k^t+i}$, then for any index $i \in \{1, 2, ..., l\}$, $ss_i \sqsubseteq_E (s_{j_i^s} \sqcap t_{j_i^t})$. Thus, to find all maximal common subsequences of s and t, we first align s and t in all possible ways. For each alignment of s and t we compute the resulting intersection. Finally, we keep only the maximal intersected subsequences.

For example, let us consider two possible alignments of s^1 and s^2:

$$\begin{array}{rl} s^1 = & \langle\{a\};\{c,d\};\{b,a\};\{d\}\ \rangle \\ s^2 = & \qquad\quad \langle\{c,d\};\{b,d\};\{a,d\}\rangle \\ ss^l = & \qquad\quad \langle\ \ \varnothing\ \ ;\{d\}\ \rangle \end{array} \qquad \begin{array}{rl} s^1 = & \langle\{a\};\{c,d\};\{b,a\};\{d\}\ \rangle \\ s^2 = & \qquad \langle\{c,d\};\{b,d\};\{a,d\}\rangle \\ ss^r = & \qquad \langle\{c,d\};\{b\};\{d\}\ \rangle \end{array}$$

The left intersection ss^l is not retained, as it is not maximal ($ss^l < ss^r$), while the right intersection ss^r is kept.

The complexity of the alignment for two sequences s and t is $O(|s| \cdot |t| \cdot \gamma)$, where γ is the complexity of computing a common ancestor in the alphabet lattice $(E, \sqcap)$.

8.4.2 Experiments and discussion

The experiments are carried out on a MacBook Pro with a 2.5GHz Intel Core i5, 8GB of RAM Memory running OS X 10.6.8. The algorithms are not parallelized and are coded in C++.

Our use-case dataset comes from a French healthcare system, called PMSI[19] (Fetter et al. 1980). Each element of a sequence has a "complex" nature. The dataset contains 500 patients suffering from *lung cancer*, who live in the Lorraine region (Eastern France). Every patient is described as a sequence of hospitalizations without any time-stamp. A hospitalization is a tuple with three elements: (i) healthcare institution (e.g. university hospital of Nancy (CHU_{Nancy})),

[19]Programme de Médicalisation des Sytèmes d'Information.

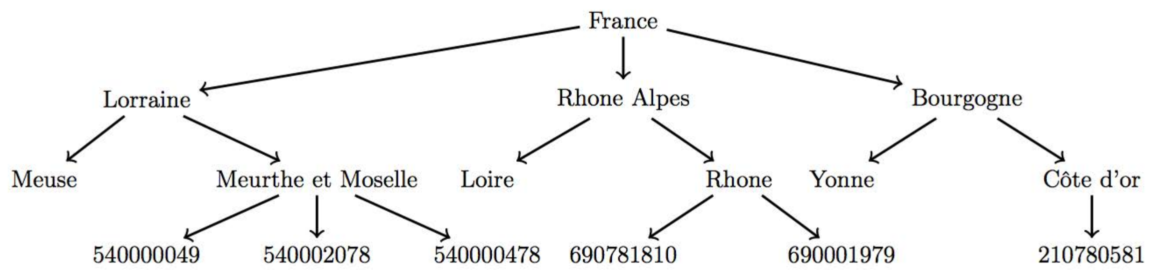


Figure 8.3: A geographical taxonomy of the healthcare institution

(ii) reason for the hospitalization (e.g. a cancer disease), and (iii) set of medical procedures that the patient undergoes. An example of a medical trajectory is given below:

$$\langle[\mathrm{CHU}_{Nancy}, \mathrm{Cancer}, \{mp_1, mp_2\}] \ ; \ [\mathrm{CH}_{Paris}, \mathrm{Chemo}, \{\}] \quad ; [\mathrm{CH}_{Paris}, \mathrm{Chemo}, \{\}]\rangle \, .$$

This sequence represents a patient trajectory with three hospitalizations. It expresses that the patient was first admitted to the university hospital of Nancy (CHU_{Nancy}) for a cancer problem as a reason, and underwent procedures mp_1 and mp_2. Then he had two consequent hospitalizations in the general hospital of Paris (CH_{Paris}) for chemotherapy with no additional procedure. Substituting the same consequent hospitalizations by the number of repetitions, we have a shorter and more understandable trajectory. For example, the above pattern is transformed into two hospitalizations where the first hospitalization repeats once and the second twice:

$$\langle[\mathrm{CHU}_{Nancy}, \mathrm{Cancer}, \{mp_1, mp_2\}] \times [1]; [\mathrm{CH}_{Paris}, \mathrm{Chemo}, \{\}] \times [2]\rangle \, .$$

Diagnoses are coded according to the 10^{th} International Classification of Diseases (ICD10). Based on this coding, diagnoses could be described at 5 levels of granularity: root, chapter, block, 3-character, 4-character, terminal nodes. This taxonomy has 1544 nodes. The healthcare institution is associated with a geographical taxonomy of 4 levels, where the first level refers to the root (France) and the second, the third and the fourth levels correspond to administrative region, administrative department and hospital respectively. Figure 8.3 presents University Hospital of Nancy (code: 540002078) as a hospital in Meurthe et Moselle, which is a department in Lorraine, region of France. This taxonomy has 304 nodes. The *medical procedures* are coded according to the French nomenclature "Classification Commune des Actes Médicaux (CCAM)". The distribution of sequence lengths is shown in Figure 8.4.

With 500 patient trajectories, the computation of the whole lattice is infeasible. We are not interested in all possible frequent trajectories, but rather in trajectories which answer medical analysis questions. An expert may know the minimal size of trajectories that he is interested in, i.e. setting the MLP projection. We use the MLP projection of length 2 and 3 and take into account that most of the patients has at least 2 hospitalizations in the trajectory (see Figure 8.4).

Figure 8.5 shows computational times for different projections as a function of dataset size. Figure 8.5a shows different alphabet projections for MLP projection with $\ell = 2$, while Figure 8.5b for MLP with $\ell = 3$. Every alphabet projection is given by the name of fields, that are considered within the projection: `G` corresponds to hospital geo-location, `R` is the reason for a

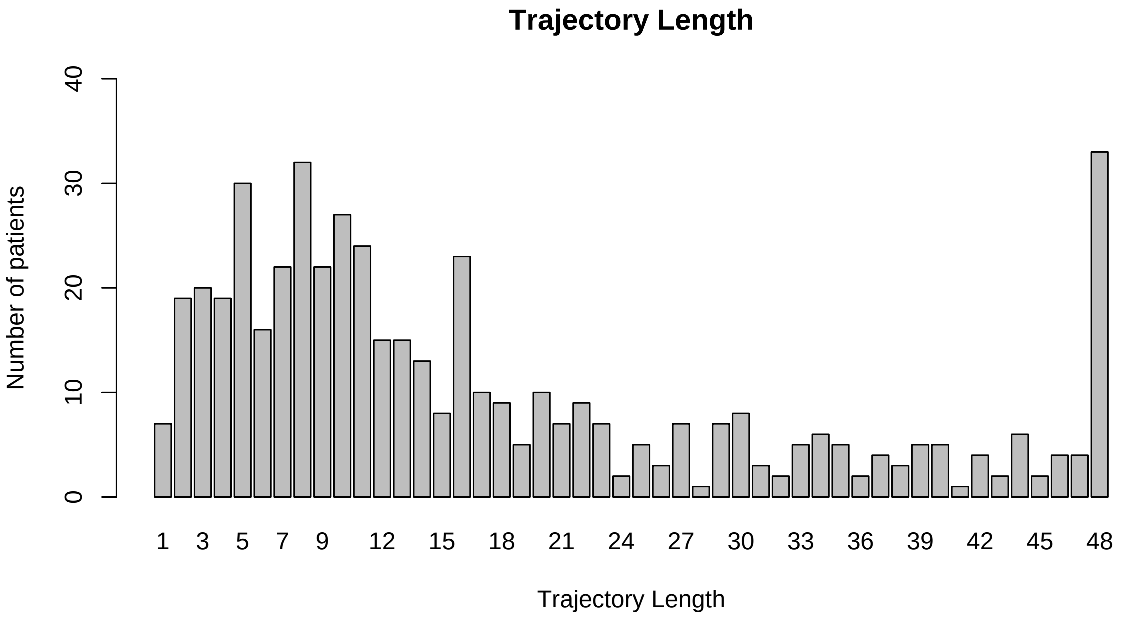
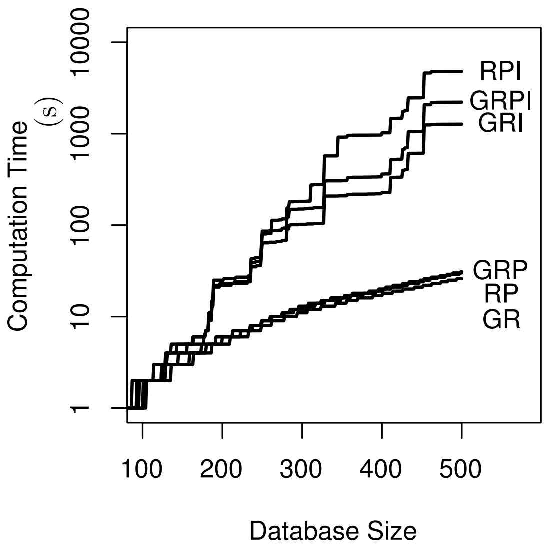
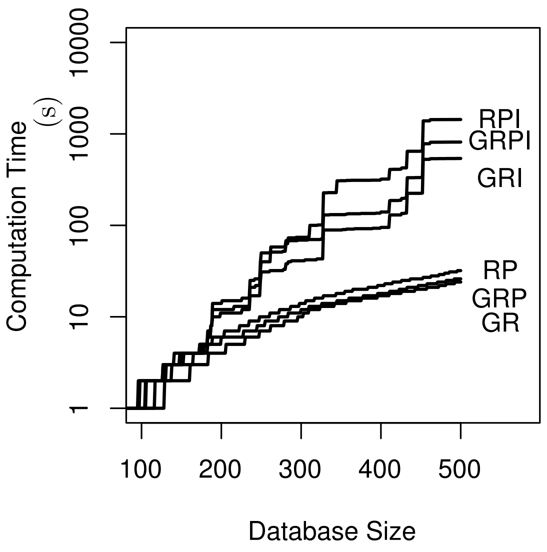


Figure 8.4: The length distribution of sequences in the dataset

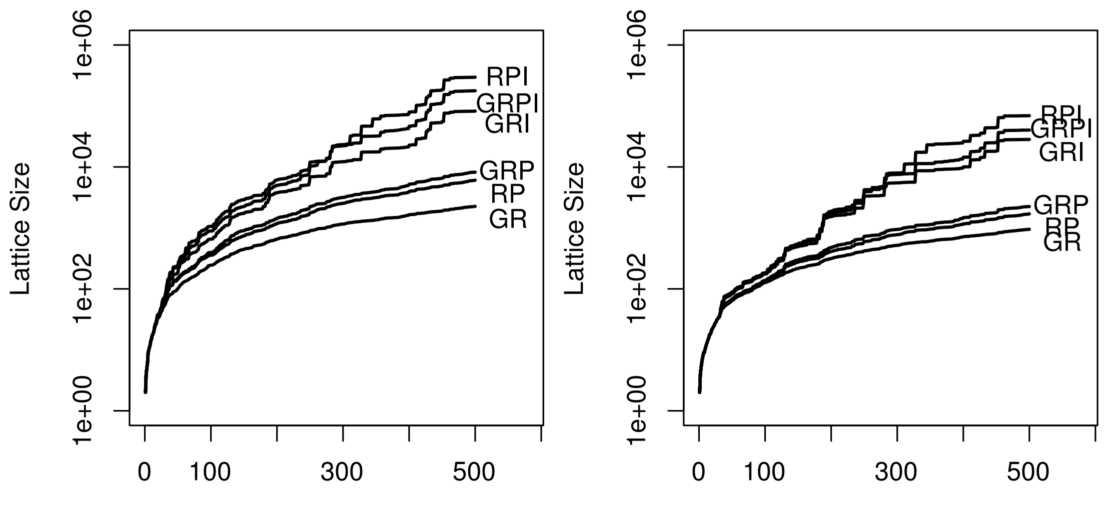


(a) MLP projection, $\ell = 2$



(b) MLP projection, $\ell = 3$

Figure 8.5: Computational time for different projections



(a) MLP projection, $\ell = 2$ (b) MLP projection, $\ell = 3$

Figure 8.6: Lattice size for different projections

Table 8.3: Interesting concepts, for different projections.

#	Projection	Intent	Stab. Rank	Support
1	GR	$\langle[Lorraine, C341\ Lung\ Cancer]\rangle$	1	287
2	$GR2$	$\langle[Lorraine, Respiratory\ Disease]; [CHU_{Nancy}, Lung\ Cancer]\rangle$	26	22
3	$GR3$	$\langle[Lorraine, Chemotherapy] \times 4\rangle$	1	176
4	$RPI3$	$\langle[Preparation\ for\ Chemotherapy, \{Lung\ \ Radiography\}];\ [Chemotherapy] \times [3,4]\rangle$	5	36

hospitalization, `P` is medical procedures and `I` is repetition interval, i.e. the number of consequent hospitalizations with the same reason. We can see from these figures that MLP allows one to save some computational resources with increasing of ℓ. The difference in computational time between $\ell = 2$ and $\ell = 3$ projections is significant, especially for time consuming cases. Even a bigger variation can be noticed for the alphabet projections. For example, computation of the `RPI` projection takes 100 times more resources than any from `GRP, RP, GR, GRP`.

The same dependency can be seen in Figure 8.6, where the number of concepts for every projection is shown. Consequently, it is important for an expert to provide a strict projection that allows him to answer his questions in order to save computational time and memory.

Table 8.3 shows some interesting concept intents with the corresponding support and ranking w.r.t. concept stability. For example the concept #1 is obtained under the projection GR (i.e., we consider only hospital and reason), with the intent $\langle[Lorraine, C341\ Lung\ Cancer]\rangle$, where `C341 Lung Cancer` is a special kind of lung cancer (malignant neoplasm in Upper lobe, bronchus or lung). This concept is the most stable concept in the lattice for the given projection, and the size of the concept extent is 287 patients.

One of the questions that the analyst would like to address here is *"Where do patients stay (i.e. hospital location) during their treatment, and for which reason ?"*. To answer this question, we consider only healthcare institutions and reason fields, requiring both to "hold" some information and we use the MLP projection of length 2 and 3 (i.e. projections $GR2$ and $GR3$). Nearly all frequent trajectories show that patients usually are treated in the same region. However, *pattern #2* obtained under $GR2$ projection shows that, *"22 patients were first admitted in some healthcare* *institution in Lorraine region for a problem related to the respiratory system and then they were treated for a lung cancer in University Hospital of Nancy."*

Another interesting question is *"What are the sequential relations between hospitalization reasons and the corresponding procedures?"*. To answer this question, we are not interested in healthcare institutions. Thus, any alphabet element is projected by substituting healthcare institution field with '*'. As hospitalization reason is important in each hospitalization, any alphabet element without the hospitalization reason is of no use and is projected to the bottom element $\bot_E$ of the alphabet. Such projections are called $RPI2$ or $RPI3$, meaning that we consider the fields "Reason" and "Procedures", while the reason should not be empty and the MLP parameter is 2 or 3. *Pattern #4* trivially states that, *"36 patients with lung cancer are hospitalized once for the preparation of chemotherapy and during this hospitalization they undergo lung radiography. Afterwards, they are hospitalized between 3 and 4 times for chemotherapy."*

Variability is high in healthcare processes and affects many aspects of healthcare trajectories: patients, medical habits and protocols, healthcare organisation, availability of treatments and settings... Mining sequential pattern structures is an interesting approach for finding regularities across one or several dimensions of medical trajectories in a population of patients. It is flexible enough to help healthcare managers to answer specific questions regarding the natural organisation of care processes and to further compare them with expected or desirable processes. The use of taxonomies plays also a key role in finding the right level of description of sequential patterns and reducing the interpretation overhead.

8.5 Related work

Agrawal and Srikant (1995) introduced the problem of mining sequential patterns over large sequential databases. Formally, given a set of sequences, where each sequence is a list of transactions ordered by time and each transaction is a set of items, the problem amounts to find all frequent subsequences that appear a sufficient number of times with a user-specified minimum support threshold (*minsup*). Following the work of Agrawal and Srikant many studies have contributed to the efficient mining of sequential patterns (Mooney and Roddick 2013). Most of them are based on the antimonotonicity property (used in *Apriori*), which states that any super pattern of a non-frequent pattern cannot be frequent. The main algorithms are PrefixSpan (Pei, Han, Mortazavi-Asl, Helen Pinto, et al. 2001), SPADE (Zaki 2001), SPAM (Ayres et al. 2002), PSP (Masseglia et al. 1998a), DISC (Chiu et al. 2004), PAID (Yang et al. 2006) and FAST (Salvemini et al. 2011). All these algorithms aim at discovering sequential patterns from a set of sequences of itemsets such as customers who frequently buy DVDs of episodes I, II and III of Stars Wars, then buy within 6 months episodes IV, V, VI of the same famous epic space opera.

Many studies about sequential pattern discovery focus on single-dimensional sequences. However, in many situations, the database is multidimensional in the sense that items can be of different nature. For example, a consumer database can hold information such as article price, gender of the customer, location of the store and so on. Helen Pinto et al. (2001) proposed the first work for mining multidimensional sequential patterns. In this work, a *multidimensional sequential database* is defined as a schema $(ID, D_1, ..., D_m, S)$, where ID is a unique customer identifier, $D_1, ..., D_m$ are dimensions describing the data and S is the sequence of itemsets. A *multidimensional sequence* is defined as a vector $\langle\{d_1, d_2, ..., d_m\}, S_1, S_2, ..., S_l\rangle$ where $d_i \in D_i$ for $(i \leqslant m)$ and $S_1, S_2, ..., S_l$, are the itemsets of sequence S. For instance, $\langle\{Metz, Male\}, \{mp_1, mp_2\}, \{mp_3\}\rangle$ describes a male patient who underwent procedures mp_1 and mp_2 in Metz and then underwent mp_3 also in Metz. Here, dimensions remain constant over time, such as the location of the treatment. This means that it is not possible to have a pattern indicating that when the patient underwent procedures mp_1 and mp_2 in Metz then he underwent mp_3 in Nancy. Among other proposals, C.-C. Yu and Chen (2005) proposed two methods AprioriMD and PrefixMDSpan for mining multidimensional sequential patterns in the web domain. This study considers pages, sessions and days as dimensions. Actually, these three different dimensions can be projected into a single dimension corresponding to web pages, gathering web pages visited during a same session and ordering sessions w.r.t the day as order.

In real world applications, each dimension can be represented at different levels of granularity, by using a poset. For example, apples in a market basket analysis can be either described as fruits, fresh food or food. The interest lies in the capacity of extracting more or less general/specific multidimensional sequential patterns and overcome problems of excessive granularity and low support. Srikant and Agrawal (1996) proposed GSP which uses posets for extracting sequential patterns. The basic approach is based on replacing every item with all the ancestors in the poset and then the frequent sequences are generated. This approach is not scalable in a multidimensional context because the size of the database becomes the product of maximum height of the posets and number of dimensions.

Plantevit, Laurent, et al. (2010) defined a *multidimensional sequence* as an ordered list of multidimensional items, where a *multidimensional item* is a tuple $(d_1, ..., d_m)$ and d_i is an item associated with the i^{th} dimension. They proposed M^3SP, an approach taking both aspects into account where each dimension is represented at different levels of granularity, by using a poset. M^3SP is able to search for sequential patterns with the most appropriate level of granularity. Their approach is based on the extraction of the most specific frequent multidimensional items, which are then used as alphabet to rephrase the original database. Then, M^3SP uses a standard sequential pattern mining algorithm to extract multidimensional sequential patterns. However, M^3SP is not adapted to mine sequential databases, where sequences are defined over a combination of sets of items and items lying in a poset. Then it is not possible to have a pattern indicating that when the patient went to uh_p for a problem of cancer ca, where he underwent procedures mp_1 and mp_2, then he went to gh_l for the same medical problem ca, where he underwent mp_3 (i.e, $\langle(uh_p, ca, \{mp_1, mp_2\}), (gh_l, ca, \{mp_3\})\rangle$). Our approach allows us to process such kind of patterns and in addition the elements of sequences are even more general. For example, beside multidimensional and multilevel sequences, sequences of graphs fall under our definition. Moreover, frequent subsequence mining gives rise to a lot of subsequences which can be hardly analyzed by an expert. Since our approach is based on FCA, we can use efficient relevance indices defined in FCA.

The approach introduced in this chapter is not the first attempt to use FCA for the analysis of sequential data. Ferré (2007) processes sequential datasets based on a “simple” alphabet without involving any partial order. In Casas-Garriga (2005) only sequences of itemsets are considered. All closed subsequences are firstly mined and then regrouped by a specialized algorithm in order to obtain a lattice similar to the FCA lattice. This approach was not verified experimentally. Moreover, compared with both approaches, i.e., Ferré (2007) and Casas-Garriga (2005), our approach suggests a more general definition of sequences and, thanks to pattern structures, there is no ‘pre-mining’ step to find frequent (or maximal) subsequences. This allows us to apply different “projections” specializing the request of an expert and simplifying the computations. In addition, in our approach nearly all state-of-the-art FCA algorithms can be used in order to efficiently process a dataset. Garriga et al. (2012) process multirelational databases by extending LCM (Uno, Asai, et al. 2004), which is closely related to FCA. Although this approach is efficient for special kinds of multirelational databases, it cannot process sequential and graph datasets for reasons explained in their paper.

There is a number of approaches that help to analyze medical treatment data. However, the direct comparison of them is hardly possible, because every approach is designed for its own problem. For example, Tsumoto et al. (2014) analyze data of one hospital and provide a different view on the processes within the hospital w.r.t. our approach. Finally, the most similar approach to work described here can be found in Egho, Jay, et al. (2014) and Egho, Raïssi, et al. (2014). They mine frequent sequences of the dataset similar to the sequences studied here. However, they approach the complexity of the analysis of such data in a different way. They use a support threshold in order to specify the outcome of the algorithm and do not provide any order in which one can analyze the result. In our case we rely on projections that are usually simpler to incorporate expert knowledge than a support threshold and we give an order (w.r.t. stability of a concept) which can be used to simplify the analysis of the treatment data.

8.6 Conclusion

In this chapter, we have presented an approach for analyzing sequential data within the framework of pattern structures. Correspondingly, we define sequential pattern structures and their projections. Our work complements the general orientations towards *statistically significant patterns* by presenting strong formal results on the notion of interestingness from a concept lattice viewpoint. The framework of pattern structures is very flexible and shows some important properties, for example in allowing to reuse state-of-the-art and efficient FCA algorithms. Using pattern structures leads to the construction of a pattern concept lattice, which does not require the setting of a support threshold, as usually needed in classical sequential pattern mining. Moreover, the use of projections gives a lot of flexibility especially for mining and interpreting special kinds of patterns (patterns can be proposed at several levels of complexity w.r.t. extraction and interpretation).

Our framework was tested on a real-world dataset with patient hospitalization trajectories. Interesting patterns answering questions of an expert are extracted and interpreted, showing the feasibility and usefulness of the approach, and the importance of the stability as a pattern-selection procedure. In particular, projections play an important role here: mainly, they provide means to select patterns of a special interest and they help to save computational time (which could be otherwise very large).

In general pattern structures allow efficient dealing with many types of data. However the number of patterns can be significant. Projections allow one to reduce the complexity of computations. However, as we have seen in this chapter, even in a projected pattern structure there could exist a large numeber of concepts. In order to deal with this problem we have applied stability, an interestingness measure of a concept. But it was done in a post-filtering manner, i.e., the whole set of concepts should be generated first. *Are we able to avoid this redundant step for building the whole lattice?* In the next part of the manuscript we will positively answer to this question.

Part III

$\Sigma o\varphi\iota\alpha$: Searching for Optimal Formal Intents Algorithm

Chapter 9

Σοφια: an Algorithm for Mining Patterns for Nonmonotonic Constraints

Contents

9.1 Introduction

Interestingness measures were proposed to overcome the problem of combinatorial explosion of the number of valid patterns that can be discovered in a dataset (Vreeken and Tatti 2014). For example, pattern support, i.e., the number of objects covered by the pattern, is one of the most famous measures of pattern quality. In particular, support satisfies the property of anti-monotonicity (aka “a priori principle”), i.e., the larger the pattern is the smaller the support is (Agrawal and Srikant 1994; Mannila, Hannu Toivonen, et al. 1994). Many other measures can be mentioned such as utility constraint (Yao and Hamilton 2006), pattern stability (Kuznetsov 2007; Roth et al. 2008), pattern leverage (Webb 2010), margin closeness (Moerchen et al. 2011), MCCS (Spyropoulou et al. 2013), cosine interest (Cao et al. 2014), pattern robustness (Tatti et al. 2014), δ-freeness (Hébert and Crémilleux 2005), probability and separation of a pattern (Klimushkin et al. 2010), association rules based measures (Zimmermann 2013), basic levels (Belohlavek and Trnecka 2013), etc.

Some of these measures (e.g., support, robustness for generators (Tatti et al. 2014), or upper bound constraint of MCCS (Spyropoulou et al. 2013)) are “globally anti-monotonic”, i.e., for any two patterns $X \sqsubseteq Y$ we have $\mathcal{M}(X) \geqslant \mathcal{M}(Y)$, where $\mathcal{M}$ is a measure and $\sqsubseteq$ denotes the (subsumption) order relation on patterns. When a measure is anti-monotonic, it is relatively easy to find patterns whose measure is higher than a certain threshold (e.g., patterns with a support higher than a threshold). In contrast some other measures are called "locally anti-monotonic", i.e., for any pattern X there is an immediate subpattern $Y \prec X$ such that $\mathcal{M}(Y) \geqslant \mathcal{M}(X)$. The corresponding constraint induces an accessible system (Boley et al. 2010) in binary data. Indeed, for any itemset (a set of attributes) selected by a locally anti-monotonic constraint, one can always find a smaller selected itemset different only in one attribute. Then the right strategy should be selected for traversing the search space, e.g., a pattern Y should be extended only to patterns X such that $\mathcal{M}(Y) \geqslant \mathcal{M}(X)$. For example, for "locally anti-monotonic" cosine interest (Cao et al. 2014), the extension of a pattern Y consists in adding only attributes with a smaller support than any attribute from Y.

The most difficult case for selecting valid patterns occurs when a measure is not locally anti-monotonic. Then, valid patterns can be retained by postfiltering, i.e., finding a (large set of) patterns satisfying an antimonotone constraint and filtering them w.r.t. the chosen nonmonotonic measure (i.e., neither monotonic nor anti-monotonic) (Moerchen et al. 2011; Roth et al. 2008; Tatti et al. 2014), or using heuristics such as leap search (Yan, Cheng, et al. 2008) or low probability of finding interesting patterns in the current branch (Webb 2010).

Most of the measures are only applicable to one type of patterns, e.g., pattern leverage or cosine interest can be applied only to binary data since their definitions involve single attributes. "Pattern independent measures" usually relies on support of the pattern and/or on support of other patterns from the search space. In particular, support, stability (Kuznetsov 2007), margin-closeness (Moerchen et al. 2011) and robustness (Tatti et al. 2014) are pattern independent measures. Here we focus on pattern independent measures in order to deal with pattern structures.

In addition, given a measure, it can be difficult to define a good threshold. Thus various approaches for finding top-K patterns were introduced (Han, Wang, et al. 2002; Webb 2011; Xin et al. 2006), with the basic idea to automatically adjust the threshold for a measure $\mathcal{M}$.

In this chapter we introduce a brand-new algorithm Σοφια, i.e. Sofia, for "Searching for Optimal Formal Intents Algorithm", for extracting the best itemsets w.r.t. a wide class of constraints. Σοφια algorithm is applicable to a class of measures called "projection-antimonotonic measures" or more precisely "measures anti-monotonic w.r.t. a chain of projections". This class includes globally anti-monotonic measures such as support, locally anti-monotonic measures such as cosine interest and some of the nonmonotonic measures such as stability or robustness of closed patterns. We also introduce a way of adjusting a measure threshold such that the number of generated patterns is limited. This allows finding the best patterns w.r.t. a projection-antimonotonic measure in polynomial time modulo complexity of measure computation.

This chapter is based on (Buzmakov et al. 2015b) and divided into two parts. First, we introduce Σοφια algorithm in Section 9.2. Then, in Section 9.3 some nonmonotonic measures are discussed.

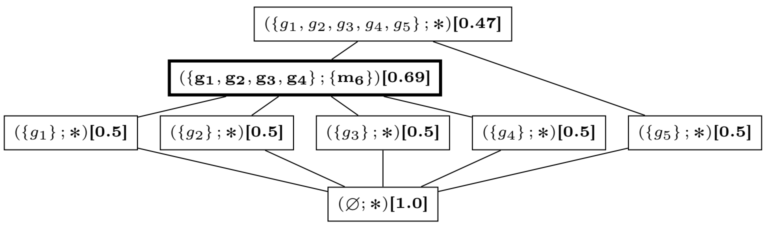
9.2 Description of Σοφια

9.2.1 Anti-monotonicity w.r.t. a Projection

Our algorithm is based on the projection-antimonotonicity, a new idea introduced in this paper. Many interestingness measures for patterns, e.g., stability (Kuznetsov 2007), robustness of closed patterns (Tatti et al. 2014), or cosine interest (Cao et al. 2014), are not (anti-)monotonic w.r.t. subsumption order on patterns. A measure $\mathcal{M}$ is called *anti-monotonic*, if for two patterns

	m_1	m_2	m_3	m_4	m_5	m_6
g_1	x		x			
g_2		x	x			
g_3			x	x		
g_4			x		x	
g_5						x

(a) A binary context.



(b) A concept lattice.

Figure 9.1: An example binary context and the corresponding concept lattice.

$q \sqsubseteq p$, $\mathcal{M}(q) \geqslant \mathcal{M}(p)$. For instance, support is a anti-monotonic measure w.r.t. pattern order and it allows for efficient generation of patterns with support larger than a threshold (Agrawal and Srikant 1994; Mannila, Hannu Toivonen, et al. 1994; Pasquier et al. 1999). The projection-antimonotonicity is a generalization of standard anti-monotonicity and allows for efficient work with a larger set of interestingness measures.

Definition 9.1. *Given a pattern structure $\mathbb{P}$ and a projection ψ, a measure $\mathcal{M}$ is called* anti-monotonic w.r.t. the projection ψ, *if*

$$(\forall p \in \psi(\mathbb{P}))(\forall q \in \mathbb{P}, \psi(q) = p)\ \mathcal{M}_\psi(p) \geqslant \mathcal{M}(q), \tag{9.1}$$

where $\mathcal{M}_\psi(p)$ is the measure $\mathcal{M}$ of a pattern p computed in $\psi(\mathbb{P})$.

Here, for any pattern p of a projected pattern structure $\psi(\mathbb{P})$, i.e., $p \in \psi(\mathbb{P})$, we check that a preimage q of p for ψ, i.e., $\psi(q) = p$, has a measure smaller than the measure of p. It should be noticed that a measure $\mathcal{M}$ for a pattern p can yield different values if $\mathcal{M}$ is computed in $\mathbb{P}$ or in $\psi(\mathbb{P})$. Thus we use the notation $\mathcal{M}_\psi$ for the measure $\mathcal{M}$ computed in $\psi(\mathbb{P})$. The property of a measure given in Definition 9.1 is called projection-antimonotonicity.

Example 9.1. *In the case of standard FCA, i.e., when descriptions are sets of attributes, the removal of attributes Y corresponds to a projection ψ. Given a set of attributes X (we assume $X \cap Y = \varnothing$), it can be seen that the set of preimages is given by*

$$\texttt{Preimages(X)} = \{Z \subseteq M \mid X \subseteq Z \subseteq X \cup Y\}, \tag{9.2}$$

where M is the set of attributes. In particular X is also a preimage of itself.

Example 9.2. *Let us consider the dataset in Figure 9.1a. If $\mathcal{M}$ is an interestingness measure w.r.t. a projection ψ and ψ removes attribute m_5, then $\mathcal{M}(\{m_3\}) \geqslant \mathcal{M}(\{m_3, m_5\})$. However it is **not** necessary that $\mathcal{M}(\{m_3\}) \geqslant \mathcal{M}(\{m_3, m_4\})$.*

Thus, given a measure $\mathcal{M}$ anti-monotonic w.r.t. a projection ψ, if p is a pattern such that $\mathcal{M}_\psi(p) < \theta$, then $\mathcal{M}(q) < \theta$ for any preimage q of p for ψ. Hence, if, given a pattern p of $\psi(\mathbb{P})$, one can find all patterns q of $\mathbb{P}$ such that $\psi(q) = p$, it is possible to first find all patterns of $\psi(\mathbb{P})$ and then to filter them w.r.t. $\mathcal{M}_\psi$ and a threshold, and finally to compute the preimages of filtered patterns only. It allows one to earlier cut unpromising branches of the search space or adjust a threshold for finding only a limited number of best patterns.

9.2.2 Anti-monotonicity w.r.t. a Chain of Projections

However, given just one projection, it can be hard to efficiently discover the best patterns, since the projection is either hard to compute or the number of unpromising patterns that can be pruned is not high. Corespondingly we need *a chain of projections* $\psi_0 < \psi_1 < \dots < \psi_k = \mathbb{1}$ (the order on projections is given by Definitions 5.6 and 5.7), where concepts for $\psi_0(\mathbb{P})$ can be easily computed and $\mathbb{1}$ is the identity projection, i.e., $(\forall x)\mathbb{1}(x) = x$. For example, to find frequent itemsets, we typically search for small frequent itemsets and then extend them to larger ones. It corresponds to the extension to a more detailed projection. In particular for binary dataset a chain of projections can be instantiated as a consequent update of a binary dataset with new attributes.

Chain of projections is a generalization of accessible system (Boley et al. 2010) in the case of binary contexts. Given a set of attributes M and a subset of its powerset $\mathcal{F} \subseteq 2^M$, the system $(M, \mathcal{F})$ is accessible if $\forall X \in \mathcal{F} \setminus \{\varnothing\}$ there is $i \in M$ such that $X \setminus \{i\} \in \mathcal{F}$. Any constraint (or measure) on 2^M produces a system of sets. If this system is accessible, then the measure is locally anti-monotonic.

Proposition 9.1. *A chain of projections can be represented as a sequence of systems* $(M_i, \mathcal{F}_i)$ *such that* $M_i \subset M_{i+1}$ *and any element* $x \in \mathcal{F}_{i+1}$ *is either (1)* $x \in \mathcal{F}_i$*, or (2)* $\exists e \in M_{i+1} \setminus M_i$ *such that* $(x \setminus \{e\}) \in \mathcal{F}_i$*, (3) or* x *accesible in* $\mathcal{F}_{i+1}$*.*

Proof. (1) by idempotency of projections, (2) by contractivity, (3) for deletion of several attributes. □

Definition 9.2. *Given a pattern structure* $\mathbb{P}$ *and a chain of projections* $\psi_0 < \psi_1 < \dots < \psi_k = \mathbb{1}$*, a measure* $\mathcal{M}$ *is called* anti-monotonic w.r.t. the chain of projections *if* $\mathcal{M}$ *is anti-monotonic w.r.t. all* ψ_i *for* $0 \leqslant i \leqslant k$*.*

9.2.3 Algorithms

Given a pattern structure $\mathbb{P}$ and a measure anti-monotonic w.r.t. a chain of projections, if we are able to find all preimages of any element in the fixed set of a projection ψ_i that belong to the fixed set of the next projection ψ_{i+1}, then we can find all patterns of the pattern structure $\mathbb{P}$ with a value of $\mathcal{M}$ higher than a given threshold θ. We call the respective algorithm ϑ-Σοφια (Algorithm 4). In lines 11-12 we find all patterns for $\psi_0(\mathbb{P})$ satisfying the constraint w.r.t. the measure $\mathcal{M}$. Then in lines 13-15 we iteratively extend projections from simpler to more detailed ones. The extension is done by constructing the set $\mathcal{P}_i$ of preimages of the set $\mathcal{P}_{i-1}$ (lines 2-5) and then by removing the patterns that do not satisfy the constraint from $\mathcal{P}_i$ (lines 6-9).

The algorithm is sound and complete, since first, a pattern p is included into the set of preimages of p (since $\psi(p) = p$) and second, if we remove the pattern p from the set $\mathcal{P}$, then the value $\mathcal{M}(p) < \theta$ and, hence, the measure value of any preimage of p is less than θ by the projection-antimonotonicity of $\mathcal{M}$. The worst case time complexity of ϑ-Σοφια algorithm is

$$\mathbb{T}(\vartheta\text{-}\Sigma o\varphi\iota\alpha) = \mathbb{T}(FindPatterns(\psi_0)) + k \cdot \max_{0<i\leqslant k} |\mathcal{P}_i| \cdot (\mathbb{T}(Preimages) + \mathbb{T}(\mathcal{M})), \tag{9.3}$$

where k is the number of projections in the chain, $\mathbb{T}(\mathcal{X})$ is the time for computing the operation $\mathcal{X}$. Since projection ψ_0 can be chosen to be very simple, in a typical case the complexity of $FindPatterns(\theta, \psi_0)$ can be low or even constant. The complexities of *Preimages* and $\mathcal{M}$ depend on the measure, the chain of projections, and the kind of patterns. In many cases $\max_{0<i\leqslant k} |\mathcal{P}_i|$ can be exponential in the size of the input, because the number of patterns can be

```
Data: A pattern structure $\mathbb{P}$, a chain of projections $\Psi = \{\psi_0, \psi_1, \cdots, \psi_k\}$, an
      anti-monotonic measure $\mathcal{M}$ for the chain $\Psi$, and a threshold $\theta$ for $\mathcal{M}$.
1  Function ExtendProjection($i$, $\theta$, $\mathcal{P}_{i-1}$)
       Data: $i$ is the projection number to which we should extend ($0 < i \leqslant k$), $\theta$ is a
             threshold value for $\mathcal{M}$, and $\mathcal{P}_{i-1}$ is the set of patterns for the projection $\psi_{i-1}$.
       Result: The set $\mathcal{P}_i$ of all patterns with the value of measure $\mathcal{M}$ higher than the
               threshold $\theta$ for $\psi_i$.
2      $\mathcal{P}_i \longleftarrow \varnothing$;
3      /* Put all preimages in $\psi_i(\mathbb{P})$ for any pattern $p$ */
4      foreach $p \in \mathcal{P}_{i-1}$ do
5          $\mathcal{P}_i \longleftarrow \mathcal{P}_i \cup$ Preimages($i$,$p$)
6      /* Filter patterns in $\mathcal{P}_i$ to have a value of $\mathcal{M}$ higher than $\theta$ */
7      foreach $p \in \mathcal{P}_i$ do
8          if $\mathcal{M}_{\psi_i}(p) \leqslant \theta$ then
9              $\mathcal{P}_i \longleftarrow \mathcal{P}_i \setminus \{p\}$
10 Function Algorithm_θ-Σοφια
       Result: The set $\mathcal{P}$ of all patterns with a value of $\mathcal{M}$ higher than the threshold $\theta$ for
               $\mathbb{P}$.
11     /* Find all patterns in $\psi_0(\mathbb{P})$ with a value of $\mathcal{M}$ higher than $\theta$ */
12     $\mathcal{P} \longleftarrow$ FindPatterns($\theta, \psi_0$);
13     /* Run through out the chain $\Psi$ and find the patterns for $\psi_i(\mathbb{P})$ */
14     foreach $0 < i \leqslant k$ do
15         $\mathcal{P} \longleftarrow$ ExtendProjection($i, \theta, \mathcal{P}$);
```

Algorithm 4: The ϑ-Σοφια algorithm for finding patterns in $\mathbb{P}$ with a value of a measure $\mathcal{M}$ higher than a threshold θ.

exponential. Thus, taken into account that every $\mathcal{P}_i$ is a solution for the projected context, i.e., a context without some attributes, this algorithm has incremental polynomial delay.

In order to have a polynomial algorithm one should limit the size of the sets $\mathcal{P}_i$. However it can be a difficult task to define the threshold θ ensuring that the size of $\mathcal{P}_i$ is limited. Thus, we introduce Σοφια algorithm (Algorithm 5), which automatically adjusts threshold θ ensuring that $\max\limits_{0<i\leqslant k} |\mathcal{P}_i| < L$. Here L can be considered as a constraint on the memory used by the algorithm. As the result it returns the threshold θ ensuring that the cardinality of the set $\mathcal{P}$ is bounded by L in any step of the algorithm and the set $\mathcal{P}$ of all patterns with the value of measure $\mathcal{M}$ higher than the threshold θ. The only difference of Σοφια w.r.t. ϑ-Σοφια is that after performing an operation that changes the set $\mathcal{P}$ (lines 4 and 10 in Algorithm 5) it adjusts θ in such a way that the cardinality of $\mathcal{P}$ does not exceed the parameter L. It can be seen from (9.3) that Σοφια has polynomial time complexity if $\mathcal{M}$ and *Preimages* are polynomial.

Example 9.3. *For a binary context, according to (9.2) if a projection removes only one attribute, the cardinality of* Preimages *is always 2. Thus, the worst case complexity for ϑ-Σοφια is*

$$\mathbb{T}(\vartheta\text{-}\Sigma o\varphi\iota\alpha_{binary}) = |M| \cdot \max_{0<i\leqslant N} |\mathcal{P}_i| \cdot \mathbb{T}(\mathcal{M}). \tag{9.4}$$

However, if we fix the available memory L, the complexity of Σοφια for binary data is $|M| \cdot L \cdot \mathbb{T}(\mathcal{M})$, i.e., it becomes input polynomial modulo complexity of the measure.

```
   Data: Pattern structure $\mathbb{P}$, a chain of projections $\Psi = \{\psi_0, \psi_1, \cdots, \psi_k\}$, a measure $\mathcal{M}$
         anti-monotonic for the chain $\Psi$, and a threshold $L$ for the maximal number of
         preserved patterns.
1  Function Algorithm_Σοφια
       Result: The threshold $\theta$ ensuring that the cardinality of the set $\mathcal{P}$ is bounded by $L$
               in any step of the algorithm. The set $\mathcal{P}$ of all patterns with the value of
               measure $\mathcal{M}$ higher than the threshold $\theta$.
2      $\theta \longleftarrow \theta_{\min}$;                          /* Set $\theta$ to the minimal value */
3      /* Find all patterns in $\psi_0(\mathbb{P})$ with a value of $\mathcal{M}$ higher than $\theta$ */
4      $\mathcal{P} \longleftarrow$ FindPatterns($\psi_0$);
5      /* Adjust threshold and filter patterns */
6      $\theta \longleftarrow$ AdjustTheta($\theta, L, \mathcal{P}$);
7      $\mathcal{P} \longleftarrow$ FilterPatterns($\theta, \mathcal{P}$);
8      /* Run through out the chain $\Psi$ and find the result patterns */
9      foreach $0 < i \leqslant k$ do
10         $\mathcal{P} \longleftarrow$ ExtendProjection($i, \theta, \mathcal{P}$);
11         /* Adjust threshold and filter patterns */
12         $\theta \longleftarrow$ AdjustTheta($\theta, L, \mathcal{P}$);
13         $\mathcal{P} \longleftarrow$ FilterPatterns($\theta, \mathcal{P}$);
```

Algorithm 5: The Σοφια algorithm for finding patterns in $\mathbb{P}$ with the bounded cardinality of the set $\mathcal{P}$.

Efficiency Considerations

Recently much work have been done in finding good strategies of enumerating (closed) patterns and, in particular, closed itemsets. Most of them start from the smallest patterns and then iteratively generate larger patterns. It can be naturally expressed as a chain of functions ψ_i that are contractive ($\psi_i(p) \sqsubseteq p$) and idempotent ($\psi_i(\psi_i(p)) = \psi_i(p)$). These functions can be ordered by inclusion of fixed sets because of idempotency. Since these functions are contractive, only patterns larger than a pattern p are preimages of p. Thus, most of the approaches for itemset mining can be formalized by means of a chain of such functions. However, in this work we require a chain of projections, i.e., functions ψ_i, to be also monotone. It allows us to efficiently mine robust and stable patterns discussed in Section 9.3. This additional monotonicity still allows one to formalize developed approaches for itemset mining as a chain of projections. However, in this chapter we does not discuss this formalization and focus on the efficient mining of patterns for nonmonotonic constraints.

9.2.4 Σοφια Algorithm for Closed Patterns

Closed frequent patterns are widely used as a condensed representation of all frequent patterns since (Pasquier et al. 1999). Here we show how we can adapt the algorithm for closed patterns. A closed pattern in $\psi_{i-1}(\mathbb{P})$ is not necessarily closed in $\psi_i(\mathbb{P})$. Indeed, if we take the example of a binary data in Figure 9.1, the pattern $\{m_1\}$ is closed in $(G, \{m_1, m_2\}, I)$ but no more closed in $(G, \{m_1, m_2, m_3\}, I)$. However, the extents of $\psi(\mathbb{P})$ are extents of $\mathbb{P}$ (Buzmakov et al. 2015c; Ganter and Kuznetsov 2001). Thus, we associate the closed patterns with extents and then work with extents instead of patterns.

In order to do this, a pattern structure $\mathbb{P} = (G, (D, \sqcap), \delta)$ is transformed into another pattern structure $\mathbb{P}_C = (G, (D_C, \sqcap_C), \delta_C)$, where $D_C = 2^G$. Moreover, for all $x, y \in D_C$ we have $x \sqcap_C y = (x^\diamond \sqcap y^\diamond)^\diamond$, where diamond operator is computed in $\mathbb{P}$ and $\delta_C(g \in G) = \{g\}$. Hence, every pattern p in D_C corresponds to a closed pattern $p^\diamond$ in D. First, $(D_C, \sqcap_C)$ is a semilattice, because it corresponds to the lattice of extents of $\mathbb{P}$. A projection ψ of $\mathbb{P}$ induces a projection ψ_C of $\mathbb{P}_C$, given by $\psi_C(X \subseteq G) = \psi(X^\diamond)^\diamond$ with $(\cdot)^\diamond$ for $\mathbb{P}$. The function ψ_C is a projection because of the properties of $(\cdot)^\diamond$ operators and ψ mappings.

In the next section we discuss some measures that are anti-monotonic w.r.t. a projection (rather than just anti-monotonic). In the end of the next section we provide an example of how Σοφια works for the case of binary data.

9.3 Pattern Constraints

Table 9.1: Values of different measures for closed itemesets of context in Figure 9.1a.

Itemset X	Cosine	$\mathtt{Stab}(X)$	$\underset{\alpha=0.9}{\mathtt{Rbst}(X)}$	$\Delta(X)$
$\varnothing$	$+\infty$	0.47	0.89991	1
$\{\mathbf{m_3}\}$	**1**	**0.69**	**0.9963**	3
$\{m_1, m_3\}$	0.5	0.5	0.9	1
$\{m_2, m_3\}$	0.5	0.5	0.9	1
$\{m_3, m_4\}$	0.5	0.5	0.9	1
$\{m_3, m_5\}$	0.5	0.5	0.9	1
$\{m_6\}$	1	0.5	0.9	1

9.3.1 Cosine Interest of an Itemset

Let us start with cosine interest (Cao et al. 2014), a projection-antimonotonic measure defined for set of attributes (the case of binary data). It is defined by

$$\mathtt{Cosine}(X) = \frac{|X^\diamond|}{\sqrt[|X|]{\prod_{m \in X} |\{m\}^\diamond|}}, \tag{9.5}$$

i.e., a cosine interest of X is the support of X over the geometric mean of supports of single attributes from X. Cao et al. (2014) have shown that this measure is not (anti-)monotonic. Then, they also have shown that if we traverse the search space from less supported attributes to more supported attributes the cosine interest never decreases. Indeed, given an itemset X and an attribute i such that $i \notin X$ and $(\forall j \in X) |\{i\}^\diamond| \geqslant |\{j\}^\diamond|$, we can see that $\mathtt{Cosine}(X) \geqslant \mathtt{Cosine}(X \cup \{i\})$ since the itemset support cannot increase while the geometric mean cannot decrease in this case.

To work with cosine interest we can define a projection chain that adds attributes from less supported ones to more supported, i.e., ψ_1 corresponds to removal of all but the least frequent attribute from the dataset, ψ_2 corresponds to removal of all but two least frequent attributes and so on. Then, cosine interesting itemsets can be mined by Σοφια. However this measure is locally anti-monotonic, in the next subsection we consider two proper nonmonotonic measures.

9.3.2 Stability and Robustness of a Pattern

Stability (Kuznetsov 2007) and robustness (Tatti et al. 2014) are similar measures when applied to closed patterns. They measure independence of a pattern wrt. subsampling. Stability can only be applied to closed patterns, while robustness is defined for any type of itemset constraints (closed patterns, generators, *etc.*) and it can be generalized for pattern constraints. In the case of closed patterns neither stability nor robustness are (anti-)monotonic. Indeed, when robustness is based on an anti-monotonic constraint, it is anti-monotonic. However, closedness of a pattern is not an anti-monotonic constraint. Since stability and robustness are similar, we define them here on a similar basis. It should be noticed, that the way stability defined here is different from Chapter 2. However, it will be the same stability and, thus, the previous understating of stability can be used as a proxy for understanding robustness.

Given a dataset (a pattern structure) $\mathbb{P} = (G, (D, \sqcap), I)$, a triple $\mathbb{P}_s(S, (D, \sqcap), I)$ where $S \subseteq G$ is called a *subdataset* of $\mathbb{P}$. If we give a weight to every subdataset of $\mathbb{P}$, then we can find the sum of weights of all subdatasets of $\mathbb{P}$ where a pattern p is closed. This sum gives us stability or robustness of the closed pattern p depending on how we define the weights of subdatasets.

In the case of *stability* the weights w of all subdatasets $\mathbb{P}_s$ of $\mathbb{P}$ are equal, i.e., $w(\mathbb{P}_s) = 2^{-|G|}$. In this case we consider every subdataset equally probable and compute the probability that the pattern p is closed.

Example 9.4. *Consider example in Figure 9.1a. The set of concepts (the pattern of every concept is a closed itemset) is shown in Figure 9.1b. Stability of every closed itemset is shown in Table 9.1. Let us consider the highlighted itemset $X = \{m_3\}$. There are 2^5 possible subdatasets. Only in the following 10 subdatasets X is not closed (only the set of objects for every subdataset is given): $\emptyset, \{g_1\}, \dots, \{g_5\}, \{g_1, g_5\}, \{g_2, g_5\}, \{g_3, g_5\}, \{g_4, g_5\}$. Thus, stability of X can be found as $\texttt{Stab}(X) = 1 - 10 \cdot 2^{-5} = 0.69$.*

It should be noticed that stability of all comparable itemsets in the lattice is smaller than stability of X, which highlights the nonmonotonicity of stability.

In the case of *robustness* the weights w of subdatasets are computed differently. These weights depend on a parameter $0 \leqslant \alpha \leqslant 1$ denoting the probability of an object to be retained in the dataset. The weight of a subdataset $\mathbb{P}_s = (S, (D, \sqcap), I)$ of $\mathbb{P} = (G, (D, \sqcap), I)$ corresponds to the probability of obtaining $\mathbb{P}_s$ by removing objects from $\mathbb{P}$ with probability $1 - \alpha$: $w(\mathbb{P}_s) = \alpha^{|S|} \cdot (1 - \alpha)^{|G| - |S|}$.

Example 9.5. *Consider example in Figure 9.1a. Robustness for $\alpha = 0.9$ for every closed itemset is shown in Table 9.1. Let us consider the highlighted itemset $X = \{m_3\}$. It is not closed in the same as above 10 subdatasets but their weights are different (the weights are shown in brackets): $\emptyset$ ($w = 10^{-5}$), $\{g_1\}$ ($w = 9 \cdot 10^{-5}$), ..., $\{g_5\}$ ($w = 9 \cdot 10^{-5}$), $\{g_1, g_5\}$ ($w = 8.1 \cdot 10^{-4}$), $\{g_2, g_5\}$ ($w = 8.1 \cdot 10^{-4}$), $\{g_3, g_5\}$ ($w = 8.1 \cdot 10^{-4}$), $\{g_4, g_5\}$ ($w = 8.1 \cdot 10^{-4}$). Thus, robustness of X for $\alpha = 0.9$ is equal to $\texttt{Rbst}_{\alpha=0.9}(X) = 0.9963$. It can be verified that robustness is not a (locally) anti-monotonic measure either.*

It is not hard to show that independently of the weights w of subdatasets, stability and robustness are anti-monotonic measures w.r.t. any projection.

Proposition 9.2. *Stability and robustness are anti-monotonic measures w.r.t. any projection.*

Proof. Here we want to show that for any projection ψ if a pattern p is closed in a subdataset $\mathbb{P}_s$ then $\psi(p)$ is closed in $\psi(\mathbb{P}_s)$, where $\mathbb{P}_s = (S, (D, \sqcap), I)$ is a subdataset of $\mathbb{P} = (G, (D, \sqcap), I)$ with $S \subseteq G$. We note that if p is closed in $\mathbb{P}_s$ it is also closed in $\mathbb{P}$. And since $\psi(p)$ closed in $\psi(\mathbb{P})$, then for projection ψ_C from Section 9.2.4 we have $\psi_C(p^{\diamond\diamond}) = \psi(p)$. Hence, we can work with images of ψ on closed patterns in order to find the corresponding images of ψ_C.

Let $q = (\psi(p)^{\diamond} \cap S)^{\diamond}$ be a closure of $\psi(p)$ in $\mathbb{P}_s$. Since $\psi(p) \sqsubseteq p$, then $\psi(p)^{\diamond} \supseteq p^{\diamond}$. Hence $S \cap \psi(p)^{\diamond} \supseteq S \cap p^{\diamond}$. Then $q = (S \cap \psi(p)^{\diamond})^{\diamond} \sqsubseteq (S \cap p^{\diamond})^{\diamond} = p$, since q is the closure of $\psi(p)$

in $\psi(\mathbb{P}_s)$ and p is the closure of p in $\mathbb{P}_s$. Thus, we have $q \sqsubseteq p$ and $q^{\diamond\diamond} \sqsubseteq p^{\diamond\diamond} = p$. Because of monotonicity of projections one has $\psi(q^{\diamond\diamond}) \sqsubseteq \psi(p)$ and hence $q \sqsubseteq \psi(p)$.

Since q is the closure of $\psi(p)$ in $\mathbb{P}_s$, then $q \sqsupseteq \psi(p)$. Hence $q = \psi(p)$. □

Estimates of Stability and Robustness

For stability and robustness it is shown that the corresponding constraint is NP-hard (Kuznetsov 2007; Tatti et al. 2014). Thus, for efficient mining, estimates of stability and robustness are essential. Here we introduce a fast computable estimate of robustness in the same way we did it for stability in Buzmakov et al. (2014d).

Let us consider closed patterns p and q such that $p \subset q$. *Can we define the subdatasets where p is not closed?* Let us define the set $\Delta(p,q)$ as the set of objects described by p but not by q: $\Delta(p,q) = p^{\diamond} \backslash q^{\diamond}$. This set is not empty since $p \neq q$ and they are closed. It is clear that p is not closed in any subdataset that removes all objects from $\Delta(p,q)$, since q is a larger pattern with the same support. Then, $\texttt{Stab}(p) \leqslant 1 - 2^{-\Delta(p,q)}$ and $\texttt{Rbst}(p) \leqslant 1 - (1-\alpha)^{\Delta(p,q)}$ for any closed pattern $q \supset p$. In particular, we can put q to the closest closed superpattern of p.

In the same way we can take all immediate closed superpattern of p and take into account all the subdatasets where p is not closed. Since some of the subdatasets are probably counted several times we get the lower bound, i.e., $\texttt{Stab}(p) \geqslant 1 - \sum_{q \prec p} 2^{-\Delta(p,q)}$ and $\texttt{Rbst}(p) \geqslant 1 - \sum_{q \prec p} (1-\alpha)^{\Delta(p,q)}$.

Proposition 9.3. *Stability and robustness are bounded as follows, where $p \prec q$ means that p is an immediate closed subpattern of q:*

$$1 - \sum_{q \prec p} 2^{-\Delta(p,q)} \leqslant \texttt{Stab}(p) \leqslant 1 - 2^{-\Delta(p,q)} \tag{9.6}$$

$$1 - \sum_{q \prec p} (1-\alpha)^{\Delta(p,q)} \leqslant \texttt{Rbst}(p) \leqslant 1 - (1-\alpha)^{\Delta(p,q)} \tag{9.7}$$

In particular we can see that when $\alpha = 0.5$ the estimates are exactly the same. As it is recently shown (Buzmakov et al. 2014d), the estimate of stability is quite precise for the concepts with stability close to 1. Then, when $\alpha > 0.5$ the precision of the estimate of robustness is even more precise.

These estimates can be computed in polynomial time in contrast to stability and robustness. And thus we can use one of the bounds as a proxy to stability and robustness. In this case the rankings based on the upper bound of stability and robustness are exactly the same as the ranking based on $\Delta(p) = \min_{q \prec p} \Delta(p,q)$. Although for the lower bound of stability and robustness it is hard to show the projection anti-monotonicity, we can show it for the upper bound. In the following $\Delta(p)$ is called Δ-measure.

Proposition 9.4. *Δ-measure is an anti-monotonic measure w.r.t. any projection.*

Proof. By properties of projections, if an extent is found in $\psi(\mathbb{P})$, it is necessarily found in $\mathbb{P}$ (Ganter and Kuznetsov 2001). Let us consider an extent E and an extent of its descendant E_c in $\psi(\mathbb{P})$. Let us suppose that E_p is a preimage of E for the projection ψ. Since E_c and E_p are extents in $\mathbb{P}$, the set $E_{cp} = E_c \cap E_p$ is an extent in $\mathbb{P}$ (the intersection of two closed sets is a closed set). Since E_p is a preimage of E, then $E_p \nleqslant E_c$ (otherwise, E_p is a preimage of E_c and not of E). Then, $E_{cp} \neq E_p$ and $E_{cp} \leqslant E_p$. Hence, $\Delta(E_p) \leqslant |E_p \backslash E_{cp}| \leqslant |E \backslash E_c|$. So, given a preimage E_p of E, $(\forall E_c < E)\Delta(E_p) \leqslant |E \backslash E_c|$, i.e., $\Delta(E_p) \leqslant \Delta(E)$. Thus, we can use Δ-measure in combination with Σοφια algorithm. □

Table 9.2: Patterns given by their extent and their stability in the contexts corresponding to a chain of projections.

#	Pattern Ext.	Δ-measure						
		M_0	M_1	M_2	M_3	M_4	M_5	M_6
1	12345	5	4	4	1	1	1	1
2	1	–	1	1	1	1	1	1
3	2	–	–	1	1	1	1	1
4	1234	–	–	–	3	3	3	3
6	3	–	–	–	–	1	1	1
7	4	–	–	–	–	–	1	1
8	5	–	–	–	–	–	–	1

Example 9.6. *Consider binary context in Figure 9.1a. Δ-measure for every closed itemset is shown in Table 9.1. Let us consider the highlighted itemset $X = \{m_3\}$ with support equal to 4. The closest superitemsets of X are $\{m_1, m_3\}$, $\{m_2, m_3\}$, $\{m_3, m_4\}$, and $\{m_3, m_5\}$, all having support equal to one. Thus, Δ-measure of X is equal to $\Delta(X) = 4 - 1 = 3$. It can be noticed that Δ-measure is not an (anti-)monotonic measure.*

Δ-measure is related to the work of margin-closeness of an itemset (Moerchen et al. 2011). In this work, given a set of patterns, e.g., frequent closed patterns, the authors rank them by the minimal distance in their support to the closest superpattern divided by the support of the pattern. In our case, the minimal distance is exactly the Δ-measure of the pattern.

9.3.3 Example of Δ-Stable Itemsets in Binary Data

Let us consider the example in Figure 9.1 and show how we can find all Δ-stable patterns with threshold $\theta = 2$. We have a binary dataset $\mathbb{K} = (T, \{m_1, \cdots, m_6\}, R)$. Let us denote $M_i = \{m_1, \cdots, m_i\}$. The sets M_i correspond to a chain of projections.

In Table 9.2 all closed itemsets are given by the corresponding tidsets, i.e., by elements of D_C. For simplicity we write 1234 instead of $\{g_1, g_2, g_3, g_4\}$. For every element Δ-measure is shown for every M_i. A cell is shown in gray if the pattern is no more considered (the value of Δ is less than 2).

For example, in the transition from M_2 to M_3 the set 1234 is discovered with $\Delta(1234) = 3$, but $\Delta(12345) = 5 - 4 = 1$ which is less than $\theta = 2$. Thus, pattern 12345 is discarded and highlighted gray. The global process is as follows (for the example in Figure 9.1). In the empty binary dataset $(T, \emptyset, R)$ the first pattern 12345 is considered. Then, in $(T, \{m_1\}, R)$ a possible preimage of 12345 can be either 12345 or $12345 \cap \{m_1\}^\diamond = 1$. The set 12345 is Δ-stable ($\Delta(12345) = 4$), while 1 is not Δ-stable ($\Delta(1) = 1$) and is discarded. Then, the process continues with $(T, \{m_1, m_2\}, R)$ and 12345 is kept while $12345 \cap \{m_2\}^\diamond = 2$ is removed for the same reason as 1. After that, with $(T, \{m_1, m_2, m_3\}, R)$ two preimages are still considered, 12345 and 1234. This time $\Delta(1234) = 3$, while $\Delta(12345) = 1$ and the set 12345 is discarded. The process continues in the same way with $\Delta(1234) = 3$ and all other possible elements are discarded.

9.4 Conclusion

In this chapter we have introduced a new class of interestingness measures that are anti-monotonic w.r.t. a chain of projections. We have related this kind of anti-monotonicity to accesible systems and show that projection-antimonotonicity is a generalization of accesible systems.

We have designed a new algorithm, called Σοφια, which is able to efficiently find the best patterns w.r.t. such interestingness measures for different types of patterns. The work of the algorithm is exemplified on binary contexts. We highlight that Σοφια can find patterns w.r.t. projection-antimonotonic measures in input polynomial time. We have also shown that anti-monotonic measures, locally anti-monotonic measures and some nonmonotonic measures, e.g., stability and robustness, are projection antimonotonic.

In next chapter we will experimentally study the efficiency of Σοφια for two kinds of patterns. Firstly we will use different binary contexts and compare Σοφια with postfiltering approaches for mining Δ-stable patterns. Then we will switch to interval patterns (Kaytoue, Kuznetsov, Napoli, and Duplessis 2011) and show how one can run Σοφια for such kind of patterns.

Chapter 10

Experimental Evaluation of Σοφια

Contents

10.1 Introduction

In the previous chapter we have introduced Σοφια, an algorithm for mining patterns w.r.t. non-monotonic constraints. Here, we study how it behaves on real data. Since Σοφια is defined for different kinds of patterns, it is necessary to study different types of real data. Correspondingly, we apply Σοφια for mining top Δ-stable itemsets and interval tuples.

Itemsets are patterns of binary contexts that are studied from the beginning of data mining (Agrawal et al. 1993b; Ganter and Wille 1999; Mannila, Hannu Toivonen, et al. 1994). In the previous chapter we have seen some examples of binary (itemset) data and its relation to Σοφια. Here we also evaluate Σοφια for interval tuple data (Kaytoue, Kuznetsov, Napoli, and Duplessis 2011), a kind of data appearing in analysis of numerical data. This data have not been yet discussed and, thus, the corresponding pattern structure and the adaptation of Σοφια is also provided in this chapter.

To the best of our knowledge Σοφια is the first algorithm that computes top Δ-stable patterns, so there are no direct competitors. Thus, in this chapter we compare Σοφια approach to approaches based on postfiltering. Indeed, the known approaches use postfiltering to mine such kind of patterns (Buzmakov et al. 2013b; Moerchen et al. 2011; Roth et al. 2008; Tatti

Table 10.1: Computational efficiency of Σοφια algorithm.

Dataset	Top-K	$\theta_{Supp.}$	θ_{Δ}	`LCMv3`	Δ	`Charm-L` ($\sim$ `Charm-L` + Δ)	Σοφια
FIMI							
chess	3	1145	234	1.62	> 100	> 100	**0.03**
	928	277	98	> 100	—	> 100	**0.13**
connect	1	25466	4224	0.21	128	111	**0.61**
	1000	8822	2602	1.25	> 100	> 100	**1.77**
mushroom	1	6272	2256	< 0.01	0.07	**0.01**	0.05
	722	216	193	0.06	2.12	0.50	**0.23**
pumsb	1	33128	2035	0.15	> 300	36.7	**0.8**
	984	8793	865	> 300	—	> 300	**38.7**
pumsb*	1	30787	8090	0.04	1.42	0.16	**0.65**
	997	2808	834	4.47	> 300	> 300	**27.8**
LUCS							
adult	1	34338	6939	0.01	0.78	0.05	**0.20**
	998	674	446	0.11	16.45	2.15	**1.27**
waveform	1	3424	1179	**$<$ 0.01**	**0.01**	**0.01**	0.03
	984	401	141	0.09	4.42	1.24	**0.25**
UCI							
plants	1	11676	6154	< 0.01	0.11	0.02	**0.11**
	984	649	148	> 100	—	> 100	**0.96**

et al. 2014). The general idea of the experiments below is to run Σοφια to find top most Δ-stable patterns. Then, we can find the corresponding support threshold ensuring that competitors finds all these top patterns and in addition they can find some other patterns but this is inevitable in postfiltering approaches. Then the computational time of Σοφια is compared to the computational time of competitors. The experiments are carried out on an "Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz" computer with 8Gb of memory under Ubuntu 14.04 operating system. The algorithms are not parallelized and are coded in `C++`.

This chapter is based on Buzmakov et al. (2015b) and is divided into two parts. The first one discusses the evaluation of Σοφια on real binary data. The second part introduces interval tuple data, discuss how it can be processed by Sofia and provide the corresponding experiments.

10.2 Σοφια and binary data

10.2.1 Comparing Computational Efficiency

In the first experiment we show the computational efficiency of Σοφια for binary data. We use public available big datasets from FIMI[20], LUCS (Coenen 2003), and UCI (Frank and Asuncion 2010) repositories.

We should note two points here. First, there is no direct competitors and, moreover, computing Δ-measure for a pattern requires either a known partial order of patterns or a search for its descendants. Thus, as an approximate competitors we decided to use two algorithms

[20] `http://fimi.ua.ac.be/data/`

LCMv3 (Uno, Kiyomi, et al. 2005) and Charm-L (Zaki and Hsiao 2005). The first one is one of the most efficient algorithms for itemset mining that should be followed by Δ-measure computation for every concept. Charm-L is less efficient than LCMv3, but allows one to find the partial order of itemsets necessary for the fast computation of Δ-measure.

Second, the current implementation of Σοφια does not use most of the modern optimization techniques, e.g., like in LCMv3 (Uno, Kiyomi, et al. 2005). The current implementation relies only on the so-called conditional database, i.e., where for every itemset X the attributes that belong to all objects from X' and the attributes that belong to neither objects from X' are recorded (Uno, Kiyomi, et al. 2005). But nevertheless, the computation with the current implementation is efficient.

The experiment is organized as following. First, Σοφια finds around the 1000 most Δ-stable itemsets and the maximal support threshold ensuring to find all these the most Δ-stable itemsets. Among them we find the most Δ-stable itemset (or itemsets if they have the same value of Δ-measure) and the corresponding support threshold. So LCMv3 and Charm-L are additionally provided with an oracle returning the required support thresholds. For these two thresholds we run LCMv3 and Charm-L algorithm and register the computation time. In addition for LCMv3 we register also the time needed for computing Δ-measure, while for Charm-L this time is insignificant. In Table 10.1 for every dataset we give the results corresponding to every threshold, and the corresponding thresholds for support and Δ-measure. For example, for dataset chess we run two experiments. In the first one we search for top-3 Δ-stable patterns having the same value (234) for Δ-measure. The less frequent pattern among these three has support equal to 1145, thus, LCMv3 and Charm-L should be run with this support threshold in order to enumerate all of these patterns. LCMv3 finds the corresponding frequent closed patterns in 1.67 seconds, then it takes more than 100 seconds for computing Δ-measure. Charm-L takes more than 100 seconds and Σοφια requires only 0.03 seconds. In the second experiment for dataset chess we search for top-928 Δ-stable patterns, all of them have support at least 277 and Δ-measure 98.

We boldify the computation time for an algorithm in Table 10.1, if it is better than the time of the competitors. We can see that even LCMv3 alone does not always beat Σοφια, while the additional time for LCMv3 for computing Δ-measure is always significant. There are only two cases when Σοφια is slightly worse (FIMI-mushroom and LUCS-waveform). For both cases the most stable pattern has a very high support and only a couple of itemsets are frequent enough in both datasets. In contrast, if the frequency of the most Δ-stable patterns is not high, then Σοφια is many times faster than even LCMv3 alone.

10.2.2 Scalability

We can study scalability of Σοφια from different points of view. First, we can measure the time necessary for finding top-L concepts, i.e., how the memory limitation L changes the efficiency. It is shown in Table 10.2 for the same datasets. We can see that the computation time changes linearly w.r.t. the memory limitation L as it is expected from Eq. (9.4).

Finally, we check how computation time depends on the size of the dataset. For that we run our experiments for $L = 1000$, and vary the number of objects in a dataset. We permute several time the order of objects of the dataset. For every permutation we construct datasets containing certain amount (the size of the dataset) of the first objects from this permutation. The computation time is averaged over the permutations. Figure 10.1 shows the computation time necessary to process a certain fraction of objects in the dataset. Time is given as a fraction of time for processing the whole dataset. We can see that computation time changes linearly w.r.t. the fraction of processed objects.

Table 10.2: Scalability of Σοφια w.r.t. the number of stored patterns

Dataset	L=100	L=1000	$L = 10^4$	$L = 10^5$
		FIMI		
chess	0.04	0.13	1.35	14.7
connect	0.70	1.77	12.7	131
mushroom	0.1	0.29	2.62	40.5
pumsb	7.15	71.5	904	—
pumsb*	4.14	45.7	832	—
		LUCS		
adult	0.30	0.99	8.79	83.97
waveform	0.06	0.18	1.97	22.13
		UCI		
plants	0.22	1.09	11.58	117.91

Table 10.3: Evaluation results of Σοφια algorithm for Δ-measure.

Datasets	$L = 10^3$			$L = 10^4$			$L = 10^5$		
	t	#	θ	t	#	θ	t	#	θ
				Decreasing order					
Mushrooms	< 1	0.99	181	2	0.87	49	39	0.89	7
Chess	< 1	0.997	97	2	0.92	69	17	0.94	46
Plants	1	1	147	14	0.96	70	146	0.94	37
				Increasing order					
Mushrooms	1	0.99	181	6	0.87	49	38	0.89	7
Chess	1	0.88	144	4	0.24	84	38	0.68	49
Plants	3	1	147	29	0.96	70	263	0.94	37
				Random order					
Mushrooms	< 1	0.99	181	3	0.87	49	117	0.89	7
Chess	< 1	0.65	103	2	0.92	69	19	0.94	46
Plants	1	1	147	14	0.96	70	143	0.94	37

10.2.3 Behavior of Σοφια on Binary Dataset

Let us now study in more details the behaviour of Σοφια on the most popular databases from UCI repository (Frank and Asuncion 2010). The dataset `Mushrooms`[21] is a dataset having a relatively small number of closed patterns, all of them can be found in some seconds, while the datasets `Chess`[22] and `Plants`[23] have a lot of closed patterns, which can be hardly found.

There are two obvious orders for adding an attribute in Σοφια by means of chain of projections: the decreasing and increasing orders of attribute support. We consider also a random order of attributes allowing one to discard any bias in the order of attributes. Another point about our algorithm that we should study is how many Δ-stable concepts it finds if the memory usage is limited by a constant L. It finds no more than L patterns allowing to compute the result in

[21]https://archive.ics.uci.edu/ml/datasets/Mushroom
[22]https://archive.ics.uci.edu/ml/datasets/Chess+(King-Rook+vs.+King-Knight)
[23]https://archive.ics.uci.edu/ml/datasets/Mushroom

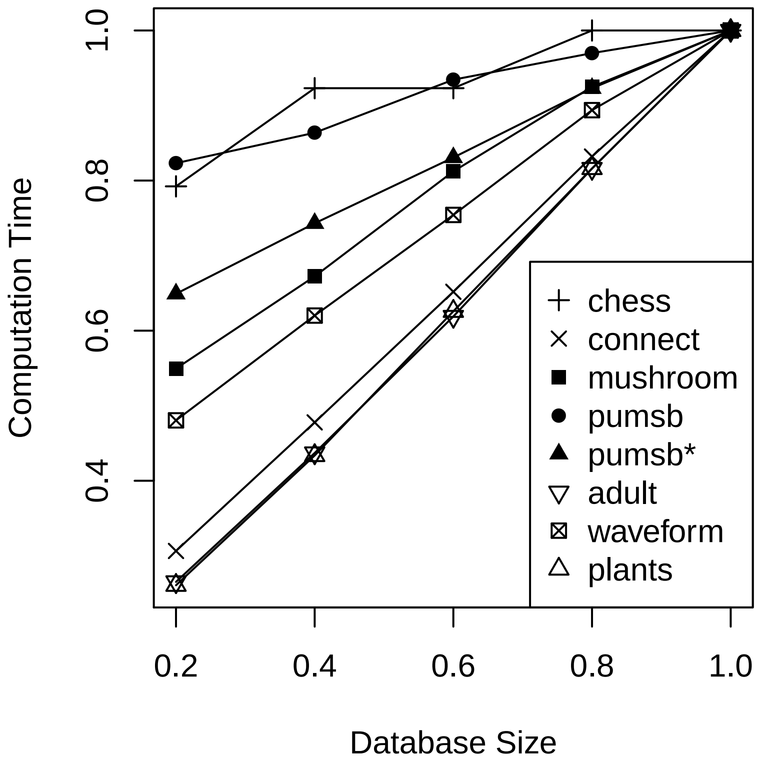


Figure 10.1: Scalability of Σοφια w.r.t. dataset size. X-axis shows the fraction of objects taken from an original dataset, and Y-axis shows the fraction of time w.r.t. the computational time needed for processing the original dataset.

polynomial time by adjusting the threshold θ of Δ-stable patterns.

Thus, in our first experiment we have checked which order is better for the attributes and how many patterns we can find for a given L. Table 10.3 shows the results and is divided into three parts corresponding to the order in which attributes were added to the context. Then all parts are divided into three subparts corresponding to a value of $L \in \{10^3, 10^4, 10^5\}$. Hence, we have 9 experiments and for every experiment we measure the computation time in seconds (t), the ratio of found patterns to L (#) and the final θ corresponding to the found patterns. For example, in the `Mushrooms` dataset, adding the attributes in the decreasing order of their support for $L = 10000$, the total computational time is equal to 2 seconds; the algorithm found around $0.87 * L = 8700$ patterns representing all patterns with Δ-measure higher than 49.

We can see that the computational time and the number of patterns for increasing order are never better than those of decreasing order and random order. Decreasing order and random order have nearly the same behavior, but in some cases the random order gives slightly worse results than the decreasing order. In fact, in the case of decreasing order we generate more patterns on earlier iterations of our algorithm, i.e., we have more chances to find an unstable pattern and filter it as earlier as possible. Since concepts are filtered earlier, we have more space for the computation, thus having smaller threshold θ and larger number of found patterns, and we should process less patterns, thus saving the computation time. We see that for the decreasing order of attributes the number of found patterns is always around or higher than $0.9 * L$, i.e., we find nearly as many patterns as the requested memory limit L.

Table 10.4: Compression with KRIMP based on Δ-stable itemsets and closed itemsets.

Dataset	Number of Candidates	Type	Time	Compression Rate
Mushrooms	96235	Δ	22	24.8%
	105430	Cls	2.2	25%
	9469	Δ	2.8	31.9%
	10006	Cls	0.2	44.4%
	718	Δ	0.4	46.7%
	728	Cls	0.1	67.6%
Chess	98528	Δ	19	42.9%
	99629	Cls	1	54.6%
	9639	Δ	2.1	46.6%
	9970	Cls	<0.1	66.8%
	928	Δ	0.3	54.6%
	999	Cls	<0.1	77.4%
Plants	94130	Δ	174	40.4%
	96819	Cls	7.5	49.6%
	9602	Δ	22	43.8%
	9923	Cls	3.4	55.8%
	1000	Δ	3	52.7%
	1006	Cls	0.9	63.6%

10.2.4 KRIMP and Δ-stable Itemsets

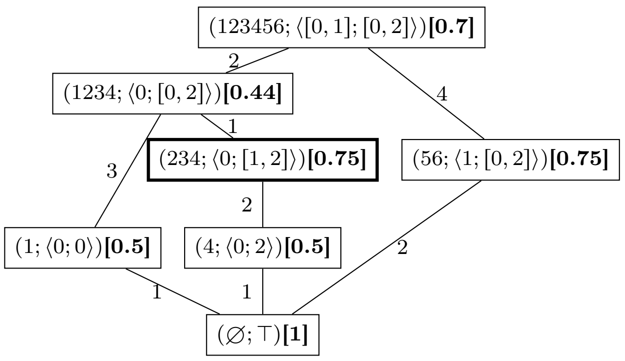
Finally we test how good are the patterns selected by Δ-measure by comparing them to closed patterns. For this we incorporate the patterns discovered by our algorithm to KRIMP (Vreeken, M. v. Leeuwen, et al. 2011), which was shown to be useful for both binary dataset compression and classification tasks. It takes a set of itemsets and a dataset and then it selects a subset of itemsets that are used for compression of the dataset. The optimization criterion for KRIMP is the smallest size of the selected itemsets plus the size of the compressed dataset. In KRIMP we can change the set of closed patterns to the set of patterns found by Σοφια algorithm. If the compression ratio increases, then the new set of patterns is more suitable for compression and classification tasks as KRIMP is known to provide very good classification results on real datasets.

The results of the experiment are shown in Table 10.4. We have experimented with the same datasets as in Section 10.2.3. For every dataset we generate a similar number of closed and Δ-stable patterns. We used three values of L ($L = 1000$, $L = 10^4$, and $L = 10^5$) and we adjusted the frequency threshold of closed patterns to find at least the same number of patterns. The type of the used patterns is shown in the column 'Type'. The patterns discovered by Σοφια algorithm are denoted by 'Δ' and closed patterns are denoted by 'Cls'. We also measured the computational time and the compressed ratio for every set of patterns. For example, the compression ratio of Mushrooms dataset by KRIMP based on 96235 most Δ-stable itemsets is 24.8%, i.e., it the dataset is compressed 4 times. The corresponding compression time is 22 seconds.

From our experiments we can see that the combination of KRIMP and Σοφια for Δ-stable patterns takes more time than the combination of KRIMP and closed patterns. However, the compression with Δ-patterns is always better than the compression with closed patterns. Moreover, in the case of Chess dataset the compression ratio for 1000 Δ-stable patterns is better than the compression ratio for 100000 most frequent closed patterns.

	m_1	m_2
g_1	0	0
g_2	0	$[1,2]$
g_3	0	$[1,2]$
g_4	0	2
g_5	1	$[0,2]$
g_6	1	$[0,2]$

(a) An interval context.



(b) An interval concept lattice with corresponding stability indexes.

Figure 10.2: A formal context and the corresponding lattice.

Thus, we conclude, that top Δ-stable patterns are more valuable than the top frequent closed patterns. Moreover, we can find them efficiently in polynomial time. Let us now switch to a more complicated pattern type.

10.3 Σοφια and Interval Tuple Data

10.3.1 Theoretical Background of Interval Tuple Data

Interval pattern structure

Interval pattern structures are introduced to support efficient processing of numerical data without binarization (Kaytoue, Kuznetsov, and Napoli 2011). Given k numerical or interval attributes whose values are of the form $[a,b]$, where $a,b \in \mathbb{R}$, the language of a pattern space is given by tuples of intervals of size k. For simplicity, we denote intervals of the form $[a,a]$ by a.

Figure 10.2a exemplifies an interval dataset. It contains 6 objects and 2 attributes. An interval as a value of an attribute corresponds to an uncertainty in the value of the attribute. For example, the value of m_1 for g_2 is known exactly, while the value of m_2 is lying in $[1,2]$. Given this intuition for intervals it is natural to define similarity of two intervals as their convex hull, since by adding new objects one increases the uncertainty. For example, for g_1 the value of m_1 is 0, while for g_6 it is 1, thus given the set $\{g_1, g_6\}$, the uncertainty of m_1 in this set is $[0,1]$, i.e., the similarity of g_1 and g_6 w.r.t. m_1 is $[0,1]$. More formally, given two intervals $[a,b]$ and $[c,d]$, the similarity of these two intervals is given by $[a,b] \sqcap [c,d] = [\min(a,c), \max(b,d)]$. Given a tuple of intervals, the similarity is computed component-wise. For example, $g_1^\diamond \sqcap g_6^\diamond = \langle[0,1];[0,2]\rangle$. Reciprocally, $\langle[0,1];[0,2]\rangle = \{g_1, g_2, \cdots, g_6\}$.

The resulting concept lattice is shown in Figure 10.2b. Concept extents are shown by indices of objects, intents are given in angle brackets, the numbers on edges and on concepts are related to interestingness of concepts.

Example 10.1. *Consider the example in Figure 10.2. The value of stability for every concept is given in square brackets. Every edge in the figure is labeled with the difference in support between the concepts this edge connects. Thus, Δ of a pattern is the minimum label of the edges going down from the concept.*

Projections of Interval Pattern Structures

Let us first consider interval pattern structures with only one attribute m. Let us denote by $W = \{w_1, \cdots, w_{|W|}\}$ all possible values of the left and right endpoints of the intervals corresponding to the attribute in a dataset, so that $w_1 < w_2 < \cdots < w_{|W|}$. By reducing the set W of possible values for the left or the right end of the interval we define a projection. For example, if $\{w_1\}$ is the only possible value for the left endpoint of an interval and $\{w_{|W|}\}$ is the only possible value of the right endpoint of an interval, then all interval patterns are projected to $[w_1, w_{|W|}]$. Let us consider this in more detail.

Let two sets $L, R \subset W$ such that $w_1 \in L$ and $w_{|W|} \in R$ be constraints on possible values on the left and right endpoints of an interval, respectively. Then a projection is defined as follows:

$$\psi_{m[L,R]}([a,b]) = [\max\{l \in L | l \leqslant a\}, \min\{r \in R | r \geqslant b\}] \,. \tag{10.1}$$

Requiring that $w_1 \in L$ and $w_{|W|} \in R$ we ensure that the sets used for minimal and maximal functions are not empty. It is not hard to see that (10.1) is a projection. The projections given by (10.1) are ordered w.r.t. simplicity (Definition 5.6 and 5.7). Indeed, given $L_1 \subseteq L$ and $R_1 \subseteq R$, we have $\psi_{m[L_1,R_1]} < \psi_{m[L,R]}$, because of inclusion of fixed sets. Let us notice that a projection $\psi_{m[W,W]}$ does not modify the lattice of concepts for the current dataset, since any interval for the value set W is possible. We also notice that a projection $\psi_{m[L,R]}$ is defined for one interval, while we can combine the projections for different attributes in a tuple to a single projection for the whole tuple $\psi_{m_1[L_1,R_1]m_2[L_2,R_2]\dots}$.

Example 10.2. *Consider example in Figure 10.2. Let us consider a projection*

$$\psi_{m_1[\{0,1\},\{1\}]m_2[\{0,2\},\{0,2\}]}.$$

The fixed set of this projection consists of $\{[0,1],1\} \times \{0,2,[0,2]\}$*, i.e., 6 intervals. Let us find the projection of* $(g_2)^\diamond = \langle 0; [1,2] \rangle$ *in a component-wise way:* $\psi_{m_1[\{0,1\},\{1\}]}(0) = [0,1]$*, since 0 is allowed on the left endpoint of an interval but not allowed to be on the right endpoint of an interval;* $\psi_{m_2[\{0,2\},\{0,2\}]}([1,2]) = [0,2]$ *since 1 is not allowed on the left endpoint of an interval. Thus,*

$$\psi_{m_1[\{0,1\},\{1\}]m_2[\{0,2\},\{0,2\}]}(\langle 0; [1,2] \rangle) = \langle [0,1]; [0,2] \rangle \,.$$

The lattice corresponding to this projection is shown in Figure 10.3.

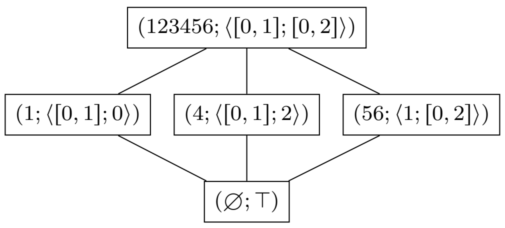


Figure 10.3: Projected lattice from example in Figure 10.2 by projection $\psi_{m_1[\{0,1\},\{1\}]m_2[\{0,2\},\{0,2\}]}$. See Example 10.2.

ϑ-Σοφια Algorithm for Interval Tuple Data

In order to demonstrate Σοφια for interval tuple data let us start from an example of a projection chain.

Example 10.3. *Let us construct a chain of projections satisfying (10.1) for the example in Figure 10.2. The value set for the first attribute is $W_1 = \{0,1\}$ and the value set for the second is $W_2 = \{0,1,2\}$. Let us start the chain from a projection $\psi_0 = \psi_{m_1[\{0\},\{1\}]m_2[\{0\},\{2\}]}$. This projection allows only for one pattern $\langle[0,1];[0,2]\rangle$, i.e., the concept lattice is easily found. Then we increase the complexity of a projection by allowing more patterns. For example, we can enrich the first component of a tuple without affecting the second one, i.e., a projection $\psi_1 = \psi_{m_1[\{0,1\},\{0,1\}]m_2[\{0\},\{2\}]}$. This projection allows for 3 patterns, i.e., any possible interval of the first component and only one interval [0,2] for the second component. Let us notice that it is not hard to find preimages for ψ_0 in $\psi_1(D)$. Indeed, for any pattern p from $\psi_0(D)$ one should just modify either the left side of the first interval of p by one value, or the right side of the first interval of p.*

Then we can introduce a projection that slightly enrich the second component of a tuple, e.g., $\psi_2 = \psi_{m_1[\{0,1\},\{0,1\}]m_2[\{0,1\},\{1,2\}]}$ and finally we have $\psi_3 = \psi_{m_1[W_1,W_1]m_2[W_2,W_2]}$. Finding preimages in this chain is not a hard problem, since on every set we can only slightly change left and/or right side of the second interval in a tuple. Thus, starting from a simple projection and making transitions from one projection to another, we can cut unpromising branches and efficiently find the set of interesting patterns.

Thus, here we consider a pattern structure $\mathbb{K} = (G,(D_I,\sqcap),\delta)$, where D_I is a semilattice of interval tuple descriptions. We say that every component of a tuple p corresponds to an attribute $m \in M$, where M is the set of interval attributes. Thus, the size of any tuple in D_I is $|M|$, and for any attribute $m \in M$ we can denote the corresponding interval by $m(p)$. We also denote the value set of m by W_m. Since the set W_m is totally ordered we also denote by $W_m^{(j)}$ and $W_m^{(-j)}$ the sets containing the first j (smallest) elements and the last j (largest) elements from W_m, respectively.

A projection chain for interval tuple data is formed in the same way as discussed in Example 10.3. We start from the projection containing only one pattern corresponding to the largest interval in each component, i.e., for an attribute m the projection is of the form $\psi_m[W_m^{(1)}, W_m^{(-1)}]$. Then to pass to a next projection, we select the attribute m, and for this attribute we extend the projection from $\psi_m[W_m^{(j)}, W_m^{(-j)}]$ to $\psi_m[W_m^{(j+1)}, W_m^{(-j-1)}]$. Thus, there are $k = \max_{m\in M}|W_m| \cdot |M|$ projections.

Finding preimages in this case is not hard, since to make a projection more detailed one should just extend the corresponding interval in left and/or on right end of the interval, i.e., there are only 4 possible preimages for a pattern when passing from one projection to another in this chain. Thus, the worst case complexity for ϑ-Σοφια algorithm for interval tuple data is

$$\mathbb{T}(\vartheta\text{-Σοφια}_{\text{intervals}}) = \max_{m\in M}|W_m| \cdot |M| \cdot \max_{0<i\leqslant k}|\mathcal{P}_i| \cdot \mathbb{T}(\mathcal{M}). \tag{10.2}$$

In particular, the complexity of Σοφια for interval data is $\max_{m\in M}|W_m| \cdot |M| \cdot L \cdot \mathbb{T}(\mathcal{M})$, i.e., it is polynomial modulo complexity of the measure.

10.3.2 Example of Δ-Stable Patterns in Interval Tuple Data

Let us consider the example in Figure 10.2 and show how we can find all Δ-stable patterns with a threshold $\theta = 2$. The chain of projections for this example is given in Example 10.3, it contains 4 projections:

$$\psi_0 = \psi_{m_1[\{0\},\{1\}]m_2[\{0\},\{2\}]} \qquad \psi_1 = \psi_{m_1[\{0,1\},\{0,1\}]m_2[\{0\},\{2\}]}$$

$$\psi_2 = \psi_{m_1[\{0,1\},\{0,1\}]m_2[\{0,1\},\{1,2\}]} \qquad \psi_3 = \psi_{m_1[\{0,1\},\{0,1\}]m_2[\{0,1,2\},\{0,1,2\}]}$$

Table 10.5: Patterns found for every projection in a chain for the example in Figure 10.2. Patterns are grey if they are removed for the corresponding projetion and they are labeled with "–" if they have not yet been found.

#	Pattern Ext.	Δ-measure			
		ψ_0	ψ_1	ψ_2	ψ_3
1	$\{g_1, g_2, g_3, g_4, g_5, g_6\}$	6	2	2	2
2	$\{g_1, g_2, g_3, g_4\}$	–	4	1	1
3	$\{g_5, g_6\}$	–	2	2	2
4	$\{g_1\}$	–	–	1	1
5	$\{g_2, g_3, g_4\}$	–	–	3	2
6	$\{g_4\}$	–	–	–	1

Since we are looking for closed patterns, every pattern can be identified by its extent. In Table 10.5 all patterns are given by their extents, i.e., by elements of D_C. For every pattern Δ-measure is shown for every ψ_i. A cell is shown in grey if the pattern is no more considered (the value of Δ less than 2). A cell has a dash "–", if a pattern in the row has not been generated for this projection.

For the example in Figure 10.2 the global process is as follows. At the beginning $\psi_0(D_I)$ contains only one element corresponding to pattern extent 123456 (a short cut for $\{g_1, g_2, g_3, g_4, g_5, g_6\}$) with a description $\langle[0,1];[0,2]\rangle$. Then, in $\psi_1(G,(D_I,\sqcap),\delta)$ possible preimages of 123456 are patterns with descriptions $\langle 0;[0,2]\rangle$ and $\langle 1;[0,2]\rangle$ given by pattern extents 1234 and 56, respectively. Then we continue with these three patterns which are all Δ-stable for the moment. The pattern extents 123456 and 56 have no preimages for the transition $\psi_1 \to \psi_2$, while the pattern extent 1234 has two preimages with descriptions $\langle 0;[0,1]\rangle$ and $\langle 0;[1,2]\rangle$ for this projection, which correspond to pattern extents 1 and 234. The first one is not Δ-stable and thus is no more considered. Moreover, the pattern extent 1234 is not Δ-stable (because of 234) and should also be removed. Finally, in transition $\psi_2 \to \psi_3$ only extent-pattern 234 has a preimage, a pattern extent 4, which is not Δ-stable. In such a way, we have started from a very simple projection ψ_0 and achieved the projection ψ_3 that gives us the Δ-stable patterns of the target pattern structure.

10.3.3 Processing of Interval Tuple Data

Recently it was also shown that it is more efficient to mine interval tuple data without binarization (Kaytoue, Kuznetsov, and Napoli 2011). In their paper the authors introduce algorithm `MinIntChange` for working directly with interval tuple data. Thus we compare ϑ-Σοφια and `MinIntChange` for finding Δ-stable patterns. We find Δ-stable concepts with ϑ-Σοφια and then adjust frequency threshold θ such that all Δ-stable patterns are among the frequent ones.

For interval tuple data stable patterns can be very deep in the search space, such that neither of the algorithms can find them quickly. Thus, we join some similar values for every attribute in an interval in the following way. Given a threshold $0 < \beta$, two consequent numbers w_i and w_{i+1} from a value set W are joined in the same interval if $w_{i+1} - w_i < \beta$. In order to properly set the threshold β, we use another threshold $0 < \gamma < 1$, which is much easier to set.

If we assume that the values of the attribute m are distributed around several states with centers $\tilde{w}^1, \cdots, \tilde{w}^l$, then it is natural to think that the difference between the closest centers $\texttt{abs}(\tilde{w}^i - \tilde{w}^{i\pm1})$ are much larger than the difference between the closest values. Ordering all values in the increasing order and finding the maximal difference $\delta_{\max}$ can give us an idea of typical distance between the states in the data. Thus, γ is defined as a proportion of this distance that should be considered as a distance between states, i.e., we put $\beta = \gamma \cdot \delta_{\max}$. If the distance between closest values in W are always the same, then even $\gamma = 0.99$ does not join values in intervals. However, if there are two states and the values are distributed very closely to one of these two states, then even $\gamma = 0.01$ can join values into one of two intervals corresponding to the states.

10.3.4 Evaluation on Real Data

Datasets

We take several datasets from the Bilkent University database [24]. The datasets are summarized in Table 10.6. The names of datasets are given by standard abbreviations used in the database of Bilkent University. For every dataset we provide the number of objects and attributes and the threshold γ for which the experiments are carried out. For example, database `EM` has 61 objects, 9 numeric attributes, and the threshold γ is set to 0.3. Categorical attributes and rows with missing values, if any, are removed from the datasets.

Results

Table 10.6: Runtime in seconds of Σοφια and `MinIntChange` for different datasets.

DS	# Objs	# Attrs	γ	Δ	# Ptrns	θ	$t_{\Sigma o\varphi\iota\alpha}$	$t_{\texttt{MIC}}$
EM	61	9	0.3	3	3	21	< 0.1	57
BK	96	4	0.3	4	50	46	< 0.1	11
CN	105	20	0.8	2	5362	30	2.4	28 ▨
CU	108	5	0.3	5	4	27	< 0.1	1.5
FF	125	3	0.3	6	3	48	< 0.1	1
AP	135	4	0.01	5	1	19	< 0.1	34
EL	211	12	0.3	6	33	83	< 0.1	34 ▨
BA	337	16	0.5	4	736	91	1.5	32 ▨
AU	398	7	0.3	7	17	234	0.7	73 ▨
HO	506	13	0.8	10	1	340	0.7	57 ▨
QU	2178	25	0.3	40	1	659	1.3	28 ▨
AB	4177	8	0.3	46	3	1400	11	86 ▨
CA	8192	21	0.3	85	6	2568	112	24 ▨
PT	9065	48	0.3	2	1	2	45	14 ▨

In Table 10.6 we show the computation time for finding the best Δ-stable pattern (or patterns if they have the same value for Δ-measure) for ϑ-Σοφια and for `MinIntChange`. The last algorithm is abbreviated as `MIC`. Since `MinIntChange` algorithm sometimes produces too many patterns, i.e., we do not have enough memory in our computer to check all of them, we interrupt the procedure and show the corresponding time in grey. We also show the number of the best patterns and the corresponding threshold Δ. The support threshold θ for finding the best Δ-stable patterns is also shown. For example, dataset `CN` contains 5362 best Δ-stable patterns, all having a Δ of 2. To find all these patterns with a postfiltering, we should mine frequent patterns with a support

[24] http://funapp.cs.bilkent.edu.tr/DataSets/

threshold lower than 30 or $\frac{30}{105} = 30\%$. ϑ-Σοφια computes all these patterns in 2.4 seconds, while MIC requires at least 28 seconds and the procedure was interrupted without continuation.

As we can see, ϑ-Σοφια is significantly faster than MIC in all datasets. In the two datasets CA and PT, MIC was stopped before computing all patterns and the runtime did not exceed the runtime of ϑ-Σοφια. However, in both cases, MIC achieved less than 10% of the required operations.

10.4 Conclusion

In this chapter we have evaluated Σοφια on two types of data: binary data and interval tuple data. We have explained how Σοφια is instantiated for both types of data. The experimental evaluation shows, that Σοφια is able to efficiently mine Δ-stable patterns in both types of data. It is significantly more computationally efficient than the competitors. We have found that the order of projections is important for the efficiency of Σοφια. The order of decreasing support of attributes seems to be the best one. We have also found that Δ-stable patterns can be combined with KRIMP in order to improve compression of the datasets.

There are two main directions for further experimental research of Σοφια. First of all, one should figure out what is the best order of attributes and more general the best chain of projections for every data type. Second, in this chapter we have only discussed two types of data, but other types of pattern structures can be also processed and hence should be evaluated.

Conclusion and Perspectives

Summary

Mining valuable information from structured data could help to solve many practical problems or to lead to a breakthrough in fundamental science. We have approached the problem of mining structured data by means of formal concept analysis (FCA), a mathematical framework having many applications in data mining and knowledge representation.

First we have shown how structured data, in particular a dataset of molecules, can be converted to a binary object-attribute representation called formal context. However, for dealing with a large dataset we have faced a problem on how to select the best pieces of information, called formal concepts, from the whole set of possibilities. Stability is a well-founded measure of formal concepts having the following intuition. If some objects are removed from the data what is the probability that the concept is preserved. However, stability computation is #P-hard task. Correspondingly, we introduced an estimate of stability that can be computed in polynomial time, the associated measure of concept relevancy is called Δ-measure. Furthermore, we have studied the behavior of stability and Δ-measure and have shown that it is a good measure for selecting concepts. Finally we have applied stability for analysis of chemical compounds associated with mutagenicity. Stable concepts have revealed some alerts of mutagenicity that are further studied by chemists.

Later we have dealt with structured data by means of pattern structures, an extension of FCA allowing for direct encoding of structured data, i.e., without a conversion to formal contexts. In order to do this efficiently, we have introduced o-projections of pattern structures, an original extension of the formalism of projections, allowing for simplification of the structured data by removing mostly irrelevant concepts. This extension has been applied for correcting RDF data and for analysis of patient hospitalization history. Thanks to reuse of heterogeneous pattern structures we were able to find association rules in the data with high confidence, we have shown that such kind of data in many cases can be converted to implications which means that certain missing triples of RDF data should be added. For patient hospitalization history patterns structures were used to represent and analyze trajectories of different patient. The largest problem was the heterogeneity of descriptions for single hospitalizations that are joined to sequences describing trajectories. Thanks to o-projections, we were able to process this complex and reach data by removing mostly irrelevant concepts. The found concepts can have a different level of granularity for the taxonomies involved into the descriptions of hospitalization in contrast to other methods addressing towards analysis of this kind of data.

Finally, since even after applying all o-projections the number of concepts can be huge, we have introduced an original and very efficient approach, called $\Sigma o\varphi\iota\alpha$, for mining stable and Δ-stable concepts directly. It allows us to process even very huge datasets, where the the whole set of concepts cannot be generated, without introducing aggressive projections that could possibly remove some important peaces of information. We have discussed this algorithm for mining itemset (binary) and interval tuple data and its computational performance has been experimentally proved.

Perspectives

In this subsection we summarize the perspectives of our work. Here we provide mostly the perspectives of the presented approach of mining structured data rather than the perspectives of applications. The last ones can be found in the conclusions of the corresponding chapters. The text below is divided into three subsections corresponding to perspectives of every part of this manuscript.

Study on stability and other measures. In the first part we have studied stability and applied it for mining a chemical dataset. This study of stability can be extended in two different ways. First of all, an approach introduced in Chapter 2 can be applied also for other measures. In order to do so, we will need to introduce a certain quantative criteria that measures the quality of the result of this approach. Second, although we have carried out a comparison of stability with some state of the art interestingness measures in Section 3, the number of measures involved into this study is quite small and, thus, should be extended to a wider set of interestingness measures. Finally, from application point of view, since it works good for chemical domain, it is interesting to apply stability to other domains where it could probably lead to interesting results.

Study on pattern structures and o-projections. Our original extension of pattern structure projections was shown to be essential in applications. It can be further extended to so-called transformations that instead of taking a suborder of the semilattice of descriptions, change one semilattice to another one. This is important because then we can describe a change from semilattice of descriptions to a semilattice of interval tuples, which could probably allow us to explain SVM in terms of projections, a well-known approach to supervised classification. From application point of view, o-projections are constructed now in an ad-hoc manner, it is interesting to know if it can be done (semi-)automatically. Finally, pattern structures and o-projections are a powerful tool for mining structured data, thus, their application to different domains is of high interest.

Study on direct mining of stable concepts. In the third part we have introduced algorithm $\Sigma o\varphi\iota\alpha$ for direct mining of Δ-stable concepts. This work is at the beginning and thus reveals a large number of possibilities for further extensions. First of all, the presented work was only applied for binary and interval-tuple data. It is of high interest to adapt $\Sigma o\varphi\iota\alpha$ to other kind of data, e.g., sequences or graphs. It faces some problems and one of the most important is how one should extend a set of graphs (or sequences) in a canonical way. For the moment only canonical extensions of single graphs (or sequences) are studied. Another important study of $\Sigma o\varphi\iota\alpha$ is that it can be applied to a wide set of measures, i.e., a set of measures anti-monotonic w.r.t. a chain of projections. *What measures belong to this class?* It is an important question that should be addressed. The next important point of this work is an efficient implementation of the algorithm. Indeed, our current implementation does not rely on most of the modern techniques for mining frequent patterns, although they do not contradict to the idea of $\Sigma o\varphi\iota\alpha$. Furthermore, $\Sigma o\varphi\iota\alpha$ can be parallelized providing a more efficient implementation.

Appendix A

cMet Dataset Details

Table A.1: Available c-Met X-Ray structures up to now

(a) From PDB

	PDB codes	Ligand codes
1	1R0P	K-252a
2	2RFN	AM7
3	2RFS	AM8
4	2WDI	ZZY
5	2WGJ	VGH
6	2WKM	PFY
7	3A4P	DFQ
8	3CIX	CKK
9	3CCN	LKG
10	3CD8	L5G
11	3CE3	1FN
12	3CTH	319
13	3CTJ	320
14	3DK[FG]	SX8
15	3DKC	ATP
16	3EFJ	MT3
17	3EFK	MT4
18	3F66	IHX
19	3F82	353
20	3I5N	B2D
21	3L8V	L8V
22	3LQ8	88Z
23	3Q6W	Q6W
24	3QTI	3QT
25	3R7O	M61
26	3RHK	M97
27	3ZXZ	KRW
28	3ZZE	6XP

(b) From patent EP/2002/1243596

	Name	Ligand codes
29	AGOU	agou-short
30	AGOU	agou-long

Table A.2: Details on principal and secondary functional groups described by Moldb5r and included in our database

Principal groups	Secondary groups
Alcohol	1,2-Aminoalcohol
	1,2-diol
	Primary Alcohol
	Secondary Alcohol
	Tertiary Alcohol
Alkene	
Alkyne	
Amine	Primary Aliphatic Amine
	Secondary Aliphatic Amine
	Tertiary Aliphatic Amine
	Primary Amine
	Secondary Amine
	Tertiary Amine
	Primary Aromatic Amine
	Secondary Aromatic Amine
	Secondary Mixed Amine
Aromatic	
Carbonyl compound	Ketone
	Oxohetarene
Carboxylic Acid derivative	Carboxylic Acid salt
	Carboxylic Acid esther
	Carboxylic Acid Amide
	Primary Carboxylic Acid Amide
	Secondary Carboxylic Acid Amide
	Tertiary Carboxylic Acid Amide
	Carboxylic Acid Amidine
	Carboxylic Acid Hydrazide
	Carboxylic Acid Imide
	Carboxylic Acid Imide (N-unsubstituted)
	Lactam
	Nitrile
C02 derivative	Thiourea
	Urea
Ether	Alkyl Aryl Ether
	Dialkyl Ether
	Diaryl Ether
Halogen	Aryl Fluoride
	Alkyl Halide
	Alkyl Fluoride
	Aryl Bromide
	Aryl Chloride
	Aryl Halide
Heterocycle	
Hydrazine	
Nitro Compound	
Hydroxy compound	Phenol
Phosphoric Acid derivative	Phosphoric Acid esther
Sulfonamide	
Sulfone	
Sulfonic Acid derivative	
Sulfoxyide	
Sulfuric Acid derivative	Sulfuric Acid Diamide
Thioether	

Index

Index

Bibliography

Aalst, Wil M. P. van der (2011). *Process mining: Discovery, conformance and enhancement of business processes.* Springer, pp. I–XVI, 1–352.

Adda, Mehdi, Petko Valtchev, Rokia Missaoui, and Chabane Djeraba (2010). "A framework for mining meaningful usage patterns within a semantically enhanced web portal". In: *Proc. 3rd C* Conf. Comput. Sci. Softw. Eng.* C3S2E '10. New York, NY, USA: ACM, pp. 138–147.

ADMET and Predictive Toxicology. on line publication.

Aggarwal, Charu C, Mansurul A Bhuiyan, and Mohammad Al Hasan (2014). "Frequent Pattern Mining Algorithms: A Survey". In: *Freq. Pattern Min.* Ed. by Charu C Aggarwal and Jiawei Han. Springer International Publishing, pp. 19–64.

Agrawal, Rakesh, Tomasz Imielinski, and Arun Swami (1993a). "Database mining: a performance perspective". In: *Knowl. Data Eng. IEEE Trans.* 5.6, pp. 914–925.

— (1993b). "Mining association rules between sets of items in large databases". In: *ACM SIGMOD Rec.* Vol. 22. 2. ACM, pp. 207–216.

Agrawal, Rakesh and Ramakrishnan Srikant (1994). "Fast algorithms for mining association rules". In: *Proc. 20th int. conf. very large data bases, VLDB.* Vol. 1215, pp. 487–499.

— (1995). "Mining sequential patterns". In: *Data Eng. 1995. Proc. Elev. Int. Conf.* Pp. 3–14.

Ahlberg, Ernst, Lars Carlsson, and Scott Boyer (2014). "Computational Derivation of Structural Alerts from Large Toxicology Data Sets". In: *J. Chem. Inf. Model.* 54.10, pp. 2945–2952.

Alam, Mehwish, Aleksey Buzmakov, Victor Codocedo, and Amedeo Napoli (2015). "Mining Definitions from RDF Annotations Using Formal Concept Analysis". In: *Proc. Twenty-Fourth Int. Jt. Conf. Artif. Intell. IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pp. 823–829.

Ames, Bruce N, Frank D Lee, and William E Durston (1973). "An Improved Bacterial Test System for the Detection and Classification of Mutagens and Carcinogens". In: *Proc. Natl. Acad. Sci. U.S.A.* 70.3, pp. 782–786.

Ashby, J and R W Tennant (1991). "Definitive Relationships among Chemical Structure, Carcinogenicity and Mutagenicity for 301 Chemicals Tested by the U.S. NTP". In: *Mutat. Res.* 257.3, pp. 229–306.

Asses, Yasmine, Aleksey Buzmakov, Thomas Bourquard, Sergei O. Kuznetsov, Amedeo Napoli, et al. (2012). "A Hybrid Classification Approach based on FCA and Emerging Patterns-An application for the classification of biological inhibitors". In: *Proc. 9th Int. Conf. Concept Lattices Their Appl.* Pp. 211–222.

Asses, Yasmine, Vincent Leroux, Safia Tairi-Kellou, Rosanna Dono, Flavio Maina, and Bernard Maigret (2009). "Analysis of c-Met Kinase Domain Complexes: A New Specific Catalytic Site Receptor Model for Defining Binding Modes of ATP-Competitive Ligands". In: *Chem. Biol. Drug Des.* 74.6, pp. 560–570.

Asses, Yasmine, Vishwesh Venkatraman, Vincent Leroux, David W Ritchie, and Bernard Maigret (2012). “Exploring c-Met kinase flexibility by sampling and clustering its conformational space”. In: *Proteins Struct. Funct. Bioinforma.* 80.4, pp. 1227–1238.

Auer, J and J Bajorath (2006). “Emerging Chemical Patterns: a New Methodology for Molecular Classification and Compound Selection”. In: *J. Chem. Inf. Model.* 46.6, pp. 2502–2514.

Ayres, Jay, Jason Flannick, Johannes Gehrke, and Tomi Yiu (2002). “Sequential PAttern mining using a bitmap representation”. In: *KDD*, pp. 429–435.

Azevedo, Paulo J. and Alípio M. Jorge (2007). “Comparing Rule Measures for Predictive Association Rules”. In: *Mach. Learn. ECML 2007.* Ed. by Joost N. Kok, Jacek Koronacki, Ramon Lopez de Mantaras, Stan Matwin, Dunja Mladenič, and Andrzej Skowron. Vol. 4701. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 510–517.

Babin, Mikhail A. and Sergei O. Kuznetsov (2012). “Approximating Concept Stability”. In: *Form. Concept Anal.* Ed. by Florent Domenach, DmitryI. Ignatov, and Jonas Poelmans. Vol. 7278. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 7–15.

Balcázar, José L, Albert Bifet, and Antoni Lozano (2006). “Intersection Algorithms and a Closure Operator on Unordered Trees”. In: *MLG*, p. 1.

Barbut, Marc and Bernard Monjardet (1970). *Ordre et classification algèbre et combinatoirs.* French. Hachette.

Bêlohlávek, Radim (1999). “Fuzzy Galois Connections”. In: *Math. Log. Q.* 45.4, pp. 497–504.

Bělohlávek, Radim and Vladimír Sklenář (2005). “Formal Concept Analysis Constrained by Attribute-Dependency Formulas”. In: *Form. Concept Anal.* Ed. by Bernhard Ganter and Robert Godin. Vol. 3403. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 176–191.

Belohlavek, Radim and Martin Trnecka (2013). “Basic Level in Formal Concept Analysis: Interesting Concepts and Psychological Ramifications”. In: *Proc. Twenty-Third Int. Jt. Conf. Artif. Intell.* IJCAI’13. AAAI Press, pp. 1233–1239.

Belohlavek, Radim and Vilém Vychodil (2009). “Formal Concept Analysis With Background Knowledge: Attribute Priorities”. In: *IEEE Trans. Syst. Man, Cybern. Part C (Applications Rev.* 39.4, pp. 399–409.

Bělohlávek, Radim and Vilém Vychodil (2006). “Formal Concept Analysis with Constraints by Closure Operators”. In: *Concept. Struct. Inspir. Appl.* Ed. by Henrik Schärfe, Pascal Hitzler, and Peter Ohrstrom. Vol. 4068. Lecture Notes in Computer Science 1. Springer Berlin Heidelberg, pp. 131–143.

Benigni, R (2008). *The Benigni/Bossa Rulebase for Mutagenicity and Carcinogenicity – a Module of Toxtree.* Tech. rep. EUR 23241. JRC Sci. Tech. Reports, pp. 1–70.

Benigni, R and C Bossa (2011). “Mechanisms of Chemical Carcinogenicity and Mutagenicity: a Review with Implications for Predictive Toxicology”. In: *Chem. Rev.* 111.4, pp. 2507–2536.

Benz, Dominik, Andreas Hotho, and Gerd Stumme (2010). “Semantics made by you and me: Self-emerging ontologies can capture the diversity of shared knowledge”. In: *Proceedings of the 2nd Web Science Conference.*

Besson, Jérémy, Ruggero G. Pensa, Céline Robardet, and Jean-François Boulicaut (2006). “Constraint-Based Mining of Fault-Tolerant Patterns from Boolean Data”. In: *Knowl. Discov. Inductive Databases.* Ed. by Francesco Bonchi and Jean-François Boulicaut. Vol. 3933. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 55–71.

Besson, Jérémy, Céline Robardet, and Jean-François Boulicaut (2005). “Mining Formal Concepts with a Bounded Number of Exceptions from Transactional Data”. In: *Knowl. Discov. Inductive Databases.* Ed. by Bart Goethals and Arno Siebes. Vol. 3377. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 33–45.

Bissell-Siders, Ryan, Bertrand Cuissart, and Bruno Crémilleux (2010). “On the Stimulation of Patterns - Definitions, Calculation Method and First Usages”. In: *18th Int. Conf. Concept. Struct. ICCS 2010, Proc.* Pp. 56–69.

Bizer, Christian, Tom Heath, and Tim Berners-Lee (2009). “Linked Data - The Story So Far”. In: *Int. J. Semantic Web Inf. Syst.* 5.3, pp. 1–22.

Björne, Jari, Antti Airola, Tapio Pahikkala, and Tapio Salakoski (2011). “Drug-drug interaction extraction from biomedical texts with svm and rls classifiers”. In: *Proceedings of DDIExtraction-2011 challenge task*, pp. 35–42.

Blinova, V. G., D. A. Dobrynin, V. K. Finn, Sergei O. Kuznetsov, and E. S. Pankratova (2003). “Toxicology analysis by means of the JSM-method”. In: *Bioinformatics* 19.10, pp. 1201–1207.

Boldyrev, N. G. (1974). “Minimization of Boolean Partial Functions with a Large Number of "Don’t Care" Conditions and the Problem of Feature Extraction”. In: *Proc. Int. Symp. "Discrete Syst.* Riga, Latvia: Publishing house "ZINATNE", pp. 101–109.

Boley, Mario, Tamás Horváth, Axel Poigné, and Stefan Wrobel (2010). “Listing closed sets of strongly accessible set systems with applications to data mining”. In: *Theor. Comput. Sci.* 411.3, pp. 691–700.

Borgelt, Christian (2006). “Combining Ring Extensions and Canonical Form Pruning”. In: *Work. Min. Learn. with Graphs*, pp. 109–116.

Borgelt, Christian and M.R. Berthold (2002). “Mining molecular fragments: finding relevant substructures of molecules”. In: *2002 IEEE Int. Conf. Data Mining, 2002. Proceedings.* IEEE Comput. Soc, pp. 51–58.

Boulicaut, Jean-François, Artur Bykowski, and Christophe Rigotti (2000). “Approximation of Frequency Queries by Means of Free-Sets”. In: *Princ. Data Min. Knowl. Discov.* Ed. by DjamelA. Zighed, Jan Komorowski, and Jan Żytkow. Vol. 1910. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 75–85.

Burdick, Doug, Manuel Calimlim, and Johannes Gehrke (2001). “MAFIA: A maximal frequent itemset algorithm for transactional databases”. In: *Data Eng. 2001. Proceedings. 17th Int. Conf.* IEEE, pp. 443–452.

Buzmakov, Aleksey, Elias Egho, Nicolas Jay, Sergei O. Kuznetsov, Amedeo Napoli, and Chedy Raïssi (2013a). “FCA and pattern structures for mining care trajectories”. In: *Work. Notes FCA4AI*, pp. 7–14.

— (2013b). “On Projections of Sequential Pattern Structures (with an application on care trajectories)”. In: *Proc. 10th Int. Conf. Concept Lattices Their Appl.* La Rochelle, France, pp. 199–208.

— (2013c). “The representation of sequential patterns and their projections within Formal Concept Analysis”. In: *Work. Notes LML*, pp. 65–79.

— (2015a). “On Mining Complex Sequential Data by Means of FCA and Pattern Structures”. In: *Int. J. Gen. Syst.* IN PRESS. arXiv: 1504.02255.

Buzmakov, Aleksey, Sergei O. Kuznetsov, and Amedeo Napoli (2014a). “Concept Stability as a Tool for Pattern Selection”. In: *Work. Notes FCA4AI*, pp. 51–58.

— (2014b). “Is Concept Stability a Measure for Pattern Selection?” In: *Procedia Comput. Sci.* 31, pp. 918–927.

— (2014c). “On Evaluating Interestingness Measures for Closed Itemsets”. In: *STAIRS 2014*. Vol. 264. Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 71–80.

— (2014d). “Scalable Estimates of Concept Stability”. In: *Form. Concept Anal.* Ed. by Christian Sacarea, Cynthia Vera Glodeanu, and Mehdi Kaytoue. Vol. 8478. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 161–176.

Buzmakov, Aleksey, Sergei O. Kuznetsov, and Amedeo Napoli (2015b). "Fast Generation of Best Interval Patterns for Nonmonotonic Constraints". In: *Mach. Learn. Knowl. Discov. Databases.* Ed. by Annalisa Appice, Pedro Pereira Rodrigues, Vítor Santos Costa, João Gama, Alípio Jorge, and Carlos Soares. Vol. 9285. Lecture Notes in Computer Science. Springer International Publishing, pp. 157–172.

— (2015c). "Revisiting Pattern Structure Projections". In: *Form. Concept Anal.* Ed. by Jaume Baixeries, Christian Sacarea, and Manuel Ojeda-Aciego. Vol. 9113. Lecture Notes in Computer Science. Springer International Publishing, pp. 200–215.

Calders, Toon and Bart Goethals (2002). "Mining All Non-derivable Frequent Itemsets". In: *Princ. Data Min. Knowl. Discov.* Ed. by Tapio Elomaa, Heikki Mannila, and Hannu Toivonen. Vol. 2431. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 74–86.

Calders, Toon, Christophe Rigotti, and Jean-François Boulicaut (2006). "A Survey on Condensed Representations for Frequent Sets". In: *Constraint-Based Min. Inductive Databases.* Ed. by Jean-François Boulicaut, Luc De Raedt, and Heikki Mannila. Vol. 3848. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 64–80.

CancerInformatics.org.uk. *Chemical Carcinogenesis Research Information System.*

Cao, Jie, Zhiang Wu, and Junjie Wu (2014). "Scaling up cosine interesting pattern discovery: A depth-first method". In: *Inf. Sci. (Ny).* 266, pp. 31–46.

Carhart, Raymond E, Dennis H Smith, and R Venkataraghavan (1985). "Atom pairs as molecular features in structure-activity studies: definition and applications". In: *J. Chem. Inf. Comput. Sci.* 25.2, pp. 64–73.

Carvalho, Deborah R., Alex A. Freitas, and Nelson Ebecken (2005). "Evaluating the Correlation Between Objective Rule Interestingness Measures and Real Human Interest". In: *Knowl. Discov. Databases PKDD 2005.* Ed. by Alípio Mário Jorge, Luís Torgo, Pavel Brazdil, Rui Camacho, and João Gama. Vol. 3721. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 453–461.

Casas-Garriga, Gemma (2005). "Summarizing Sequential Data with Closed Partial Orders." In: *Proc. 5th SIAM Int'l Conf. Data Min.*

Cellier, Peggy, Sébastien Ferré, Olivier Ridoux, and Mireille Ducasse (2008). "A parameterized algorithm to explore formal contexts with a taxonomy". In: *Int. J. Found. Comput. Sci.* 19.02, pp. 319–343.

Chiu, Ding-Ying, Yi-Hung Wu, and Arbee L. P. Chen (2004). "An Efficient Algorithm for Mining Frequent Sequences by a New Strategy without Support Counting". In: *ICDE*, pp. 375–386.

Chowdhury, Faisal Mahbub, Asma Ben Abacha, Alberto Lavelli, and Pierre Zweigenbaum (2011). "Two different machine learning techniques for drug-drug interaction extraction". In: *Challenge Task on Drug-Drug Interaction Extraction*, pp. 19–26.

Chowdhury, Md Faisal Mahbub and Alberto Lavelli (2011). "Drug-drug interaction extraction using composite kernels". In: *Challenge Task on Drug-Drug Interaction Extraction*, pp. 27–33.

Codocedo, Víctor and Amedeo Napoli (2014). "A Proposition for Combining Pattern Structures and Relational Concept Analysis". In: *12th International Conference on Formal Concept Analysis.*

Coenen, F. (2003). *The LUCS-KDD Discretised and normalised ARM and CARM Data Library*[25]. Department of Computer Science, The University of Liverpool, UK.

Commission, European (2007). *REACH: Registration, Evaluation, Authorisation and Restriction of Chemicals.* on line publication.

[25] http://www.csc.liv.ac.uk/~frans/KDD/Software/LUCS_KDD_DN/

Cook, Diane J. and Lawrence B. Holder (1994). “Substructure discovery using minimum description length and background knowledge”. In: *J. Artif. Intell. Res.* 1, pp. 231–255.

Coquin, Laurence, Steven J. Canipa, William C. Drewe, Lilia Fisk, Valerie J. Gillet, Mukesh Patel, Jeffrey Plante, Richard J Sherhod, and Jonathan D Vessey (2015). “New structural alerts for Ames mutagenicity discovered using emerging pattern mining techniques”. In: *Toxicol. Res.* 4, pp. 46–56.

Coulet, Adrien, Florent Domenach, Mehdi Kaytoue, and Amedeo Napoli (2013). “Using Pattern Structures for Analyzing Ontology-Based Annotations of Biomedical Data”. In: *Form. Concept Anal. SE - 5.* Ed. by Peggy Cellier, Felix Distel, and Bernhard Ganter. Vol. 7880. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 76–91.

Cuissart, Bertrand, Guillaume Poezevara, Bruno Crémilleux, Alban Lepailleur, and Ronan Bureau (2013). “Emerging Patterns as Structural Alerts for Computational Toxicology”. In: *Contrast Data Min.* Ed. by Guozhu Dong and James Bailey. CRC Press, pp. 269–282.

Deshpande, Makund, Michihiro Kuramochi, Nikil Wale, and Gearge Karapis (2005). “Frequent substructure-based approaches for classifying chemical compounds”. In: *IEEE Trans. Knowl. Data Eng.* 17.8, pp. 1036–1050.

Devillers, J. and A. T. Balaban (1999). *Topological Indices and Related Descrip- tors in QSAR and QSPR.* Ed. by Boca Raton. FL: CRC Press.

Dias, Sérgio M. and Newton J. Vieira (2013). “Applying the JBOS reduction method for relevant knowledge extraction”. In: *Expert Syst. Appl.* 40.5, pp. 1880–1887.

Ding, Bolin, David Lo, Jiawei Han, and Siau-Cheng Khoo (2009). “Efficient Mining of Closed Repetitive Gapped Subsequences from a Sequence Database”. In: *Proc. IEEE 25th Int. Conf. Data Eng.* IEEE, pp. 1024–1035.

Dong, Guozhu and James Bailey, eds. (2013). *Contrast Data Mining: Concepts, Algorithms, and Applications.* CRC Press.

Dong, Guozhu and Jinyan Li (1999). “Efficient mining of emerging patterns: Discovering trends and differences”. In: *Proc. fifth ACM SIGKDD Int. Conf. Knowl. Discov. data Min.* KDD ’99. New York: ACM, pp. 43–52.

Dussault, Isabelle and Steven F Bellon (2008). “c-Met inhibitors with different binding modes: Two is better than one”. In: *CellCycle* 7, pp. 1157–1160.

Eathiraj, Sudharshan, Rocio Palma, Erika Volckova, Marscha Hirschi, Dennis S France, Mark A Ashwell, and Thomas C K Chan (2011). “Discovery of a Novel Mode of Protein Kinase Inhibition Characterized by the Mechanism of Inhibition of Human Mesenchymal-epithelial Transition Factor (c-Met) Protein Autophosphorylation by ARQ 197”. In: *J. Biol. Chem.* 286.23, pp. 20666–20676.

Egho, Elias, Nicolas Jay, Chedy Raïssi, Dino Ienco, Pascal Poncelet, Maguelonne Teisseire, and Amedeo Napoli (2014). “A contribution to the discovery of multidimensional patterns in healthcare trajectories”. In: *J. Intell. Inf. Syst.* 42.2, pp. 283–305.

Egho, Elias, Chedy Raïssi, Nicolas Jay, and Amedeo Napoli (2014). “Mining Heterogeneous Multidimensional Sequential Patterns”. In: *ECAI 2014 - 21st Eur. Conf. Artif. Intell.* Pp. 279–284.

Faulon, J L, C J Churchwell, and D P Visco (2003). “The signature molecular descriptor. 2. Enumerating molecules from their extended valence sequences”. In: *J. Chem. Inf. Comput. Sci.* 43.3, pp. 721–734.

Faulon, J L, D P Visco, and R S Pophale (2003). “The signature molecular descriptor. 1. Using extended valence sequences in QSAR and QSPR studies”. In: *J. Chem. Inf. Comput. Sci.* 43.3, pp. 707–720.

Fayyad, Usama, Gregory Piatetsky-Shapiro, and Padhraic Smyth (1996). “From data mining to knowledge discovery in databases”. In: *AI Mag.* 17.3, p. 37.

Fda.gov. *Genetic Toxicity, Reproductive and Developmental Toxicity, and Carcinogenicity Database.*

Feng, Jun, Laura Lurati, Haojun Ouyang, Tracy Robinson, Yuanyuan Wang, Shenglan Yuan, and S Stanley Young (2003). “Predictive Toxicology: Benchmarking Molecular Descriptors and Statistical Methods”. In: *J. Chem. Inf. Comput. Sci.* 43.5, pp. 1463–1470.

Ferré, Sébastien (2007). “The Efficient Computation of Complete and Concise Substring Scales with Suffix Trees”. In: *Form. Concept Anal. SE - 7.* Ed. by Sergei O. Kuznetsov and Stefan Schmidt. Vol. 4390. Lecture Notes in Computer Science. Springer, pp. 98–113.

Ferré, Sébastien and Olivier Ridoux (2002). “The Use of Associative Concepts in the Incremental Building of a Logical Context”. English. In: *Concept. Struct. Integr. Interfaces SE - 23.* Ed. by Uta Priss, Dan Corbett, and Galia Angelova. Vol. 2393. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 299–313.

Fetter, Robert B., Youngsoo Shin, Jean L. Freeman, Richard F. Averill, and John D. Thompson (1980). “Case mix definition by diagnosis-related groups.” In: *Med Care* 18.2, pp. 1–53.

Finn, V. K. (1991). “Plausible reasoning in systems of JSM type”. In: *Itogi Nauk. i Tekhniki, Seriya Inform.* 15, pp. 54–101.

Frank, A. and A. Asuncion (2010). *UCI Machine Learning Repository [http://archive.ics.uci.edu/ml].* University of California, Irvine, School of Information and Computer Sciences.

Fürber, Christian and Martin Hepp (2011). “Swiqa - a semantic web information quality assessment framework”. In: *19th European Conference on Information Systems.*

Galitsky, Boris A., Dmitry Ilvovsky, Sergei O. Kuznetsov, and Fedor Strok (2014). “Finding Maximal Common Sub-parse Thickets for Multi-sentence Search”. In: *Graph Struct. Knowl. Represent. Reason.* Ed. by Madalina Croitoru, Sebastian Rudolph, Stefan Woltran, and Christophe Gonzales. Vol. 8323. Lecture Notes in Computer Science. Springer International Publishing, pp. 39–57.

Galitsky, Boris A., Sergei O. Kuznetsov, and Daniel Usikov (2013). “Parse Thicket Representation for Multi-sentence Search”. In: *Concept. Struct. STEM Res. Educ.* Ed. by HeatherD. Pfeiffer, DmitryI. Ignatov, Jonas Poelmans, and Nagarjuna Gadiraju. Vol. 7735. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 153–172.

Ganter, Bernhard (1984). “Two Basic Algorithms in Concept Analysis”. In: *Form. Concept Anal.* Ed. by Léonard Kwuida and Baris Sertkaya. Vol. 5986. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 312–340.

Ganter, Bernhard, Peter A. Grigoriev, Sergei O. Kuznetsov, and Mikhail V. Samokhin (2004). “Concept-Based Data Mining with Scaled Labeled Graphs”. In: *Concept. Struct. Work SE - 6.* Ed. by KarlErich Wolff, HeatherD. Pfeiffer, and HarryS. Delugach. Vol. 3127. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 94–108.

Ganter, Bernhard and Sergei O. Kuznetsov (2000). “Formalizing Hypotheses with Concepts”. In: *Concept. Struct. Logical, Linguist. Comput. Issues SE - 24.* Ed. by Bernhard Ganter and GuyW. Mineau. Vol. 1867. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 342–356.

— (2001). “Pattern Structures and Their Projections”. In: *Concept. Struct. Broadening Base.* Ed. by Harry S. Delugach and Gerd Stumme. Vol. 2120. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 129–142.

— (2003). “Hypotheses and Version Spaces”. In: *Concept. Struct. Knowl. Creat. Commun. SE - 6.* Ed. by Bernhard Ganter, Aldo de Moor, and Wilfried Lex. Vol. 2746. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 83–95.

Ganter, Bernhard and Rudolf Wille (1999). *Formal Concept Analysis: Mathematical Foundations.* 1st. Springer, pp. I–X, 1–284.

Garcia-Blasco, Sandra, Santiago M Mola-Velasco, Roxana Danger, and Paolo Rosso (2011). "Automatic Drug-Drug Interaction Detection: A Machine Learning Approach With Maximal Frequent Sequence Extraction". In: *Challenge Task on Drug-Drug Interaction Extraction*, pp. 51–58.

Garofalakis, Minos N, Rajeev Rastogi, and Kyuseok Shim (1999). "SPIRIT: Sequential pattern mining with regular expression constraints". In: *VLDB*. Vol. 99, pp. 7–10.

Garriga, Gemma C., Roni Khardon, and Luc De Raedt (2012). "Mining closed patterns in relational, graph and network data". English. In: *Ann. Math. Artif. Intell.* Pp. 1–28.

Geng, Liqiang and Howard J. Hamilton (2006). "Interestingness Measures for Data Mining: A Survey". In: *ACM Comput. Surv.* 38.3, 9–es.

Gouda, Karam and Mohammed Javeed Zaki (2005). "Genmax: An efficient algorithm for mining maximal frequent itemsets". In: *Data Min. Knowl. Discov.* 11.3, pp. 223–242.

Grahne, Gösta and Jianfei Zhu (2003). "Efficiently Using Prefix-trees in Mining Frequent Itemsets." In: *FIMI*. Vol. 90.

Guns, Tias, Siegfried Nijssen, and Luc De Raedt (2011a). "Itemset mining: A constraint programming perspective". In: *Artif. Intell.* 175.12, pp. 1951–1983.

— (2011b). "k -Pattern Set Mining under Constraints". In: 1.1, pp. 1–18.

Guzelian, P S, M S Victoroff, N C Halmes, R C James, and C P Guzelian (2005). "Evidence-Based Toxicology: a Comprehensive Framework for Causation". In: *Hum. Exp. Toxicol.* 24.4, pp. 161–201.

Hacene, Mohamed Rouane, Marianne Huchard, Amedeo Napoli, and Petko Valtchev (2013). "Relational concept analysis: mining concept lattices from multi-relational data". In: *Ann. Math. Artif. Intell.* 67.1, pp. 81–108.

Haider, N (2010). "Functionality pattern matching as an efficient complementary structure/reaction search tool: an open-source approach." In: *Molecules* 15.8, pp. 5079–5092.

Han, Jiawei, Hong Cheng, Dong Xin, and Xifeng Yan (2007). "Frequent pattern mining: current status and future directions". In: *Data Min. Knowl. Discov.* 15.1, pp. 55–86.

Han, Jiawei and Jian Pei (2000). "Mining frequent patterns by pattern-growth: methodology and implications". In: *ACM SIGKDD Explor. Newsl.* 2.2, pp. 14–20.

Han, Jiawei, Jian Pei, Behzad Mortazavi-Asl, Qiming Chen, Umeshwar Dayal, and Meichun Hsu (2000). "FreeSpan: frequent pattern-projected sequential pattern mining". In: *Proc. 6th ACM SIGKDD Int'l Conf. Knowl. Discov. data Min.* Pp. 355–359.

Han, Jiawei, Jianyong Wang, Ying Lu, and P Tzvetkov (2002). "Mining top-k frequent closed patterns without minimum support". In: *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE Int. Conf.* Pp. 211–218.

Hanley, J A and B J McNeil (1982). "The Meaning and Use of the Area Under a Receiver Operating Characteristic (ROC) Curve". In: *Radiology* 143.1, pp. 29–36.

Hansen, K, S Mika, T Schroeter, A Sutter, A ter Laak, T Steger-Hartmann, N Heinrich, and K.-R. Müller (2009). "Benchmark Data Set for In silico Prediction of Ames Mutagenicity". In: *J. Chem. Inf. Model.* 49.9, pp. 2077–2081.

Hasan, Mohammad Al, Vineet Chaoji, Saeed Salem, Jeremy Besson, and Mohammed Javeed Zaki (2007). "ORIGAMI: Mining Representative Orthogonal Graph Patterns". In: *Seventh IEEE Int. Conf. Data Min. (ICDM 2007)*. IEEE, pp. 153–162.

Hasan, Mohammad Al and Mohammed Javeed Zaki (2009a). "MUSK: Uniform Sampling of k Maximal Patterns". In: *Proc. SDM*, pp. 650–661.

Hasan, Mohammad Al and Mohammed Javeed Zaki (2009b). “Output space sampling for graph patterns”. In: *Proc. VLDB Endow.* 2.1, pp. 730–741.

He, Zhisong, Jian Zhang, Xiao-He Shi, Le-Le Hu, Xiangyin Kong, Yu-Dong Cai, and Kuo-Chen Chou (2010). “Predicting Drug-Target Interaction Networks Based on Functional Groups and Biological Features”. In: *PLoS One* 5, e9603.

Hébert, Céline and Bruno Crémilleux (2005). “Mining Frequent δ-Free Patterns in Large Databases”. In: *Discov. Sci.* Ed. by Achim Hoffmann, Hiroshi Motoda, and Tobias Scheffer. Vol. 3735. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 124–136.

Helma, Christoph (2006). “Lazy Structure-Activity Relationships (lazar) for the Prediction of Rodent Carcinogenicity and Salmonella Mutagenicity”. In: *Mol. Divers.* 10.2, pp. 147–158.

Helma, Christoph, Tobias Cramer, Stefan Kramer, and Luc De Raedt (2004). “Data Mining and Machine Learning Techniques for the Identification of Mutagenicity Inducing Substructures and Structure Activity Relationships of Noncongeneric Compounds”. In: *J. Chem. Inf. Comput. Sci.* 44.4, pp. 1402–1411.

Hilderman, Robert J. and Howard J. Hamilton (1999). “Heuristic Measures of Interestingness”. In: *Princ. Data Min. Knowl. Discov.* Ed. by Jan M. Żytkow and Jan Rauch. Vol. 1704. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 232–241.

Huan, Jun, Wei Wang, Jan Prins, and Jiong Yang (2004). “SPIN: mining maximal frequent subgraphs from graph databases”. In: *Proc. 2004 ACM SIGKDD Int. Conf. Knowl. Discov. data Min. - KDD ’04.* New York, New York, USA: ACM Press, p. 581.

Jay, Nicolas, François Kohler, and Amedeo Napoli (2008). “Analysis of Social Communities with Iceberg and Stability-Based Concept Lattices”. In: *Form. Concept Anal.* Ed. by Raoul Medina and Sergei Obiedkov. Vol. 4933. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 258–272.

Jiang, Chuntao, Frans Coenen, and Michele Zito (2013). “A survey of frequent subgraph mining algorithms”. English. In: *Knowl. Eng. Rev.* The Knowledge Engineering Review 28.01, pp. 75–105.

Jianyong Wang, Zhiping Zeng, and Lizhu Zhou (2006). “CLAN: An Algorithm for Mining Closed Cliques from Large Dense Graph Databases”. In: *22nd Int. Conf. Data Eng.* IEEE, pp. 73–73.

Jin, Ning, Calvin Young, and Wei Wang (2009). “Graph classification based on pattern co-occurrence”. In: *Proceeding 18th ACM Conf. Inf. Knowl. Manag. - CIKM ’09.* New York, New York, USA: ACM Press, p. 573.

— (2010). “GAIA: graph classification using evolutionary computation”. In: *Proc. 2010 Int. Conf. Manag. data - SIGMOD ’10.* SIGMOD ’10. New York, NY, USA: ACM Press, pp. 879–890.

Judson, Philip N., P A Cooke, N G Doerrer, N Greene, R P Hanzlik, C Hardy, A Hartmann, D Hinchliffe, J Holder, L Müller, T Steger-Hartmann, A Rothfuss, M Smith, K Thomas, J D Vessey, and E Zeiger (2005). “Towards the Creation of an International Toxicology Information Centre.” In: *Toxicology* 213.1–2, pp. 117–128.

Jung, Kyung, Byung Park, and Soon-Sun Hong (2012). “Progress in cancer therapy targeting c-Met signaling pathway”. In: *Arch. Pharm. Res.* 35.4, pp. 595–604.

Kaiser, Tim B. and Stefan E. Schmidt (2011). “Some Remarks on the Relation between Annotated Ordered Sets and Pattern Structures”. In: *Pattern Recognit. Mach. Intell. SE - 9.* Ed. by Sergei O. Kuznetsov, DebaP. Mandal, MalayK. Kundu, and SankarK. Pal. Vol. 6744. Lecture Notes in Computer Science x. Springer Berlin Heidelberg, pp. 43–48.

Kavlock, Robert J, Gerald Ankley, Jerry Blancato, Michael Breen, Rory Conolly, David Dix, Keith Houck, Elaine Hubal, Richard Judson, James Rabinowitz, Ann Richard, R Woodrow

Setzer, Imran Shah, Daniel Villeneuve, and Eric Weber (2008). “Computational Toxicology – A State of the Science Mini Review”. In: *Toxicol. Sci.* 103.1, pp. 14–27.

Kaytoue, Mehdi, Sergei O. Kuznetsov, and Amedeo Napoli (2011). “Revisiting Numerical Pattern Mining with Formal Concept Analysis”. In: *IJCAI 2011, Proc. 22nd Int. Jt. Conf. Artif. Intell. Barcelona, Catalonia, Spain, July 16-22, 2011*, pp. 1342–1347.

Kaytoue, Mehdi, Sergei O. Kuznetsov, Amedeo Napoli, and Sébastien Duplessis (2011). “Mining gene expression data with pattern structures in formal concept analysis”. In: *Inf. Sci. (Ny).* 181.10, pp. 1989–2001.

Kazius, Jeroen, Ross McGuire, and Roberta Bursi (2005). “Derivation and validation of toxicophores for mutagenicity prediction”. In: *J. Med. Chem.* 48.1, pp. 312–320.

Kazius, Jeroen, Siegfried Nijssen, J N Kok, T Bäck, and A P Ijzerman (2006). “Substructure Mining Using Elaborate Chemical Representation”. In: *J. Chem. Inf. Comput. Sci.* 46.2, pp. 597–605.

King, Ross D., Ashwin Srinivasan, and Luc Dehaspe (2001). “Warmr: a data mining tool for chemical data”. English. In: *J. Comput. Aided. Mol. Des.* 15.2, pp. 173–181.

Klein, Dan and Christopher D Manning (2003). “Accurate unlexicalized parsing”. In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1.* Association for Computational Linguistics, pp. 423–430.

Klimushkin, Mikhail, Sergei A. Obiedkov, and Camille Roth (2010). “Approaches to the Selection of Relevant Concepts in the Case of Noisy Data”. In: *Proc. 8th Int. Conf. Form. Concept Anal.* ICFCA’10. Springer, pp. 255–266.

Klopman, G J (1984). “Artificial Intelligence Approach to Structure-Activity Studies: Computer Automated Structure Evaluation of Biological Activity of Organic Molecules.” In: *J. Am. Chem. Soc.* 106.24, pp. 7315–7321.

Krajca, Petr, Jan Outrata, and Vilem Vychodil (2010). “Advances in Algorithms Based on CbO.” In: *Proc. 8th Int. Conf. Concept Lattices Their Appl. (CLA’10).* Pp. 325–337.

Krishna, Varun, N. N. R. Ranga Suri, and G. Athithan (2011). “A comparative survey of algorithms for frequent subgraph discovery”. In: *Curr. Sci.* 100.2, pp. 190–198.

Kruhlak, N L, J F Contrera, R D Benz, and E J Matthews (2007). “Progress in QSAR Toxicity Screening of Pharmaceutical Impurities and Other FDA Regulated Products”. In: *Adv. Drug Deliv. Rev.* 59, pp. 43–55.

Kum, Hye-Chung, Joong Hyuk Chang, and Wei Wang (2007). “Benchmarking the effectiveness of sequential pattern mining methods”. In: *Data Knowl. Eng.* 60.1, pp. 30–50.

Kuramochi, Michihiro and George Karypis (2001). “Frequent subgraph discovery”. In: *Proc. 2001 IEEE Int. Conf. Data Min.* IEEE Comput. Soc, pp. 313–320.

— (2004). “GREW-A Scalable Frequent Subgraph Discovery Algorithm”. In: *Proc. Fourth IEEE Int. Conf. Data Min.* ICDM ’04. Washington, DC, USA: IEEE Computer Society, pp. 439–442.

— (2005). “Finding Frequent Patterns in a Large Sparse Graph”. English. In: *Data Min. Knowl. Discov.* 11.3, pp. 243–271.

Kuznetsov, Sergei O. (1989). “Interpretation on graphs and complexity characteristics of a search for specific patterns”. In: *Nauchno-Tekhnicheskaya Informatsiya Seriya 2 (Autom. Doc. Mathem. Ling.)* 23.1, pp. 23–27.

— (1990). “Stability as an Estimate of the Degree of Substantiation of Hypotheses on the Basis of Operational Similarity”. In: *Nauchno-Tekhnicheskaya Informatsiya Seriya 2 (Autom. Doc. Math. Linguist.* 24.12, pp. 21–29.

Kuznetsov, Sergei O. (1993). “A fast algorithm for computing all intersections of objects from an arbitrary semilattice”. In: *Nauchno-Tekhnicheskaya Informatsiya Seriya 2 (Autom. Doc. Mathem. Ling.)* 1, pp. 17–20.

— (1996). “Mathematical aspects of concept analysis”. In: *J. Math. Sci.* 80.2, pp. 1654–1698.

— (1999). “Learning of Simple Conceptual Graphs from Positive and Negative Examples”. In: *Princ. Data Min. Knowl. Discov. SE - 47*. Ed. by Jan M. Żytkow and Jan Rauch. Vol. 1704. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 384–391.

— (2001). “On Computing the Size of a Lattice and Related Decision Problems”. In: *Order* 18.4, pp. 313–321.

— (2007). “On stability of a formal concept”. In: *Ann. Math. Artif. Intell.* 49.1-4, pp. 101–115.

— (2013). “Fitting Pattern Structures to Knowledge Discovery in Big Data”. In: *Form. Concept Anal. SE - 17*. Ed. by Peggy Cellier, Felix Distel, and Bernhard Ganter. Vol. 7880. Lecture Notes in Computer Science. Springer, pp. 254–266.

Kuznetsov, Sergei O. and Sergei A. Obiedkov (2002). “Comparing performance of algorithms for generating concept lattices”. In: *J. Exp. Theor. Artif. Intell.* 14.2-3, pp. 189–216.

Kuznetsov, Sergei O., Sergei A. Obiedkov, and Camille Roth (2007). “Reducing the Representation Complexity of Lattice-Based Taxonomies”. In: *Concept. Struct. Knowl. Archit. Smart Appl.* Ed. by Uta Priss, Simon Polovina, and Richard Hill. Vol. 4604. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 241–254.

Kuznetsov, Sergei O. and Jonas Poelmans (2013). “Knowledge representation and processing with formal concept analysis”. In: *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* 3.3, pp. 200–215.

Kuznetsov, Sergei O. and Mikhail V. Samokhin (2005). “Learning Closed Sets of Labeled Graphs for Chemical Applications”. In: *Inductive Log. Program. SE - 12*. Ed. by Stefan Kramer and Bernhard Pfahringer. Lecture No. Vol. 3625. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 190–208.

Kwuida, Léonard, Rokia Missaoui, Beligh Ben Amor, Lahcen Boumedjout, and Jean Vaillancourt (2010). “Restrictions on Concept Lattices for Pattern Management.” In: *CLA*. Dm, pp. 235–246.

Lagunin, A, A Stepanchikova, D Filimonov, and V Poroikov (2000). “PASS: Prediction of Activity Spectra for Biologically Active Substances”. In: *Bioinformatics* 16, pp. 747–748.

Lai, David Y. and Yin-Tak Woo (2005). “OncoLogic: a mechanism-based expert system for predicting the carcinogenic potential of chemicals”. In: *Predict. Toxicol.* CRC Press. Chap. A Mechanis, pp. 385–413.

Lai, David Y., Yin-Tak Woo, M F Argus, and J C Arcos (1995). “Development of Structure-Activity Relationship Rules for Predicting Carcinogenic Potential of Chemicals”. In: *Toxicol. Lett.* 95.1–3, pp. 219–228.

Leach, Andrew R, Brian K Shoichet, and Catherine E Peishoff (2006). “Prediction of Protein-Ligand Interactions. Docking and Scoring: Successes and Gaps”. In: *J. Med. Chem.* 49.20, pp. 5851–5855.

Leeuwen, K. van, T. W. Schultz, T. Henry, B. Diderich, and G. D. Veith (2009). “Using chemical categories to fill data gaps in hazard assessment”. In: *SAR QSAR Environ. Res.* 20.3–4, pp. 207–220.

Leeuwenberg, Artuur, Aleksey Buzmakov, Yannick Toussaint, and Amedeo Napoli (2015). “Exploring Pattern Structures of Syntactic Trees for Relation Extraction”. In: *Form. Concept Anal.* Vol. 9113. Lecture Notes in Computer Science. Springer, pp. 153–168.

Lozano, Sylvain, Guillaume Poezevara, Marie-Pierre Halm-Lemeille, Elodie Lescot-Fontaine, Alban Lepailleur, Ryan Bissell-Siders, Bruno Crémilleux, Sylvain Rault, Bertrand Cuissart, and

Ronan Bureau (2010). “Introduction of jumping fragments in combination with QSARs for the assessment of classification in ecotoxicology.” In: *J. Chem. Inf. Model.* 50.8, pp. 1330–1339.

Luo, Congnan and Soon M. Chung (2004). “A scalable algorithm for mining maximal frequent sequences using sampling”. In: *Proc. - Int. Conf. Tools with Artif. Intell. ICTAI*, pp. 156–165.

Mabroukeh, Nizar R and C I Ezeife (2010). “A Taxonomy of Sequential Pattern Mining Algorithms”. In: *ACM Comput. Surv.* 43.1, 3:1–3:41.

Mannila, Heikki and H Toivonen (1997). “Levelwise search and borders of theories in knowledge discovery”. In: *Data Min. Knowl. Discov.* 1.3, pp. 241–258.

Mannila, Heikki, Hannu Toivonen, and A Inkeri Verkamo (1994). “Efficient Algorithms for Discovering Association Rules”. In: *Knowl. Discov. Data Min.* Pp. 181–192.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schtze (2008). *Introduction to Information Retrieval.*

Martin, Todd M, Paul Harten, Douglas M Young, Eugene N Muratov, Alexander Golbraikh, Hao Zhu, and Alexander Tropsha (2012). “Does Rational Selection of Training and Test Sets Improve the Outcome of QSAR Modeling?” In: *J. Chem. Inf. Model.* 52.10, pp. 2570–2578.

Masood, Adnan and Stephen Soong (2013). “Measuring Interestingness – Perspectives on Anomaly Detection”. In: *Comput. Eng. Intell. Syst.* 4.1, pp. 29–40.

Masseglia, Florent, Fabienne Cathala, and Pascal Poncelet (1998a). “The PSP Approach for Mining Sequential Patterns”. In: *PKDD*, pp. 176–184.

— (1998b). “The psp approach for mining sequential patterns”. In: *Princ. Data Min. Knowl. Discov.* Springer, pp. 176–184.

Maunz, Andreas, Christoph Helma, Tobias Cramer, and Stefan Kramer (2010). “Latent Structure Pattern Mining”. In: *Mach. Learn. Knowl. Discov. Databases SE - 23.* Ed. by JoséLuis Balcázar, Francesco Bonchi, Aristides Gionis, and Michèle Sebag. Vol. 6322. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 353–368.

McGarry, Ken (2005). “A survey of interestingness measures for knowledge discovery”. English. In: *Knowl. Eng. Rev.* 20.01, p. 39.

Merwe, Dean Van Der, Sergei A. Obiedkov, and Derrick G. Kourie (2004). “AddIntent: A new incremental algorithm for constructing concept lattices”. In: *Concept Lattices.* Ed. by Gerhard Goos, Juris Hartmanis, Jan Leeuwen, and Peter Eklund. Vol. 2961. Springer, pp. 372–385.

Messai, Nizar, Marie-Dominique Devignes, Amedeo Napoli, and Malika Smaïl-Tabbone (2008). “Extending Attribute Dependencies for Lattice-Based Querying and Navigation”. In: *Concept. Struct. Knowl. Vis. Reason.* Ed. by Peter Eklund and Ollivier Haemmerlé. Vol. 5113. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 189–202.

Métivier, Jean-Philippe, Alban Lepailleur, Aleksey Buzmakov, Guillaume Poezevara, Bruno Crémilleux, Sergei Kuznetsov, Jérémie Le Goff, Amédéo Napoli, Ronan Bureau, and Bertrand Cuissart (2015). “Discovering structural alerts for mutagenicity using stable emerging molecular patterns”. In: *J. Chem. Inf. Model.* 55.5, pp. 925–940.

Minard, Anne-Lyse, Lamia Makour, Anne-Laure Ligozat, and Brigitte Grau (2011). “Feature Selection for Drug-Drug Interaction Detection Using Machine-Learning Based Approaches”. In: *Challenge Task on Drug-Drug Interaction Extraction*, pp. 43–50.

Mitchell, Tom Michael (1978). *Version spaces: an approach to concept learning.* Tech. rep. Stanford, CA, USA: STANFORD UNIV CALIF DEPT OF COMPUTER SCIENCE.

Moerchen, Fabian, Michael Thies, and Alfred Ultsch (2011). “Efficient mining of all margin-closed itemsets with applications in temporal knowledge discovery and classification by compression”. In: *Knowl. Inf. Syst.* 29.1, pp. 55–80.

Mooney, Carl H. and John F. Roddick (2013). "Sequential pattern mining – approaches and algorithms". In: *ACM Comput. Surv.* 45.2, pp. 1–39.

Murtagh, F. (1983). "A Survey of Recent Advances in Hierarchical Clustering Algorithms". In: *Comput. J.* 26.4, pp. 354–359.

Muster, W, A Breidenbach, H Fischer, S Kirchner, L Müller, and A Pähler (2008). "Computational Toxicology in Drug Development". In: *Drug Discov. Today* 13, pp. 303–310.

Najdenova, K A (1963). "A formal model of knowledge interpretation on the basis of classification process". In: *IFAC*. Publishing house "ZINATNE", pp. 175–180.

Nijssen, Siegfried and Joost N. Kok (2005). "The Gaston Tool for Frequent Subgraph Mining". In: *Electron. Notes Theor. Comput. Sci.* 127.1, pp. 77–87.

Nikolaev, Nikolay I. and Evgueni N. Smirnov (1996). "Stochastically Guided Disjunctive Version Space Learning". In: John Wiley & Sons, Ltd.

Norman, Mark H, Longbin Liu, Matthew Lee, Ning Xi, Ingrid Fellows, Noel D D'Angelo, Celia Dominguez, Karen Rex, Steven F Bellon, Tae-Seong Kim, and Isabelle Dussault (2012). "Structure-Based Design of Novel Class II c-Met Inhibitors: 1. Identification of Pyrazolone-Based Derivatives". In: *J. Med. Chem.* 55.5, pp. 1858–1867.

Norris, Eugene M. (1978). "An algorithm for computing the maximal rectangles in a binary relation". In: *Rev. Roum. Mathématiques Pures Appliquées* 23.2, pp. 243–250.

Nourine, Lhouari and Olivier Raynaud (2002). "A fast incremental algorithm for building lattices". In: *J. Exp. Theor. Artif. Intell.* 14.2-3, pp. 217–227.

Oosthuizen, G.D. (1988). "The use of of a lattice in Knowledge Processing". PhD thesis. University of Strathclyde, Glasgow.

Orlando, Salvatore, Raffaele Perego, and Claudio Silvestri (2004). "A new algorithm for gap constrained sequence mining". In: *Proc. 2004 ACM Symp. Appl. Comput.* ACM, pp. 540–547.

Papadopoulos, Apostolos N., Apostolos Lyritsis, and Yannis Manolopoulos (2008). "SkyGraph: an algorithm for important subgraph discovery in relational graphs". In: *Data Min. Knowl. Discov.* 17.1, pp. 57–76.

Pasquier, Nicolas, Yves Bastide, Rafik Taouil, and Lotfi Lakhal (1999). "Efficient Mining of Association Rules Using Closed Itemset Lattices". In: *Inf. Syst.* 24.1, pp. 25–46.

Paulheim, Heiko and Christian Bizer (2013). "Type Inference on Noisy RDF Data". In: *12th International Semantic Web Conference.*

Pearl, G M, S Livingston-Carr, and S K Durham (2001). "Integration of Computational Analysis as a Sentinel Tool in Toxicological Assessments". In: *Curr. Top. Med. Chem.* 1, pp. 247–255.

Pei, Jian, Jiawei Han, Behzad Mortazavi-Asl, H. Pinto, Qiming Chen, U. Dayal, and Mei-Chun Hsu (2001). "PrefixSpan Mining Sequential Patterns Efficiently by Prefix Projected Pattern Growth". In: *17th Int. Conf. Data Eng.* Pp. 215–226.

Pei, Jian, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu (2001). "PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth". In: *ICDE*, pp. 215–224.

Pennerath, Frédéric, Gilles Niel, Philippe Vismara, Philippe Jauffret, Claude Laurenço, and Amedeo Napoli (2010). "Graph-Mining Algorithm for the Evaluation of Bond Formability". In: *J. Chem. Inf. Model.* 50.2, pp. 221–239.

Pernelle, Nathalie, Marie-Christine Rousset, Henry Soldano, and Véronique Ventos (2002). "ZooM: a nested Galois lattices-based system for conceptual clustering". In: *J. Exp. Theor. Artif. Intell.* 14.2-3, pp. 157–187.

Pinto, Helen, Jiawei Han, Jian Pei, Ke Wang, Qiming Chen, and Umeshwar Dayal (2001). "Multi-Dimensional Sequential Pattern Mining". In: *CIKM*, pp. 81–88.

Plantevit, Marc, Yeow Wei Choong, Anne Laurent, Dominique Laurent, and Maguelonne Teisseire (2005). “M2SP: Mining Sequential Patterns Among Several Dimensions”. In: *Knowl. Discov. Databases PKDD 2005*. Ed. by Alípio Mário Jorge, Luís Torgo, Pavel Brazdil, Rui Camacho, and João Gama. Vol. 3721. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 205–216.

Plantevit, Marc, Anne Laurent, Dominique Laurent, Maguelonne Teisseire, and Yeow Wei Choong (2010). “Mining multidimensional and multilevel sequential patterns”. In: *ACM Trans. Knowl. Discov. Data* 4.1, pp. 1–37.

Plotkin, Gordon D (1970). “A note on inductive generalization”. In: *Mach. Intell.* 5.1, pp. 153–163.

Poelmans, Jonas, Dmitry I. Ignatov, Sergei O. Kuznetsov, and Guido Dedene (2013). “Formal concept analysis in knowledge processing: A survey on applications”. In: *Expert Syst. Appl.* 40.16, pp. 6538–6560.

Poelmans, Jonas, Sergei O. Kuznetsov, Dmitry I. Ignatov, and Guido Dedene (2013). “Formal Concept Analysis in knowledge processing: A survey on models and techniques”. In: *Expert Syst. Appl.* 40.16, pp. 6601–6623.

Poezevara, Guillaume, Bertrand Cuissart, and Bruno Crémilleux (2011). “Extracting and Summarizing the Frequent Emerging Graph Patterns from a Dataset of Graphs”. In: *J. Intell. Inf. Syst.* 37.3, pp. 333–353.

Qian, Gang, Shamik Sural, Yuelong Gu, and Sakti Pramanik (2004). “Similarity between euclidean and cosine angle distance for nearest neighbor queries”. In: *Proc. 2004 ACM Symp. Appl. Comput.* ACM Press, pp. 1232–1237.

Raïssi, Chedy, Toon Calders, and Pascal Poncelet (2008). “Mining conjunctive sequential patterns”. In: *Data Min. Knowl. Discov.* 17.1, pp. 77–93.

Rees, David C, Miles Congreve, Christopher W Murray, and Robin Carr (2012). “Fragment-based lead discovery”. In: *Nat Rev Drug Discov* 3.8, pp. 660–672.

Rickert, Keith W, Sangita B Patel, Timothy J Allison, Noel J Byrne, Paul L Darke, Rachael E Ford, David J Guerin, Dawn L Hall, Maria Kornienko, Jun Lu, Sanjeev K Munshi, John C Reid, Jennifer M Shipman, Elizabeth F Stanton, Kevin J Wilson, Jonathon R Young, Stephen M Soisson, and Kevin J Lumb (2011). “Structural Basis for Selective Small Molecule Kinase Inhibition of Activated c-Met”. In: *J. Biol. Chem.* 286.13, pp. 11218–11225.

Ridings, J. E., M. D. Barratt, R. Cary, C. G. Earnshaw, C. E. Eggington, M. K. Ellis, P. N. Judson, J. J. Langowski, C. A. Marchant, M. P. Payne, W. P. Watson, and T. D. Yih (1996). “Computer Prediction of Possible Toxic Action from Chemical Structure: an Update on the DEREK System”. In: *Toxicology* 106.1–3, pp. 267–279.

Rogers, M D (2003). “The European Commission’s White Paper “Strategy for a Future Chemicals Policy": a Review”. In: *Risk Anal.* 23.2, pp. 381–388.

Roth, Camille, Sergei A. Obiedkov, and Derrick G. Kourie (2006). “Towards concise representation for taxonomies of epistemic communities”. In: *Proc. 4th Int. Conf. Concept lattices their Appl.* CLA’06. Berlin, Heidelberg: Springer-Verlag, pp. 240–255.

— (2008). “On succinct representation of knowledge community taxonomies with formal concept analysis”. In: *Int. J. Found. Comput. Sci.* 19.02, pp. 383–404.

Salvemini, Eliana, Fabio Fumarola, Donato Malerba, and Jiawei Han (2011). “FAST sequence mining based on sparse id-lists”. In: *Proceedings of the 19th international conference on Foundations of intelligent systems*. ISMIS’11. Berlin, Heidelberg: Springer-Verlag, pp. 316–325.

Sanderson, D. M. and C. G. Earnshaw (1991). “Computer-Prediction of Possible Toxic Action from Chemical-Structure - The DEREK System”. In: *Hum. Exp. Toxicol.* 10.4, pp. 261–273.

Schietgat, Leander, Jan Ramon, Maurice Bruynooghe, and Hendrik Blockeel (2008). "An Efficiently Computable Graph-Based Metric for the Classification of Small Molecules". In: *Discov. Sci. SE - 20*. Ed. by Jean-François Boulicaut, Michael Berthold, and Tamás Horváth. Vol. 5255. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 197–209.

Seal, Abhik, Anurag Passi, U C Abdul Jaleel, and David J Wild (2012). "In-silico Predictive Mutagenicity Model Generation Using Supervised Learning Approaches". In: *J. Cheminformatics* 4, p. 10.

Sebag, Michèle (1996). "Delaying the choice of bias: A disjunctive version space approach". In: *ICML*. Morgan Kaufmann, pp. 444–452.

Segura-Bedmar, Isabel, Paloma Martınez, and Daniel Sánchez-Cisneros (2011). "The 1st DDIExtraction-2011 challenge task: Extraction of Drug-Drug Interactions from biomedical texts". In: *Challenge Task on Drug-Drug Interaction Extraction* 2011, pp. 1–9.

Shelokar, Prakash, Arnaud Quirin, Oscar Cordón, and Óscar Cordón (2013). "MOSubdue: a Pareto dominance-based multiobjective Subdue algorithm for frequent subgraph mining". English. In: *Knowl. Inf. Syst.* 34.1, pp. 75–108.

Sherhod, Richard, Valerie J. Gillet, Philip N. Judson, and Jonathan D. Vessey (2012). "Automating Knowledge Discovery for Toxicity Prediction Using Jumping Emerging Pattern Mining". In: *J. Chem. Inf. Model.* 52.11, pp. 3074–3087.

Sherhod, Richard, Philip N. Judson, Thierry Hanser, Jonathan D. Vessey, Samuel J. Webb, and Valerie J Gillet (2014). "Emerging Pattern Mining To Aid Toxicological Knowledge Discovery". In: *J. Chem. Inf. Model.* 54.7, pp. 1864–1879.

Smithing, Michael P. and Ferenc Darvas (1992). "HazardExpert - An Expert System for Predicting Chemical Toxicity". In: *ACS Sym. Ser.* 484, pp. 191–200.

Socher, Richard, John Bauer, Christopher D Manning, and Andrew Y Ng (2013). "Parsing with compositional vector grammars". In: *In Proceedings of the ACL conference*. Citeseer.

Soldano, Henry (2015). "Extensional Confluences and Local Closure Operators". In: *Form. Concept Anal.* Ed. by Jaume Baixeries, Christian Sacarea, and Manuel Ojeda-Aciego. Vol. 9113. Lecture Notes in Computer Science. Springer International Publishing, pp. 128–144.

Soldano, Henry and Guillaume Santini (2014). "Graph abstraction for closed pattern mining in attributed network". In: *Eur. Conf. Artif. Intell. (ECAI), IOS Press*, pp. 849–854.

Soldano, Henry and Véronique Ventos (2011). "Abstract Concept Lattices". In: *Form. Concept Anal.* Ed. by Petko Valtchev and Robert Jäschke. Vol. 6628. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 235–250.

Soulet, Arnaud and Bruno Crémilleux (2005). "Optimizing constraint-based mining by automatically relaxing constraints". In: *Proc. 5th IEEE Inter- Natl. Conf. Data Min. (ICDM 2005)*. Houston, Texas, USA: IEEE Computer Society, pp. 777–780.

— (2008). "Adequate condensed representations of patterns". In: *Data Min. Knowl. Discov.* 17.1, pp. 94–110.

Soulet, Arnaud, Chedy Raïssi, Marc Plantevit, and Bruno Cremilleux (2011). "Mining Dominant Patterns in the Sky". In: *2011 IEEE 11th Int. Conf. Data Min.* Vancouver, B.C, Canada: IEEE, pp. 655–664.

Spyropoulou, Eirini, Tijl De Bie, and Mario Boley (2013). "Interesting pattern mining in multi-relational data". English. In: *Data Min. Knowl. Discov.* April, pp. 1–42.

Srikant, Ramakrishnan and Rakesh Agrawal (1996). *Mining sequential patterns: Generalizations and performance improvements.* Springer.

Szathmary, Laszlo, Petko Valtchev, Amedeo Napoli, and Robert Godin (2009). "Efficient Vertical Mining of Frequent Closures and Generators". In: *Adv. Intell. Data Anal. VIII*. Ed. by Niall

M. Adams, Céline Robardet, Arno Siebes, and Jean-François Boulicaut. Vol. 5772. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 393–404.

Szathmary, Laszlo, Petko Valtchev, Amedeo Napoli, Robert Godin, A Boc, and V Makarenkov (2014). "A fast compound algorithm for mining generators, closed itemsets, and computing links between equivalence classes". In: *Ann. Math. Artif. Intell.* 70.1-2, pp. 81–105.

Tatti, Nikolaj, Fabian Moerchen, and Toon Calders (2014). "Finding Robust Itemsets under Subsampling". In: *ACM Trans. Database Syst.* 39.3, pp. 1–27.

Thomas, Lini T., Satyanarayana R. Valluri, and Kamalakar Karlapalem (2010). "MARGIN: Maximal frequent subgraph mining". In: *ACM Trans. Knowl. Discov. Data* 4.3, pp. 1–42.

Thomas, Philippe, Mariana Neves, Illés Solt, Domonkos Tikk, and Ulf Leser (2011). "Relation extraction for drug-drug interactions using ensemble learning". In: *Challenge Task on Drug-Drug Interaction Extraction*, pp. 11–18.

Tiedt, Ralph, Elisa Degenkolbe, Pascal Furet, Brent A Appleton, Sabrina Wagner, Joseph Schoepfer, Emily Buck, David A Ruddy, John E Monahan, Michael D Jones, Jutta Blank, Dorothea Haasen, Peter Drueckes, Markus Wartmann, Clive McCarthy, William R Sellers, and Francesco Hofmann (2011). "A Drug Resistance Screen Using a Selective MET Inhibitor Reveals a Spectrum of Mutations That Partially Overlap with Activating Mutations Found in Cancer Patients". In: *Cancer Res.* 71.15, pp. 5255–5264.

Tropsha, Alexander and Alexander Golbraikh (2007). "Predictive QSAR Modeling Workflow, Model Applicability Domains, and Virtual Screening". In: *Curr. Pharm. Des.* 13.34, pp. 3494–3504.

Tsumoto, Shusaku, Haruko Iwata, Shoji Hirano, and Yuko Tsumoto (2014). "Similarity-based behavior and process mining of medical practices". In: *Futur. Gener. Comput. Syst.* 33, pp. 21–31.

Uno, Takeaki, Tatsuya Asai, Yuzo Uchida, and Hiroki Arimura (2004). "An efficient algorithm for enumerating closed patterns in transaction databases". In: *Discov. Sci.* Ed. by Einoshin Suzuki and Setsuo Arikawa. Vol. 3245. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 16–31.

Uno, Takeaki, Masashi Kiyomi, and Hiroki Arimura (2005). "LCM Ver.3: Collaboration of Array, Bitmap and Prefix Tree for Frequent Itemset Mining". In: *Proc. 1st Int. Work. Open Source Data Min. Freq. Pattern Min. Implementations.* OSDM '05. New York, NY, USA: ACM, pp. 77–86.

Valerio, Luis G Jr. (2009). "In silico Toxicology for the Pharmaceutical Sciences". In: *Toxicol. Appl. Pharmacol.* 241.3, pp. 356–370.

Ventos, Vëronique, Henry Soldano, and Thibaut Lamadon (2004). "Alpha Galois lattices". In: *Fourth IEEE Int. Conf. Data Mining, 2004. ICDM '04.* Pp. 555–558.

Villanueva-rosales, Natalia and Michel Dumontier (2007). "Describing chemical functional groups in OWL-DL for the classification of chemical compounds." In: *InEuropean Semant. Web Conf.*

Vreeken, Jilles, Matthijs van Leeuwen, and Arno Siebes (2011). "Krimp: mining itemsets that compress". In: *Data Min. Knowl. Discov.* 23.1, pp. 169–214.

Vreeken, Jilles and Nikolaj Tatti (2014). "Interesting Patterns". In: *Freq. Pattern Min.* Ed. by Charu C Aggarwal and Jiawei Han. Springer International Publishing, pp. 105–134.

Webb, Geoffrey I. (2007). "Discovering Significant Patterns". English. In: *Mach. Learn.* 68.1, pp. 1–33.

— (2010). "Self-sufficient itemsets". In: *ACM Trans. Knowl. Discov. Data* 4.1, pp. 1–20.

— (2011). "Filtered-top-k association discovery". In: *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* 1.3, pp. 183–192.

Webb, Geoffrey I. and Songmao Zhang (2005). “K-Optimal Rule Discovery”. English. In: *Data Min. Knowl. Discov.* 10.1, pp. 39–79.

Wienand, Dominik and Heiko Paulheim (2014). “Detecting Incorrect Numerical Data in DBpedia”. In: *11th Extended Semantic Web Conference.*

Wörlein, Marc, Thorsten Meinl, Ingrid Fischer, and Michael Philippsen (2005). “A Quantitative Comparison of the Subgraph Miners MoFa, gSpan, FFSM, and Gaston”. In: *Knowl. Discov. Databases PKDD 2005 SE - 39.* Ed. by AlípioMário Jorge, Luís Torgo, Pavel Brazdil, Rui Camacho, and João Gama. Vol. 3721. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 392–403.

Xin, Dong, Hong Cheng, Xifeng Yan, and Jiawei Han (2006). “Extracting redundancy-aware top-k patterns”. In: *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discov. data Min. - KDD ’06.* New York, New York, USA: ACM Press, p. 444.

Xu, Congying, Feixiong Cheng, Lei Chen, Zheng Du, Weihua Li, Guixia Liu, Philip W Lee, and Yun Tang (2012). “In silico Prediction of Chemical Ames Mutagenicity”. In: *J. Chem. Inf. Model.* 52.11, pp. 2840–2847.

Yamagishi, Michel, Natália Martins, Goran Neshich, Wensheng Cai, Xueguang Shao, Alexandre Beautrait, and Bernard Maigret (2006). “A fast surface-matching procedure for protein–ligand docking”. In: *J. Mol. Model.* 12.6, pp. 965–972.

Yan, Xifeng, Hong Cheng, Jiawei Han, and Philip S. Yu (2008). “Mining significant graph patterns by leap search”. In: *Proc. 2008 ACM SIGMOD Int. Conf. Manag. data - SIGMOD ’08.* New York, New York, USA: ACM Press, pp. 433–444.

Yan, Xifeng and Jiawei Han (2002). “gSpan: Graph-Based Substructure Pattern Mining”. In: *Data Mining, 2002. ICDM 2003. Proceedings. . . .* Pp. 721–724.

— (2003). “CloseGraph: mining closed frequent graph patterns”. In: *Proc. ninth ACM SIGKDD Int. Conf. Knowl. Discov. data Min.* KDD ’03. New York, NY, USA: ACM, pp. 286–295.

Yan, Xifeng, Jiawei Han, and Ramin Afshar (2003). “CloSpan: Mining Closed Sequential Patterns in Large Databases”. In: *Proc. SIAM Int’l Conf. Data Min.* Pp. 166–177.

Yang, Zhenglu, Masaru Kitsuregawa, and Yitong Wang (2006). “PAID: Mining Sequential Patterns by Passed Item Deduction in Large Databases”. In: *IDEAS*, pp. 113–120.

Yao, Hong and Howard J. Hamilton (2006). “Mining itemset utilities from transaction databases”. In: *Data Knowl. Eng.* 59.3, pp. 603–626.

Yu, Chung-Ching and Yen-Liang Chen (2005). “Mining Sequential Patterns from Multidimensional Sequence Data”. In: *IEEE Trans. Knowl. Data Eng.* 17.1, pp. 136–140.

Yu, Yang and Jeff Heflin (2011a). “Detecting abnormal data for ontology based information integration”. In: *2011 International Conference on Collaboration Technologies and Systems.*

— (2011b). “Extending Functional Dependency to Detect Abnormal Data in RDF Graphs”. In: *10th International Semantic Web Conference.*

Zaki, Mohammed Javeed (1998). “Efficient enumeration of frequent sequences”. In: *Proc. seventh Int. Conf. Inf. Knowl. Manag.* ACM, pp. 68–75.

— (2000). “Sequence mining in categorical domains: incorporating constraints”. In: *Proc. ninth Int. Conf. Inf. Knowl. Manag.* ACM, pp. 422–429.

— (2001). “SPADE: An Efficient Algorithm for Mining Frequent Sequences”. In: *Mach. Learn.* 42.1-2, pp. 31–60.

Zaki, Mohammed Javeed and Karam Gouda (2003). “Fast vertical mining using diffsets”. In: *Proc. ninth ACM SIGKDD Int. Conf. Knowl. Discov. data Min.* ACM, pp. 326–335.

Zaki, Mohammed Javeed and C. J. Hsiao (2005). “Efficient algorithms for mining closed itemsets and their lattice structure”. In: *IEEE Trans. Knowl. Data Eng.* 17.4, pp. 462–478.

Zaveri, Amrapali, Dimitris Kontokostas, Mohamed Ahmed Sherif, Lorenz Bühmann, Mohamed Morsey, Sören Auer, and Jens Lehmann (2013). "User-driven quality evaluation of DBpedia". In: *I-SEMANTICS 2013 - 9th International Conference on Semantic Systems.*

Zbidi, Naim, Sami Faiz, and Mohamed Limam (2006). "On Mining Summaries by Objective Measures of Interestingness". In: *Mach. Learn.* 62.3, pp. 175–198.

Zeng, Zhiping, Jianyong Wang, Jun Zhang, and Lizhu Zhou (2009). "FOGGER : An Algorithm for Graph Generator Discovery". In: *Proc. 12th Int. Conf. Extending Database Technol. Adv. Database Technol. - EDBT '09.* New York, New York, USA: ACM Press, pp. 517–528.

Zeng, Zhiping, Jianyong Wang, Lizhu Zhou, and George Karypis (2006). "Coherent closed quasi-clique discovery from large dense graph databases". In: *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discov. data Min. - KDD '06.* New York, New York, USA: ACM Press, p. 797.

Zhu, Feida, Xifeng Yan, Jiawei Han, and Philip S. Yu (2007). "gPrune: A Constraint Pushing Framework for Graph Pattern Mining". In: *Adv. Knowl. Discov. Data Min. SE - 38.* Ed. by Zhi-Hua Zhou, Hang Li, and Qiang Yang. Vol. 4426. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 388–400.

Zimmermann, Albrecht (2013). "Objectively evaluating condensed representations and interestingness measures for frequent itemset mining". In: *J. Intell. Inf. Syst.* Pp. 1–19.

Résumé

Aujourd'hui de plus en plus de données de différents types sont accessibles. Ces données variées contiennent des informations précieuses qui peuvent aider à résoudre des problèmes pratiques ou servir à la science fondamentale. *Mais comment peut-on extraire cette information précieuse ?* L'Analyse Formelle de Concepts (AFC) et les pattern structures sont des systèmes formels qui permettent de traiter les données ayant une structure complexe. *Mais comment peut-on les appliquer en pratique ?* De plus, le nombre de concepts, i.e., de pièces élémentaires d'information, trouvé par l'AFC est fréquemment très grand.

Pour faire face à ce problème, on peut simplifier la représentation des données, soit par projection de pattern structures, soit par introduction de contraintes pour sélectionner les concepts les plus pertinents. *Quelle est la meilleure simplification de données ? Comment peut-on efficacement trouver les concepts satisfaisant une contrainte donnée ?* Ce sont les questions que nous abordons dans cette thèse.

Le manuscrit commence avec l'application de l'AFC à l'exploration de structures moléculaires et la recherche de structures particulières. Ces structures moléculaires sont codées comme des ensembles d'attributs. Même pour ce codage simple et sans contraintes supplémentaires, l'AFC est capable d'extraire des informations importantes dans de petits ensembles de données. Avec l'augmentation de la taille des ensembles de données, de bonnes contraintes deviennent essentielles. Pour cela on explore la stabilité d'un concept. La stabilité est une contrainte formelle bien-fondée. On montre expérimentalement que c'est un bon choix et on l'applique à l'exploration d'un ensemble de données de substances chimiques mutagènes. La recherche de concepts stables dans cet ensemble de données nous a permis de trouver de nouveaux candidats mutagènes potentiels qui peuvent être interprétés par les chimistes.

Cependant, pour les cas plus complexes, la représentation simple par des attributs binaires ne suffit pas. En conséquence, on se tourne vers des pattern structures qui peuvent traiter différents types de données complexes. Le point important sur les pattern structures est qu'elles permettent la simplification des données au moyen de projections. On étend le formalisme original des projections pour avoir plus de liberté dans la manipulation de données. On montre que cette extension est essentielle pour analyser les trajectoires de patients décrivant l'historique de l'hospitalisation des patients. En effet, les trajectoires de patients sont des séquences d'hospitalisations et chaque hospitalisation est décrite par une description hétérogène. Ces données sont très riches et donc produisent un grand nombre de concepts. Le nouveau type de projection permet une réduction efficace de la notion d'espace et en combinaison avec des contraintes de stabilité on peut trouver des trajectoires communes importantes. En outre, les projections sont utiles pour corriger et compléter les données liées (linked open data) où les erreurs sont inévitables. On peut efficacement trouver certaines erreurs avec une approche à base de pattern structures. Une autre application des pattern structures est la recherche des interactions médicamenteuses dans un corpus de textes. On est capable d'extraire et d'expliquer les structures syntaxiques codant ce genre des relations.

Les approches présentées jusque là ne permettent pas la découverte directe de motifs sous la contrainte de stabilité. En conséquence, le manuscrit se termine par une approche originale et très efficace qui permet de trouver directement des motifs stables. Cette approche est appelée $\Sigma o \varphi \iota \alpha$ et est capable de trouver les meilleurs modèles stables en temps polynomial. L'efficacité est essentielle pour l'analyse de grands ensembles de données et cela souligne l'importance de $\Sigma o \varphi \iota \alpha$. On évalue ce nouvel algorithme sur plusieurs ensembles de données et les expériences montrent l'amélioration significative de $\Sigma o \varphi \iota \alpha$ par rapport à ses concurrents pour les données binaires et des n-uplets d'intervalles. En outre, il ouvre une nouvelle direction de recherche pour l'exploitation de différents types de motifs en temps polynomial ce qui est très important dans le monde des mégadonnées.

Mots-clés: Exploration de données, Analyse Formelle de Concepts, Pattern Structures, Contraintes, Stabilité, Projections

Abstract

Nowadays, more and more data of different kinds is becoming available. Various datasets contain valuable information that could help to solve many practical problems or to lead to a breakthrough in fundamental science. *But how can one extract these precious pieces of information?* Formal concept analysis (FCA) and pattern structures are theoretical frameworks that allow dealing with an arbitrary structured data. *But how can one put it into practice?* Furthermore, the number of concepts, i.e., elementary pieces of information, extracted by FCA is typically huge. To deal with this problem one can either simplify the data representation, which can be done by projections of pattern structures, or by introducing constraints to select the most relevant concepts. *What is the best data simplification? How to find concepts efficiently satisfying a given constraint?* These are the questions that we address in this work.

The manuscript starts with application of FCA to mining important pieces of information from molecular structures. These molecular structures are encoded as sets of attributes. Even for this simple encoding and without any additional constraints, FCA is able to extract important pieces of information from small datasets. With the growth of dataset size good constraints begin to be essential. For that we explore stability of a concept, a well-founded formal constraint. We show experimentally that it is a good choice and apply it to analyze a dataset of mutagenetic chemical substances. Finding stable concepts in this dataset allows us finding new possible mutagenetic candidates that can be further interpreted by chemists.

However for more complex cases, the simple attribute representation of data is not enough. Correspondingly, we turn to pattern structures that can deal with many different kinds of descriptions. The important point about pattern structures is that they allow data simplification by means of projections. We extend the original formalism of projections to have more freedom in data simplification. We show that this extension is essential for analyzing patient trajectories, describing patients hospitalization histories. Indeed, patient trajectories are sequences of hospitalizations and every hospitalization is described by a heterogeneous description. This data is very rich and hence, produce a lot of concepts. The new type of projections allows efficient reduction of concept space and in combination with stability constraints can find important common trajectories. In addition, projections are useful to correct linked open data, a data that is distributed all over the world and that can be enriched by any person. The errors are inevitable but some of them can be efficiently found by an approach based on pattern structures. Yet another application of pattern structures is mining of drug-drug interactions from text corpuses. Based on a text of corpuses we are able to find and explain syntactic structures encoding this kind of relation.

So far pattern structures do not allow direct finding of patterns satisfying the stability constraint. Correspondingly, the manuscript ends by an original and very efficient approach that enables to mine stable patterns directly. This approach is called $\Sigma o \varphi \iota \alpha$ and is able to find the best stable patterns in polynomial time. The efficiency is essential for mining large datasets and this highlights the importance of $\Sigma o \varphi \iota \alpha$. We evaluate this new algorithm on several datasets and the experiments show the significant improvement of $\Sigma o \varphi \iota \alpha$ w.r.t. its competitors for attribute and interval tuple data. Moreover it open a new direction of research for mining different types of patterns in polynomial time that is very important in the world of large data.

Keywords: Data Mining, Formal Concept Analysis, Pattern Structures, Constraints, Stability, Projections