Real-time Distributed Computation of Formal Concepts and Analytics

Cassio de Alburquerque Melo

École Centrale Paris

Doctoral Thesis

Real-time Distributed Computation of Formal Concepts and Analytics

Author:
Cassio Melo

Supervisors:
Marie-Aude Aufaure
Bénédicte Le Grand

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in the

Business Intelligence Team
MAS Laboratoire

November 2013

Declaration of Authorship

I, Cassio Melo, declare that this thesis titled, 'Real-time Distributed Computation of Formal Concepts and Analytics' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at École Centrale Paris.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at École Centrale Paris or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed: _____

Date: _____

"Better try and fail
To worry as life passes by
Better still try in vain
To sit down and wait until the end
I prefer walking in the rain
To hide in sad days
I'd rather be happy, mad though,
To a conformity life."

Martin Luther King Jr.

ÉCOLE CENTRALE PARIS

Résumé

Business Intelligence Team
MAS Laboratoire

Doctor of Philosophy

Real-time Distributed Computation of Formal Concepts and Analytics

by Cassio Melo

Les progrés de la technologie pour la création, le stockage et la diffusion des données ont considérablement augmenté le besoin d'outils qui permettent effectivement aux utilisateurs les moyens d'identifier et de comprendre l'information pertinente. Malgré les possibilités de calcul dans les cadres distribuées telles que des outils comme Hadoop offrent, il a seulement augmenté le besoin de moyens pour identifier et comprendre les informations pertinentes.

L'Analyse de Concepts Formels (ACF) peut jouer un rôle important dans ce contexte, en utilisant des moyens plus intelligents dans le processus d'analyse. ACF fournit une compréhension intuitive de la généralisation et de spécialisation des relations entre les objets et leurs attributs dans une structure connue comme un treillis de concepts.

Cette thèse aborde le problème de l'exploitation et visualisation des concepts sur un flux de données. L'approche proposée est composé de plusieurs composants distribués qui effectuent le calcul des concepts d'une transaction de base, filtre et transforme les données, les stocke et fournit des fonctionnalités analytiques pour l'exploitation visuelle des données.

La nouveauté de notre travail consiste à: (i) une architecture distribuée de traitement et d'analyse des concepts et l'exploitation en temps réel, (ii) la combinaison de l'ACF avec l'analyse des techniques d'exploration, y compris la visualisation des règles d'association, (iii) des nouveaux algorithmes pour condenser et filtrage des données conceptuelles et (iv) un système qui met en uvre toutes les techniques proposées, Cubix, et ses étude de cas en biologie, dans la conception de systèmes complexes et dans les applications spatiales.

ÉCOLE CENTRALE PARIS

Abstract

Business Intelligence Team
MAS Laboratoire

Doctor of Philosophy

Real-time Distributed Computation of Formal Concepts and Analytics

by Cassio Melo

Formal Concept Analysis (FCA) has been extensively used in a wide range of problems, e.g, as the basis for semantic search engines and as a means of organising information based on its meaning. FCA has also been used to mine data for groups of items and their associated transactions to identify, for example, groups of products that are frequently purchased together.

One critical issue of the traditional concept lattice visualisation is that it grows dramatically with the number of objects and attributes. On the other hand, the analysis process can be greatly enhanced with aid of visual analytics techniques.

Following a user-centered design methodology, the present thesis addresses the problem of mining and visualising concepts over a data stream. The proposed approach is twofold: First, compute formal concepts in a data stream using a distributed architecture, in order to overcome the limitations of scalability of traditional approaches. Second, we design a visual analytics interface to allow selecting, comparing, filtering, detailing and overview of concept lattice features.

We present three use cases that demonstrate the application of the proposed techniques in a real-world setting. The three use cases are: a) Aircraft cabin design, where new visualisations for continuous data helped analysts to quickly identify classes of comfort for passengers; b) Genes co-expression analysis using a combination of both analytics features and semantic integration; and c) Prediction of possible failures from telemetry data in real-time.

Acknowledgements

I would never have been able to finish this thesis without the guidance of my committee members, help from friends, and support from my family.

I sincerely thank my supervisors Marie-Aude Aufaure and Bénédicte Le Grand for their endless support and encouragement without which this research work would not have been attainable. Marie-Aude always motivated me to deliver the best and offered me many great opportunities. Bénédicte guided my through the course of my Ph.D. with brilliant ideas and discussions. I felt encouraged every time I attended her meeting.

I woud like to express my greatest gratitude to professors Nizar Messai and Anastasia Bezerianos for their useful comments, remarks and engagement through the learning process of this thesis.

Thanks to everyone who I interacted with at École Centrale, all MAS-BI members, staff and Centrale Recherche S.A., for their support and worthy contributions during the course of my studies.

I have spent part of my Ph.D studies at UCSD - databases lab, where I had an great experience as a researcher. I would like to thank professor Yannis Papakonstantinou, Kian Win and the FORWARD team for the opportunity, knowledge and pleasant moments we shared together.

I would like to thank the members of the CUBIST project (Combining and Uniting Business Intelligence with Semantic Technologies). They have been very enthusiastic of this work and made it possible.

I would like to thank my loved ones, who have supported and encouraging me with their best wishes. A special thanks to Dina, who reviewed this work over and over giving me excellent suggestions.

Last but not least, I would like to acknowledge the European Commission 7th Framework Programme of ICT for funding the integrity of my Ph.D thesis.

Contents

List of Figures

List of Tables

Abbreviations

BI	**B**usiness **I**ntelligence
CET	**C**losed **E**numeration **T**ree
CFI	**C**losed **F**requent **I**temset
FCA	**F**ormal **C**oncept **A**nalysis
FP	**F**requent **P**attern
FIM	**F**requent **I**temsets **M**ining
HCI	**H**uman **C**omputer **I**nteraction
MV	**M**ulti **V**alued
MVVM	**M**odel–**V**iew–**V**iew**M**odel
OLAP	**O**n**L**ine **A**nalytical **P**rocessing
RDF	**R**esource **D**escription **F**ramework
RT	**R**eal-**T**ime
SFCA	**S**imilarity **F**ormal **C**oncept **A**nalysis
SPARQL	**S**PARQL **P**rotocol and **R**DF **Q**uery **L**anguage
UCD	**U**ser **C**entered **D**esign
UI	**U**ser **I**nterface

Chapter 1

Introduction

1.1 Introduction

Last century was pivotal in the course of humanity. World War I and World War II were the deadliest moments in history, yet these were notable landmarks in the development of new technologies. The need of secure transmission of information motivated the emergence of the field today known as *cryptography*. The first cryptographic machine, *Enigma*, was invented by a German at the end of World War I and used by Britain's codebreakers as a way of deciphering German signals traffic during World War II. It has been claimed that thanks to the information gained through this machine, the war was reduced by two years.

Computing technology quickly scaled in importance during and after WWW I and II, a period when a number of breakthroughs were conceived, such as the transistor in early 1950s, space exploration missions between 1957 and 1975, personal computers with Apple I in 1976, to name a few. Great computing power and new techniques fostered innovations in virtually all areas, from military to science, government, manufacturing, agriculture, and even literature. In the late twentieth century, the world saw the rise of Capitalism, an economic system based on capital accumulation. Capitalism's focus on economic growth kept the wheels of technology turning at even greater pace. As Capitalism had started to spread more widely by the end of the last century, almost the entire world shared one ambition: *produce* more to *sell* more and *produce efficiently* to *cost less*. A new world, with an number of possibilities ever experienced by humanity.

We now live in a age of acceleration with a torrent of inventions, devices and interconnectedness. The advance of technology is a by-product of humans eagerness to consume, accumulate wealth and live longer. Understanding the forces that drive these innovations is a preamble to any modern scientific research. Moreover, dealing with the huge volume of information generated daily has become a challenge.

Humans deal with information complexity by seeing patterns, using their imagination, experience and intuition. Computers can only deal with numbers as they have no intuition, as commonly thought. The idea that machines could outstrip humans intellectually is a question that long appeared in science fiction. A range of technologies that enhances human cognition are used to discover patterns, predict trends; they have allowed us to take *better* and *fast* decisions, in ways that were previously unachievable.

Information has become the umbrella resource of many applications to a point where unimportant information prevails over meaningful information by a large margin. In this context, Business Intelligence (BI) emerged as a discipline that aims to take all relevant, available data and convert it into knowledge. BI uses technologies and applications to gather, represent and analyse data to support decision making.

The greater computing power achieved by recent distributed computation techniques made it more challenging to extract relevant information. Although the field of data mining is being heavily investigated, there are still few methods able to convey a large volume of stream data in meaningful insights.

1.2 Motivation

For each produced bottle of milk, there are roughly 2,000 transactions involved in the whole process from harvesting to distribution. The whole milk, once approved for use, is pumped into storage silos where it undergoes pasteurization, homogenization, separation and further processing. In each phase, hundreds of transactions are produced and serve as input for the subsequent phases. A transaction in this case may contain information about the operation performed, current state, history, nutritional information, batch, validity, and other domain specific properties (FAO, 1999).

In other contexts, WalMart records 20 million sales transactions in a single day, whereas Google handles 150 million searches, and AT&T produces 270 million call records. The figures also hold for an increasing number of small-sized institutions that are generating more data, more *often*. It is not only important to process huge amounts of data stream, but do *fast* and *reliably*.

Data in transactions are an increasingly part of our daily lives and there is a growing need to extract relevant information from them. Facebook reached 4 billion active users as of 2011. Youtube claims they reached the same number by 2013. Although being known for their huge IT infrastructure, none of theses services can deal effectively with the entirety of data their users produce everyday. Facebook's CEO Mark Zuckerberg has admitted that they were unable to answer a simple question such as "*Who was the 4 billionth user to sign up?*". Like most companies, sampling is Facebook's best bet.

> *"Doing data analytics at this scale is a big challenge, and one of the things you have to do is sample"* (BusinessWeek, October 04 2012).

Distributed computation techniques such as Map/Reduce along with powerful frameworks like *Hadoop*[1] have helped fill this gap by employing a *divide-and-conquer* approach: big problems are split into smaller parts that can be computed concurrently and partial results are merged at the end of the computation.

Despite the great computing opportunities these frameworks provide, it has only increased the need for means of identifying and understanding relevant information. Modern support decision programs have shift from purely "number crunching" methods in favour of more semantic approaches.

Formal Concept Analysis (FCA) may play an important role in this context, by employing more intelligent means in the analysis process. FCA became popular in the early 80's as a mathematical theory of data analysis based on the philosophical notion of a concept and concept lattice (Ganter and Wille, 1999).

In FCA, concepts are formalized as groups of objects (representing studied elements) associated with groups of attributes, which are the objects' features. Hierarchical relationships between these groups are formed and visualized and can be used for information

[1] Apache Hadoop. http://hadoop.apache.org

retrieval. Information organized in this way has a close correlation to human perception and the combination with visualisation is therefore interesting for Business Intelligence. FCA has been used as the basis for semantic search engines (Ducrou et al., 2008) and as a means of organising information based on its meaning (Becker and Correia, 2005). FCA has also been used to mine data for groups of items and their associated transactions (Lakhal and Stumme, 2005) to identify, for example, groups of products that are frequently purchased together. Other use case examples of FCA include:

- Marketing: finding groups of customers with similar behaviour given a large database of customer data containing their properties and past buying records;
- Biology: hierarchical classification of plants, animals, genes, given their features;
- Internet: document classification; clustering *clickstream* data to discover groups of similar access patterns and creating recommender systems;
- Search: Google's algorithm predicts and displays search queries based on other users search activities.
- E-Commerce: Amazon Recommendation, e.g. customers who bought digital camera, may be interested in a memory card or extra battery.

Formal concept mining is a computationally intensive task and the vast majority of existing algorithms do not take advantage of concurrent processing techniques (Krajca et al., 2008). In the case of data stream mining, this becomes even more important as data stream algorithms should not block their process when new data arrives in the stream.

Another gap in the application of FCA to Business Intelligence concerns visual analytics. In FCA, the hierarchical relationships between concepts are displayed in a structure called *concept lattice*, which is traditionally represented by a static node-link diagram called *Hasse diagram* (Ganter and Wille, 1999). The concept lattice visualisation can be greatly enhanced by visual analytics features and interlinked with best practices from known BI visualisations. Visual analytics features may be used to manage and navigate the complex concept interrelationships, by condensing and clustering the results, and by sub-dividing and filtering data. Today, only a small number of tools are able to deal

with lattice visualisation and the support for interactive analysis is limited (Carpineto and Romano, 2004a, Eklund et al., 2010, Bach, 2010, Kuznetsov et al., 2007).

1.3 Current Challenges

Business Intelligence traditionally focuses on storing large amounts of structured data into a data warehouse and makes use of OLAP (OnLine Analytical Processing) to provide multi-dimensional analytical queries capabilities.

Two new trends, Semantic BI and Real-time BI, are emerging in order to deal with unstructured data and real-time reporting, respectively (see Figure 1.1 for a comparison of traditional BI, Semantic BI and Real-Time BI life cycles). Semantic BI employs the RDF conceptual model instead of traditional "star schema" approaches. It transforms and stores a wide variety of data types in RDF triplestores (information warehouse) and implicit facts can be inferred within the database using sound logical rules. The semantic analytics deals mostly with *qualitative* information and usually relies on graph-based visualisations.

Real-time BI is designed to provide computation results over a stream as soon as they become available. In Real-time BI, the processing runs indefinitely as the system aggregates statistics every time new data arrives in the stream. Storage of computed information is usually kept in memory allowing fast read/write of data.

Real-time FCA mining over data stream poses several theoretical and practical challenges:

Computational Complexity. Best performing FCA algorithms such as In-Close2 (Andrews, 2011), FCbO (Outrata and Vychodil, 2012a), Norris (Norris, 1978) and Bordat (Bordat, 1986) have polynomial computational complexity regarding the number of objects or attributes in a dataset. In contrast, lookups in stream mining must have *linear complexity* at most in order to avoid the lack computational resources. To achieve low time-complexity, the stream mining algorithm should make use of techniques such as *caching, pipelining* and *hash checking*, to store and use results of previous computation.

Memory Space. A stream mining algorithm needs to guarantee that data will fit in current storage space. In worst case, mining concepts generates 2^M itemsets for

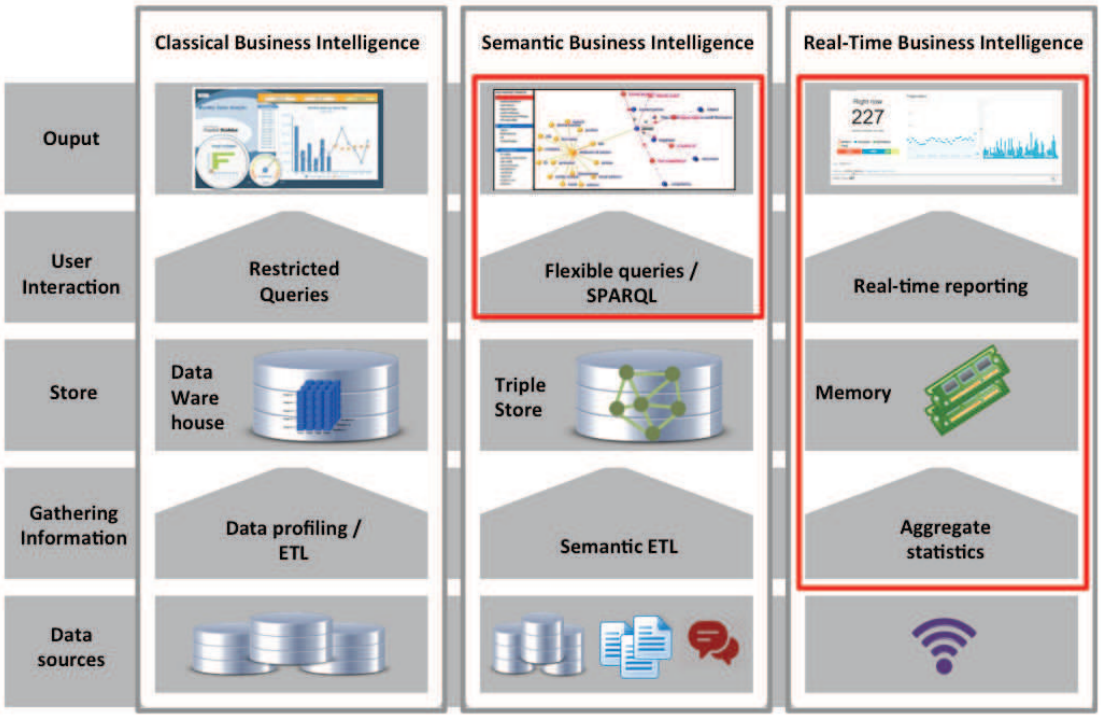


FIGURE 1.1: Lifecycle of different Business Intelligence paradigms: Classical BI; Semantic BI and Real-time BI. The scope of this thesis is highlighted in red.

each data stream of size M, thus the memory space requirement may quickly scale. Some computing models have been proposed to limit the computation to portions of the stream, like the *sliding-window* model. Although the model enables monitoring gradual changes in the data stream, capturing recent data and keeping the size of memory under control is a challenging task.

Parallelism and Distribution. There is an increasing need for parallel and distributed algorithms to deal effectively with *big data* analysis. Parallel algorithms avoid blocking the process (and thus new data) when a concept is being computed. Very few algorithms in FCA are designed to run in parallel and carry the needed changes *incrementally* rather than in *batch* when a data arrives in the stream. In a distributed environment, this adds extra challenges: *resource competition* between nodes, *fault-tolerance*, and *security* are some of them.

Real-time User Feedback. A stream mining algorithm runs indefinitely when deployed, therefore results of the current computation should be promptly available at user's request. The real-time system should push updates to user's interface without

further request. For instance, in case of monitoring systems, analysts are usually concerned with data within the most recent window of time (current day, last few hours, etc).

Interactive Visualisations. User interaction with data is closely related to its visual representation. Interactive visualisation technology displays numerous aspects of multidimensional data using interactive pictures and charts. The colour, size, shape and motion of objects represent the multidimensional aspects of the data. Card et al. (1999) defined information visualisation as "*the use of computer-supported, interactive, visual representations of abstract data to amplify cognition*". A good visual representation can amplify user cognition by providing more information, faster, with less cognitive effort. BI platform vendors are currently promoting these technologies as an alternative and enrichment to traditional reporting and online analytical processing capabilities (Andrews et al., 2008).

The objectives of this thesis are presented in the following Section.

1.4 Goals

The goals of this thesis are two-fold: Firstly, to provide **distributed scalable real-time computing of formal concepts over data stream**. Real-time computation in FCA goes beyond the aggregate functions used in traditional batch processing of concepts and poses several challenges, as explained in the previous Section. In particular, high-performance computing and low memory footprint will be required to deal with large data streams.

Secondly, to develop **new visual analytic features** to navigate in complex concept interrelationships, by going beyond the standard approaches used for FCA hierarchical structures visualization, and by interlinking with best practices from known BI visualizations. In particular, by adding new features such as condensing and clustering the results, and sub-dividing and filtering data.

The the proposed solution is outlined in next section.

1.5 Overview of the Proposed Solution

In order to meet the first goal, a novel distributed approach for mining formal concepts over data streams is proposed. It computes and maintains closed itemsets incrementally and can return the current closed frequent itemsets in real time on user's request. The approach is comprised of several components that carry out the computation of concepts from a basic transaction, filter and transform data, store and provide analytic features to visually explore data.

To meet the second goal, a visual analytics tool for FCA is implemented, called **Cubix**. The techniques implemented in Cubix allow selecting, comparing, filtering, detailing and overview of concept lattice features. Cubix's workflow allows users to carry out an analysis starting from a real data set, converting it into a formal context, simplifying the context to make it manageable, and visualizing the result as a concept lattice.

Another way of providing a different perspective of the conceptual structure is to represent it as a tree rather than as a lattice. Transforming a concept lattice into a tree can thus be useful for navigating in large lattices. In the experiments, statistically motivated criteria were explored to evaluate single parent concepts in the *tree transformation* process.

Our approach was tested in three real-world use cases. The three use cases are: a) Aircraft cabin design, where new visualisations for continuous data helped engineers to quickly identify classes of comfort for passengers; b) Genes co-expression analysis using a combination of both analytics features and semantic integration; and c) Prediction of possible failures from telemetry data in real-time.

Contributions. The novelty of the work consists of: (i) a distributed processing and analysis architecture for mining concepts in real-time; (ii) the combination of FCA with visual analytics and exploration techniques, including association rules analytics; (iii) new algorithms for condensing and filtering conceptual data and (iv) a system that implements the proposed visual analytics techniques, called Cubix, and its use cases in Biology, Complex System Design and Space Applications.

This work is part of a three year FP7 funded project CUBIST[2], which aims at uniting Semantic Technologies and Business Intelligence in order to facilitate analysis of large volumes of structured and unstructured data.

1.6 Thesis Organisation

The remainder of this thesis is organised as follows.

Chapter 2 provides an overview of FCA algorithms and their place in modern computing systems. This chapter defines what formal concepts are mathematically and describes the main concept mining algorithms. It also describes the challenges when mining concepts in the light of big data applications.

Chapter 3 describes the proposed approach for the real-time distributed computation of formal concepts. It provides an overview of the main paradigms and data structures used for frequent itemset pattern mining over data stream. The frequent itemsets mining algorithms we examine are based on *candidate generation*, *batch algorithms*, *closed frequent itemset* (CFI) and *CFI over a sliding window model*. The distributed architecture of the proposed approach is built on top of a real-time platform called *Storm*. At the end of the chapter, benchmarks of the approach compared with current state of the art algorithms are presented.

Chapter 4 presents the Visual Analytics techniques developed in this thesis. The chapter starts by providing an overview on the current state of art of concept lattice drawing algorithms. New proposed visualisations are explained such as the *concept-matrix* and the *concept-sankey* visualisation. Several analytics features that allow filtering, selecting, searching and clustering data are also described. Finally, a new approach for transforming concept lattices into trees is presented, in order to facilitate lattice browsing in large contexts. We finish the chapter with a case study in the web tourism domain.

We introduce the use cases for our approach in Chapter 5. It describes Cubix, an analytics tool for Formal Concept Analysis that implements all the visual analytics features presented in Chapter 4. In Chapter 5 we present three use cases that demonstrate the application of the approach in real-world settings. The three use cases are: a) Aircraft

[2]Combining and Uniting Business Intelligence with Semantic Technologies - www.cubist-project.eu

cabin design, where new visualisations for continuous data helped analysts to quickly identify classes of comfort for passengers; b) Genes co-expression analysis using a combination of both analytics features and semantic integration; and c) Our distributed approach was used to the prediction of possible failures from telemetry data in real-time.

Finally, a summary of the contributions of this thesis is presented in Chapter 6, along with the conclusions and future work.

Appendices A to B contain other material that you will find helpful as you read this thesis.

Chapter 2

Background on Formal Concept Analysis

2.1 Introduction

Formal Concept Analysis (FCA) is a field of mathematics based on a formalization of the philosophical notion of concept and concept hierarchy. In a seminal paper from Rudolf Wille in 1982, "Restructuring Lattice Theory" (Wille, 1982), Wille creates the basis of FCA upon previous works on *applied lattice theory* and *order theory* that emerged in the 30's. He advocates for a formal interpretation basis that would allow a rational communication. In the popular book "Formal Concept Analysis: Foundations and Applications" (Ganter and Wille, 1999), the following quote expresses the ambitions of FCA community:

> *The aim and meaning of Formal Concept Analysis as mathematical theory of concepts and concept hierarchies is to support the rational communication of humans by mathematically developing appropriate conceptual structures which can be logically activated. (p.2)*

The notion of formal conceptual logics for communication is not new, though. It draws its roots from *Propositional Logics* which seeks to define a formal system to assess the validity of statements while avoiding the bias of natural language. The development of

the modern so-called "symbolic" logic from mid-nineteenth century is one remarkable event in paving the ground for FCA as we know it.

An important aspect of Formal Concept Analysis, in contrast to its philosophical notion, is that it relies on a *context* rather than *reality*. Therefore, the real-world meaning of a concept lattice is subject to a particular interpretation depending on the context. One implication is that concepts may not have an explicit correspondence between different contexts. Some effort has been put to make FCA constructs flexible enough to comply with different degrees of interpretation (see Fuzzy FCA Quan et al. (2004)).

Nowadays, FCA find its practical applications in a wide range of problems, e.g., as the basis for semantic search engines (Ducrou et al., 2008) and as a means of organising information based on its meaning (Becker and Correia, 2005). FCA has also been used to mine data for groups of items and their associated transactions (Lakhal and Stumme, 2005) to identify, for example, groups of products that are frequently purchased together.

2.2 Terminology and Formalism

In the following, FCA terminology used in this thesis will be introduced. We used the same terminology employed in the original work by Ganter and Wille (1999) where G denotes the set of objects (in German: *Gegenstände*) and M the set of attributes (in German: *Merkmale*). More details can be found in monographs (Ganter and Wille, 1999) and (Carpineto and Romano, 2004b).

In classic FCA, data is converted into a binary matrix called formal context through a process involving *discretisation* and *booleanisation*. *Discretisation* (Jin et al., 2007) refers to the process of converting continuous data into sets of intervals or classes, e.g. age from 18 to 25, 26 to 50, etc, whereas data *booleanisation* (Imberman et al., 1999) involves creating a binary relation with pairs of attribute-values for each attribute's value. This process is known as "*Scaling*" and can be seen as a type of *binned aggregation*.

In mathematical terms, a formal context is defined as a triple $\mathbb{K} = (G, M, I)$, with G being a set of objects, M a set of attributes and I a relation defined between G and M. The relation I is understood to be a subset of the cross product between the sets it

relates, so $I \subseteq G \times M$. If an object g has an attribute m, then $g \in G$ relates to m by I, so we write $(g, m) \in I$, or gIm.

Definition (Derivation Operators). For a subset of objects $A \subseteq G$, a derivation operator $\uparrow$ is defined to obtain the set of attributes, common to the objects in A, as follows:

$$A^{\uparrow} = \{m \in M \mid \forall g \in A : gIm\}$$

In a similar manner, for a subset of attributes $B \subseteq M$, the derivation operator $\downarrow$ is defined to obtain the set of objects, common to the attributes in B, as follows:

$$B^{\downarrow} = \{g \in G \mid \forall m \in B : gIm\}$$

Operators $\uparrow: 2^X \longrightarrow 2^Y$ and $\downarrow: 2^Y \longrightarrow 2^X$ form the so-called *Galois connection* (Ganter and Wille, 1999). We may use the single quote symbol ' as a replacement for $\uparrow$ and $\downarrow$. The double derivation operator $\uparrow\downarrow$ ("") is a closure operator. The images of a closure operator are *closed* sets.

Definition (Formal Concept). A pair $\langle A, B\rangle$ is a *formal concept* in a given formal context (G, M, I) only if $A \subseteq G$, $B \subseteq M$, $A^{\uparrow} = B$ and $B^{\downarrow} = A$. The set A is the extent of the concept and the set B is the intent of the concept. A formal concept is, therefore, a closed set of object/attribute relations, in that its *extension* contains all objects that have the attributes in its *intension*, and the *intension* contains all attributes shared by the objects in its extension.[1] The Table 2.1 shows an example of a formal context of some living organisms in the water.

For example, w.r.t. the context in Table 2.1, {has limbs}' = {bream, frog, dog} is not a formal concept because {bream, frog, dog}' = {needs water to live, can move, has limbs}. Notice that formal concepts forms maximal rectangles in the incidence matrix.

A partial order $\leq$ can be established between concepts *iff*:

$\langle A, B\rangle \leq \langle C, D\rangle \longleftrightarrow \langle A, B\rangle \subseteq \langle C, D\rangle$

[1] Formal concepts are also known as closed itemsets. The data mining community often uses the word "item" for what we call "attribute" here. "Itemsets" are thus just the subsets of M.

TABLE 2.1: Example of a formal context.

Life in Water	needs water to live	lives in water	lives on land	needs chlorophyll	dicotyledon	monocotyledon	can move	has limbs	breast feeds
fish leech	×	×					×		
bream	×	×					×	×	
frog	×	×	×				×	×	
dog	×		×				×	×	×
water weeds	×	×		×		×			
reed	×	×	×	×		×			
bean	×		×	×	×				
corn	×		×	×		×			

The set $\mathbb{K} = (G, M, I)$ together with $\leq$ form a complete lattice whose structure is described by the Main Theorem of Formal Concept Analysis (Ganter and Wille, 1999).

Property (Concept Lattice). Let $\mathcal{C}_\mathcal{K}$ be a set with all concepts in context $\mathbb{K}$ and $\mathcal{L} = \langle \mathcal{C}_\mathcal{K}, \leq_\mathcal{K} \rangle$ be a complete lattice where the *supremum* and *infimum* are defined respectively as:

$$\bigvee_{i=1}^{n} (A_i, B_i) = ((\bigcup_{i=1}^{n} A_i)'', \bigcap_{i=1}^{n} B_i) \text{ and}$$
$$\bigwedge_{i=1}^{n} (A_i, B_i) = ((\bigcap_{i=1}^{n} A_i, (\bigcup_{i=1}^{n} B_i)'')$$

2.3 Concept Lattices

As mentioned above, FCA analysis produces lattices, usually represented as layered directed acyclic graph graphs, named **Hasse diagrams** illustrating the groupings of objects described by common attributes. Hasse diagrams display the partially ordered sets (posets) between concepts in a hierarchical fashion, where each concept may have several parent concepts. The following concept lattice about organisms that live on the water was generated by the formal context in Table 2.1 (Figure 2.1). The partial order among concepts of the lattice is materialized through the generalization and specialization relationships. For instance, the concept representing the set of animals that "has limbs" (middle left) is more specific than the concept "can move" (upper left), in other

words, all animals that "has limbs" can move, but not all animals that "can move" has limbs, which is the case of "fish leech"[2]. This partial order provides different levels of abstraction and native navigation links from a given concept.

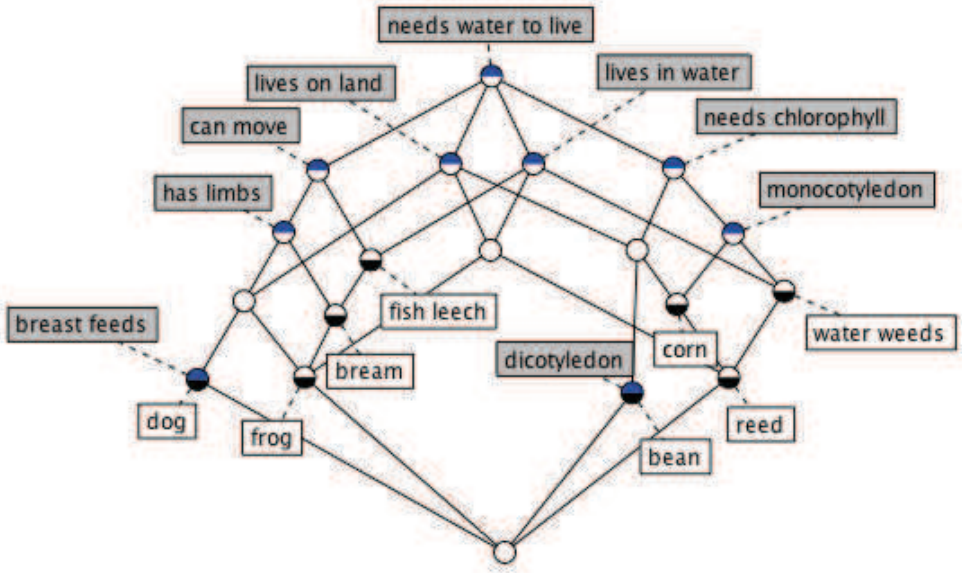


FIGURE 2.1: Concept lattice for the life in the water context. Dark background labels represent attributes, white background labels represent objects.

Concept lattices in particular suffer from considerable edge crossings, especially if the number of concepts exceeds a few dozen as is the case in more real word applications (Kuznetsov et al., 2007), which leads to reduced graph readability and aesthetics (Ware et al., 2002) (Figure 2.2).

To reduce the complexity of lattices, simplified diagrams can be produced by condensing or clustering concepts according to similarity (Gerd et al., 2002a). Visualisations can also be restricted to portions of the data (Ducrou et al., 2008), and concept size reduction is possible by incorporating conditions into the data mining process (Zaki and Hsiao, 2005). Finally, conceptual measures can be applied to identify the most relevant concepts and filter outliers (Le Grand et al., 2009).

To deal specifically with the visual complexity of Hasse diagrams, several approaches allow users to dynamically explore and reveal specific parts of the diagram, using visual

[2]Statement valid for the particular formal context in question.

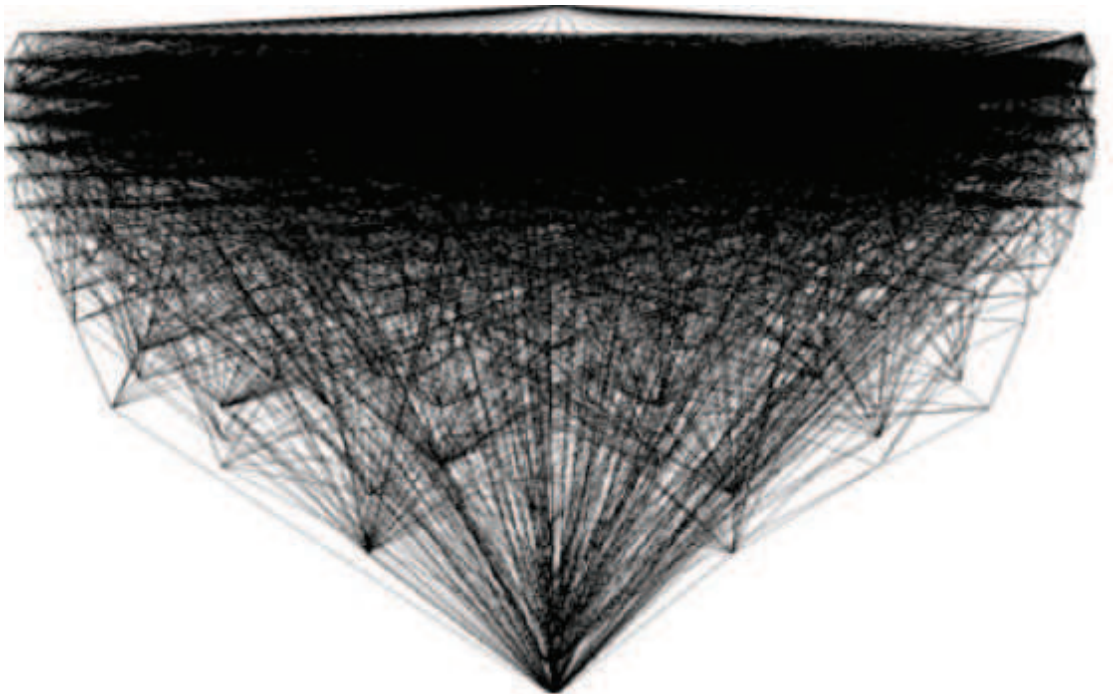


FIGURE 2.2: Concept lattice containing 83 concepts.

query languages (Blau et al., 2002, Cruz et al., 1987, Consens and Mendelzon, 1993). However these techniques do not provide a clear view of the entire lattice.

Other FCA visualisation approaches map the distances between concepts to visual variables, in order to highlight patterns. For example in Michel Soto (2009) similar concepts are represented as similarly coloured pixels placed in the 2D space along a Peano-Hilbert curve, so that similar concepts are placed close to each other. Nevertheless, in these representations detailed relationships between concepts are lost.

Finally, systems often provide users with hybrid/combined lattice visualisation, e.g. showing both a general Hasse diagram and a tag cloud for representing the neighbours of a specific concept (for a review see Eklund and Villerd (2010)).

There are many algorithms for drawing such lattices, additive-line diagrams, hybrid layouts and even 3D layouts have been proposed. In Chapter 4 we will discuss them in detail.

2.4 Overview of FCA algorithms

A naive algorithm for computing concepts would check if every combination of subsets $B \subseteq M$ is a closed itemset. This would require an exponential number of lookups in a list that may have exponential size[3]. Instead, FCA algorithms typically employ some heuristic to verify if a given subset is worth of computation. This can be done for example, using extra canonicity tests, defining an ordering between subsets, using data structures for storing previous results, etc.

The algorithms in FCA are basically divided in three classes: The **batch** algorithms, which procedure takes into account the entire context for batch processing; **Incremental** algorithms, which build the concept lattice from each object in the context and update the structure as a new object is added; and the **assembling** algorithms derive new concepts from previously calculated subsets (for a survey see (Kuznetsov and Obiedkov, 2002)). In the following sections, we describe the mainstream algorithms in each category.

2.4.1 Batch Algorithms

Batch algorithms generate sets of concepts from scratch, in a top-down fashion (from minimal to maximal intents) or vice-versa. Bernhard Ganter (Ganter, 2010) introduced the idea of defining a lexicographic order for subsets in order to avoid multiple computations of the same concept. A lexicographic order relation $A \preceq B$ exists if there is some lexically-ordered element $e \in A$ that does not belong to B. For example, $\{a, c, d\}$ comes before $\{a, c, e\}$ because d is the smallest item that belongs to A but not B. This is a common technique among FCA algorithms.

Ganter's **NextClosure** algorithm (Ganter, 2010) works by finding closures incrementally from a lexicographically ordered group of items. The generation of a concept is considered canonical if its extent contains no object preceding the current object. The procedure is illustrated in Algorithm 1.

NextClosure takes as input a subset $B \subseteq M$ of the *lexicographic next closure set*. It starts from the last (lexicographic) attribute (line 4) and checks if attribute is not already in

[3]For a set with N elements, there exists 2^N possible subsets (powerset).

Algorithm 1: NextClosure

Data: A closed subset B such that $B \subseteq M$.

```
1  begin
2      /* Print or list < B', B > as a formal concept                    */
3      print(< B', B >)
4      i ⟵ |M|
5      success ⟵ False
6      while not success and i > 0 do
7          i ⟵ i − 1
8          if not B[i] then
9              D ⟵ B ∪ {i}
10             C ⟵ D''
11             if C \ B contains no element < i then
12                 NextClosure(C)
13                 success ⟵ True
14             end
15         end
16     end
17 end
```

the subset (line 11). It then creates a new pair $(C, D) = ((B \cup \{i\})'', B \cup \{i\})$ (lines 9-10). If the canonicity test[4] passes (line 11) the algorithm selects the pair as concept candidate and continues the execution with the new concept as input (line 12). In the example, the algorithm implemented is recursive.

Ganter's algorithm computes the set of all concepts in time $O(|G|^2|M||L|)$, where $|L|$ is the size of the concept lattice, and has polynomial delay $O(|G|^2|M|)$.

Another class of algorithms has a slightly different way of checking concepts: the **Close-by-One (CbO)** familly of algorithms (Krajca et al., 2010). It is possible to determine if a concept is new by efficiently examining its canonicity. The original CbO procedure is illustrated in Algorithm 2.

The procedure takes as input a formal concept $\langle A, B\rangle$ and an attribute $y \in M$ (first attribute to be processed) as its arguments. The procedure recursively descends through the space of formal concepts, beginning with $\langle A, B\rangle = \langle \emptyset', \emptyset''\rangle$. In line 3, the algorithm checks the halting condition: when the least formal concept has been reached or $y > |M|$, i.e., there are no more remaining attributes to be processed. If the halting condition test fails, the algorithm goes through all attributes $j \in Y$ such that $j >= y$ and j is not in

[4]The canonicity test checks if a given concept is closed, in that *none* of its immediate supersets has the name support as the current concept.

Algorithm 2: Close-by-One

Data: A concept $< A, B >$ such that $A \subseteq G$ and $B \subseteq M$ and an attribute $y \in M$.

```
1  begin
2      /* Print or list < A, B > as a formal concept                    */
3      print(< A, B >)
4      if B = M or y > |M| then
5          return;
6      end
7      for j ← y to |M| do
8          if j ∉ B then
9              C ⟵ A ∩ {j}′
10             D ⟵ C′
11             if B ∩ M_j = D < ∩M_j then
12                 CloseByOne(< C, D >, j + 1)
13             end
14         end
15     end
16 end
```

the intent B (lines 5 and 6). A new formal concept is then created (lines 9-10) and the canonicity test is performed with $M_j = \{m \in M | m < j\}$ (line 11). If the new concept passes the canonicity test, the procedure uses it in the next recursive call. Otherwise, the concept is skipped and the procedure continues with the next attribute in the order.

Although CbO's complexity is $O(|G|^2|M||L|)$ in worst case, in average it performs better than NextClosure because it prunes computation steps with its canonicity test.

An improvement over the original CbO was proposed in (Outrata and Vychodil, 2012a) called **Fast Close-by-One** or **FCbO**. The algorithm used an extra canonicity test to avoid redundant computations. The result is equivalent to a pruned CBo recursion tree where the number of comparisons is drastically reduced.

The new canonicity test takes advantage of the fact that the failure in the canonicity test in line 11 of CbOs algorithm can be propagated to sub-nodes and thus eliminating the need of extra computation. Because this information must be propagated in the top-down direction, from the root node of the call tree to the leaves, the algorithm uses a breath-first search strategy as opposed to the recursive calls of the original CbO. The algorithm and proof of its correctness can be found in (Outrata and Vychodil, 2012a).

Notice that in the worst case, FCbO collapses into CbO (all branches are computed) with an additional linear time-delay overhead introduced by the new canonicity test.

In this scenario (worst case) FCbO has polynomial time-delay $O(|G||M|^3)$. However, heuristically it performs better than the previously mentioned algorithms because the pruning happens frequently (A performance comparison can be found in Outrata and Vychodil (2012b)).

Parallel and distributed versions of FCbO were proposed (Krajca et al., 2008). We will discuss them in Section 2.5.4.

Another variant of CbO is the **In-Close** algorithm (Andrews, 2009). It uses incremental closure and matrix searching to quickly compute all formal concepts in a formal context. While in CbO the closure is computed at each iteration, In-Close on the other hand, completes closure incrementally and only once per concept, as it iterates across the attributes. As a result the test of canonicity requires iteration over a relatively small portion of the formal context.

A new version of In-Close was proposed in (Andrews, 2011). In **In-Close2**, attributes tested in parents are propagated down in the tree, so that during the closure of a child concept these attributes do not need to be tested for inclusion. In-Close2 also reorders the context table in a way that maximal rectangles formed by the incidence relation are grouped for performance gain. Experiments show that in average, In-Close2 outperforms FCbO by a small margin (Andrews, 2011).

Whilst batch FCA algorithms can perform better in average than the other classes of algorithms, it requires several lookups in the entire dataset. Any change in the context table means that the algorithm should re-compute all items again. As we will see further in this chapter, this become an issue as a growing number of applications have dynamic datasets.

2.4.2 Incremental Algorithms

The incremental paradigm aims at carrying efficiently the required changes of the current result leading to its updated version. The incremental update of an existing lattice $\mathcal{L}$ starts from a single object $o_i \in G$, and progressively incorporates any new object o_{i+1} upon its arrival in the lattice, carrying out a set of structural updates. A good incremental algorithm will make the necessary changes at a minimal cost.

An incremental version of the CbO algorithm is presented in (Norris, 1978), commonly referred to as **Norris' algorithm**.

In Godin's algorithm (Godin et al., 1995) lattice construction can be described in terms of four sets of concepts: *modified concepts*, *generator concepts*, *new concepts* and *old concepts*. A concept $\langle C, D\rangle \in$ lattice $\mathcal{L}_{i+1}$ is new if D is not an intent of any concept in $\mathcal{L}_i$. We call a concept $\langle A, B\rangle \in \mathcal{L}_i$ modied if $B \subseteq g$ since g has to be added to its extent in $\mathcal{L}_{i+1}$ (lines 19-22). Otherwise, $B \cap g = D = B$ for some concept $\langle C, D\rangle \in \mathcal{L}_{i+1}$ (line 24). If $\langle C, D\rangle$ is a *new concept*, then $\langle A, B\rangle$ is called a *generator* of $\langle C, D\rangle$; if not, $\langle A, B\rangle$ is called *old* (lines 25–30).

To decrease the number of comparisons, the algorithm uses a hash with the intent's cardinality in its key and a corresponding list of subsets with that cardinality (line 14). The execution goes through the subsets with lower cardinality, checking if a concept needs to be updated and carrying out the changes (lines 17–30).

In (Merwe et al., 2004) authors presented the **AddIntent** algorithm. The algorithm uses new heuristics to identify all modied concepts (in order to add o to their extents) and all canonical generators of new concepts (in order to generate every new concept exactly once).

AddIntent approaches this problem by traversing the diagram graph of $\mathcal{L}_i$ in a recursive fashion. Whenever the algorithm finds a non-canonical generator, it uses the diagram graph to find the canonical generator of the same concept. It then works only with concepts above that canonical generator, ignoring all other concepts above the non-canonical generator. AddIntent's time-complexity is $O(|G|^2|M||L|)$.

Particularly interesting for dynamic contexts, the incremental algorithms are more suitable to stream processing because updating the concept lattice is less computationally expensive than building the concept lattice from scratch for each transaction in the stream. As we demonstrate in Chapter 3, we implemented a distributed algorithm *FCAStream* which draws principles from incremental algorithms in FCA.

2.4.3 Assembling Algorithms

The Assembly algorithms are an evolution of incremental algorithms in that they assemble lattices from partial structures based on context concatenation upon a shared object

Algorithm 3: Godin's incremental algorithm

Data: A new object $o \in G$ and its corresponding attributes $B \subseteq M$.

```
1  begin
2      /* Concept ⟨X, Y⟩ is the top concept in the lattice                */
3      ⟨X, Y⟩ = sup(ℒ)
4      if ⟨X, Y⟩ = ⟨∅, ∅⟩ then
5          replace ⟨X, Y⟩ by ⟨{o}, B⟩
6      else if not B ⊆ Y then
7          if X = ∅ then
8              Y ∪ B
9          else
10             create ⟨C, D⟩ ⟵ ⟨∅, (Y ∪ B)⟩
11             add link ⟨X, Y⟩ → ⟨C, D⟩
12
13     /* H is a hash with keys as cardinalities and values as a list of
           intents                                                        */
14     H ⟵ {key : |B ⊆ M|, value : list([B])}
15     Hprime ⟵ []
16     for i ← 0 to |M| do
17         foreach pair H_i in H do
18             if H_i ⊆ B then
19                 /* H_i is a modified intent                            */
20                 H'_i ⟵ H'_i ∪ {o}
21                 append H_i to Hprime
22             else
23                 newIntent ⟵ H_i ∩ B
24
25             if newIntent not in Hprime then
26                 /* H_i is a generator intent                           */
27                 create new concept ⟨E, F⟩ ⟵ ⟨newIntent, (H_i ∪ {o})⟩
28                 add link from ⟨E, F⟩ to ⟨H_i, H'_i⟩
29                 /* Change existing links to accommodate the new concept */
30                 ModifyEdges(⟨E, F⟩)
31
32         end
33     end
34 end
```

set. The only known algorithm of this family is the algorithm **Divide & Conquer** (Valtchev and Missaoui, 2001). It divides a formal context into parts (vertical or horizontal split) and then calculates the concept lattice for each corresponding part. The partial lattices are finally assembled into a single one. One advantage of this approach is that the computation of partial lattices can be done in parallel, thus making this approach one of the most *scalable* among all existing FCA algorithms.

2.4.4 General Remarks on FCA Algorithm's Performance

The behaviour of an FCA algorithm may vary significantly depending on a number of factors including the relative sizes of G and M, the size of I, and the density of the context, i.e., the size of I relative to the product $|G||M|$ (Kuznetsov and Obiedkov, 2002). Experimental results of (Kuznetsov and Obiedkov, 2002) highlight Norris (incremental), CloseByOne and NextClosure (both bottom up) algorithms as the best algorithms when the context is dense and large, whereas Godin's algorithm should be used for small and sparse contexts.

Evidently, the construction of the lattice diagram graph requires an additional computational effort. Hence, algorithms generating only the concept set are in general faster than those that carry the ordering and linking of concepts.

Most FCA algorithms employ techniques to avoid repetitive generation of the same concept. In this task, choosing an adequate data structure is essential. For instance Bordat (Bordat, 1986) uses a tree structure to store previously computed concepts and allow efficient search. Godin's algorithm (Algorithm 3) uses a hash based on intent cardinality.

Subsets of attributes can be represented by a bit-array where each bit corresponds to an attribute in lexicographic order. Set operations can be replaced by the corresponding logical operations, e.g. $A \cap B = bit\_array(A)$ & $bit\_array(B)$.

For example, if $M = \{a < b < c < d < e < f\}$, the subset $S = \{a, c, d, e\}$ will be written as:

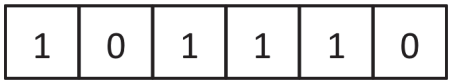


FIGURE 2.3: Bit-array illustration for the set $S = \{a, c, d, e\}$.

Some algorithms for FCA achieve better performance if attributes are processed in a particular order. The overhead of physically sorting the context is outweighed by the saving in memory load. For example, In-Close2 (Andrews, 2011) sorts context columns in ascending order of support. This allows quick memory retrieval according to the *spatial locality principle* of memory. It also sorts rows in the context to reduce the *Hamming Distance*, i.e., the number of positions in which the corresponding symbols are different in a string or, in their case, a bit-array. Experiments show that these techniques improve the algorithm's efficiency over 33% for large contexts (Andrews, 2011).

2.5 Current Issues in FCA for Big Data Analysis

In this section we discuss the challenges current FCA algorithms face in the light of big data analysis, when the input data exceeds hundreds of thousands objects/attributes.

2.5.1 Size and Complexity of Data

Since the number of concepts can grow dramatically with the number of attributes and objects, reducing the number of concepts generated has become a major challenge in the FCA community (Priss, 2000, Gerd et al., 2002b, Carpineto and Romano, 2005). Data-intensive applications like biogenetics, space telemetry data and real-time monitoring generate massive, noisy data, impractical to handle with current FCA representations.

Density and noise of a context are factors that increase the number of formal concepts (Andrews and Orphanides, 2010). In this case, relevance measures such as stability and support can minimise input data needed for the computation of concept lattices (Godin et al., 1998).

The problem of reduction in FCA is analogous to the dimensionality reduction in data mining. The assumption is that not all variables or dimensions are important or that a few set of variables are significant for understanding the phenomena of interest. However, in FCA, concepts add a new challenge: how to remove information while preserving some of the most essential conceptual features?

Reduction methods are basically divided into two strategies: those that group similar objects or concepts, and those that act by removing irrelevant concepts. Reduction can

take place in the scaling phase, for instance, reducing the columns with low support or through the creation of sub-contexts (Andrews, 2011), during the concept mining phase, e.g. keeping concepts with enough support (Gerd et al., 2002b), or in the concept lattice, through visual cluttering reduction techniques (Michel Soto, 2009).

An intuitive solution for reducing the formal context is to select sub-contexts or restrict the scaling of data to only attributes and values of interest (Cole and Eklund, 1999). For instance, it is possible to specify larger intervals for a continuous attribute such as 'date'; or group together similar objects and attributes. Kumar *et al.* (Aswani Kumar and Srinivas, 2010) use Fuzzy K-Means (FKM) clustering to collapse certain rows of a formal context, introducing equivalence relations between certain nodes of the object-attribute lattice. Another intuitive idea is to filter out irrelevant objects and/or attributes below a certain frequency threshold, thus simplifying the context table (Gerd et al., 2002b).

Many reduction methods are based on the factorisation of the binary table into some canonical form, thereby simplifying the formal context. A direct application of Singular Value Decomposition (SVD) and Non-negative Matrix Factorisation was done to reduce formal contexts in (Snsel et al., 2008). Although this method is able to 'compress' the formal context, it does not take into consideration any conceptual heuristic and it is not clear to the user which changes were made to the original context. On the other hand, the number of nodes and edges of a lattice could be significantly reduced by establishing an equivalence relationship between certain nodes of the lattice with aid of matrix factorisation methods.

Cheung and Vogel (Cheung and Vogel, 2005) introduce the notions of '*congruence*' and '*quotient*' that use SVD to define the equivalence relation for the construction of quotient lattices. The resulting context is a merge of rows or columns that maintains the homomorphism in a simplified concept lattice. These methods are very sensible to noise, meaning that minor differences in data may lead to highly biased contexts during the factorisation process. Another problem is that the computational complexity of SVD methods makes it impractical for large matrices (Aswani Kumar and Srinivas, 2010).

Alternatively, reduction can happen at concept level, during or after concepts computation. The selection of relevant concepts and groupings can be performed by the algorithm that computes concepts. For instance, the **'Iceberg' lattice** approach (Gerd

et al., 2002b) removes concepts below a certain frequency threshold, i.e. *minimum support*. One problem with the *support* is that it tends to select the most generic concepts as they are more likely to contain more objects. The *stability* index measures the proportion of subsets of objects of a given concept whose derivation is equal to the intent of this concept. In other words, *stability* indicates the probability of preserving a concept intent while removing some objects of its extent (Kuznetsov et al., 2007).

The notion of clustering is well known in the data-mining domain. It is a process of grouping together entities based on their similarities. Clustering of concepts (as opposed to clustering of objects or attributes) can be useful to facilitate the browsing of concepts and to identify zones of interest. Some conceptual similarity measures can be based on: a) the concept lattice topology (e.g. counting the number of links between two concepts); b) the intent/extent similarity (e.g. applying *Jaccard*) or c) calculating the confidence between pairs of concepts.

2.5.2 Scaling

FCA method is usually limited by the rigidity of its input format (binary data). Some works have proposed to extend it to complex data (Ferré and Ridoux, 2000, Ganter and Kuznetsov, 2001, Messai et al., 2008), among them the Similarity-based Formal Concept Analysis (SFCA) method which considers similarity to directly classify non-binary data into lattice structures called **Many-Valued Concept Lattices (MV lattices)** (Messai et al., 2008). Besides extending FCA to complex data and avoiding loss of information in transformation phases, the SFCA classification process produces MV lattices with different granularity levels which allows progressive data exploration (Messai et al., 2010).

Another approach consists in transforming continuous-valued formal context into many-valued formal context first, then on the basis of many-valued formal context, finding equivalence classes for the continuous-value contexts (Ganter and Meschke, 2009).

Scaling in a data stream is dynamic, that is, the process of scaling occurs *as new data arrives in the stream*. For example, in many cases one cannot know the distribution and possible maximal and minimal values in case of continuous attributes. A temperature sensor can transmit streams of data with current temperature, but it is not known

TABLE 2.2: A 'noisy' and a 'quiet' context

	1	2	3	4	5	6
a	×	×	×			
b	×	×	×	×		×
c			×	×		
d	×				×	

	1	2	3	4	5	6
a	×	×	×			
b	×	×	×	×		
c			×	×		
d						

a priori whether a given temperature is the possible maximum. Because of this, the minimum and maximum values need to be maintained and eventually updated for each incoming data, thus requiring the scaling process to restart.

2.5.3 Noisy and Missing Data

Quite often, a data set contains outliers or missing data. These exceptions can drastically increase the number of concepts generated (Andrews and Orphanides, 2010, Pensa and Boulicaut, 2005, Boulicaut et al., 2003). Noise is most common in real data such as surveys, logs, genes expression, when values can be biased, missing or some mistake was introduced in the data. Table 2.2 illustrates a noisy and a 'quiet' data set (Andrews and Orphanides, 2010).

Reduction of formal contexts can be achieved by removing noise or outliers from the data. The notion of **fault-tolerant FCA**[5] was introduced by Pensa and Boulicaut in (Pensa and Boulicaut, 2005) to allow a certain number of 'exceptions' to occur in a concept. Based on the idea of "free-sets" (Boulicaut et al., 2003), the method seeks to find maximal rectangles of true values in the context bounded by δ exceptions.

Andrews and Orphanides (2010) proposed a method to reduce noise based on mining a context for concepts that satisfy a minimum support (and/or stability) and then re-writing the context using only those concepts. One advantage of their approach compared to the "truncation" of formal context is that it preserves the concepts essential features at a cost of computing concepts twice.

[5]The noisy/missing data problem is often referred in the FCA literature as *fault tolerance*, although this term is commonly used in distributed systems. In this thesis, we refer to "fault tolerance" as the ability of a distributed system to keep running even when some of its computation units are shut down.

2.5.4 Parallel and Distributed Computation of Concepts

Most existing FCA algorithms were designed to work on small binary tables, with less than hundreds objects/attributes stored in the memory of a single computer. Among the few parallel and distributed algorithms, a parallel variant of FCbO was proposed, called **Parallel FCbO** (Krajca et al., 2008). The authors later proposed a distributed *map-reduce* version of the FCbO algorithm (Krajca and Vychodil, 2009). The algorithm modifies the original CbO (Algorithm 2) splitting it into two parts, one (*Map function*) responsible for the iteration over the attribute set and concept generation (equivalent to lines 1-10 of CbO's algorithm); and the other (*Reduce function*) performs the canonicity test (equivalent to line 11 of Algorithm 2). If the canonicity test passes then the input pair is passed on to the next iteration. Experiments showed that the distributed variant significantly outperforms the original version with at least five computing nodes. Since the problem of computing formal concepts can be translated to the computation of closed frequent itemsets of their intents, research in distributed frequent itemset mining can offer insights on the algorithm design. Surveys on the issues related to the distributed frequent itemset mining can be found in (Zaki, 1999) and (Kumar et al., 2010).

2.5.5 Data Stream Processing

High memory and computational requirement of FCA based algorithms prohibits their use in data stream environment. Each node of the concept lattice stores the extent (set of transactions) along with the intent (a *closed itemset*) which contributes to high memory usage. Furthermore, processing of a new transaction involves computation of the intersection of its extent with the extent of different nodes in the lattice, making it computationally expensive (Gupta et al., 2010).

2.5.6 Visualisation of Large Concept Lattices

The Hasse diagram used to represent FCA is well suitable for a small number of concepts. When it expands to a few dozens of concepts, as is the case in more real word applications, its comprehension becomes compromised. This is an issue in relation to the amount of data displayed, as well as graph readability and aesthetics (Melo et al., 2011a). In spite of this, concept lattice visualisation saw a marginal improvement over several

years, and current FCA tools are mostly limited to the Hasse diagram representation (Figure 2.4).

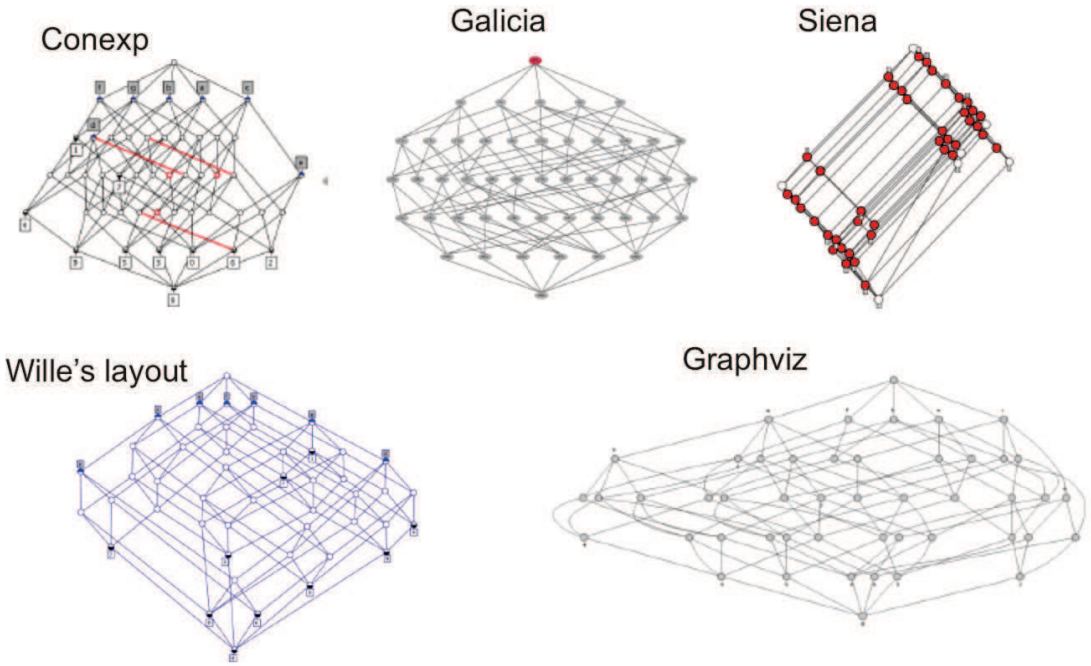


FIGURE 2.4: Tools for visualising concept lattice.

To deal specifically with the visual complexity of Hasse diagrams, several approaches allow users to dynamically explore and reveal specific parts of the diagram, using visual query languages (Blau et al., 2002, Cruz et al., 1987). Layout algorithms like the additive lines diagram (Ganter and Wille, 1999) have the advantage of being configurable through the choice of the representation set, and the technique produces a high number of parallel edges, which in turn improves readability. Nested lines diagrams (Kuznetsov and Obiedkov, 2001) have been proposed to display the direct product of two concept lattices, however, they are not always simple to interpret.

2.5.7 Visual Analytics Features

Visual analytics techniques are often used in Business Intelligence to synthesize information and provide insights in the decision making process (Keim et al., 2008). Current FCA tools lack interactive features that allow selecting, filtering, searching and transforming conceptual data. Recent studies have highlighted that the visualisation of concept lattices can be greatly enhanced with aid of visual analytics techniques.

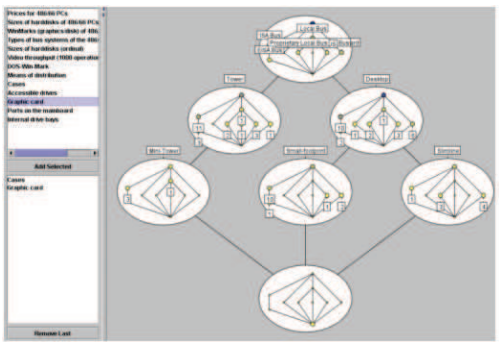
To cite a few examples, Priss (2000) use the lattice representation to show the concepts hierarchy in *thesauri*. Each concept is viewed as a facet in an information retrieval system. In (Akand et al., 2010) authors propose an algorithm that generates a browse-able concept lattice designed for biology applications. Ducrou et al. (2006) introduce ImageSleuth, a tool for browsing and searching annotated collections of images. It allows drill-up and drill-down the results and it suggests similar concepts if none was retrieved by the user's query. In (Villerd et al., 2007) authors use visual analytics in an indexed document collection to display concept lattices in a global view and concept details in a local view. The system expands/collapses information according to selections made in the concept lattice. Toscana (Becker and Correia, 2005) and ConExp[6] also use concept lattice representations but do not provide further interaction and other visualisation features (Figures 2.5(a) and 2.5(b) respectively). These tools focus on the usability of concept lattices and consider them as primary interface for interacting with a formal context. A more interactive tool, ConfExplore[7] uses animated transitions and incremental lattice exploration (Figure 2.5(c)). More recently, Bach (2010) introduces a FCA tool, called Facettice, for faceted navigation and lattice exploration for multidimensional taxonomies (Figure 2.5(d)).

In common with these studies is that they are application-specific and provide few analytic features, mostly based on drawing options and attribute/object selection. There is little or no emphasis on the knowledge discovery *beyond* what the concept lattice inherently provides. In this thesis, we focus on the combination of techniques to support analytical reasoning and insights based on concept lattices.

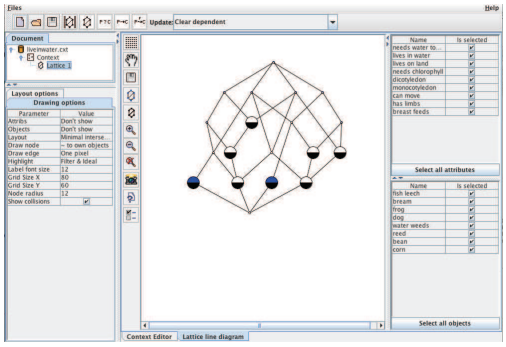
2.6 Chapter Summary

This chapter introduced FCA, its formal definitions and the terminology that will be used throughout this thesis. We investigated the mainstream algorithms in the FCA literature: batch algorithms, incremental algorithms and assembling algorithms. Each algorithm has strengths and weaknesses, and its success will largely depend on the constraints imposed by the application domain, such as dynamic data, availability of resources, concept lattice visualisation, and so on.

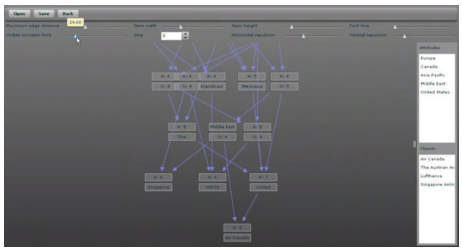
[6] http://conexp.sourceforge.com
[7] http://code.google.com/p/openfca/



(a) ToscanaJ. Nested line diagram.



(b) Conexp. Drawing options of the left, concept lattice in the center and attribute/object selection on the right.



(c) Confexplore. On top, drawing options, concept lattice in the center and attribute/object selection on the right.

(d) Facettice. On the left, the facet and value hierarchy, *Facet Lattices* on top, the concept lattice in the center and the lists of results on the right.

FIGURE 2.5: Visual Analytic tools for Formal Concept Analysis.

We listed the main issues that current FCA methods face when applied to Big Data analysis. To reduce the complexity of lattices simplified diagrams can be produced by condensing or clustering concepts. Visualisations can also be restricted to portions of the data and concept size reduction is possible by incorporating conditions into the data mining process, such as *minimum support* and *stability*. Conceptual measures can be applied to identify the most relevant concepts and filter outliers.

There is not a single all-encompassing method to solve all the mentioned problems. In general, combined approaches are required to tackle with domain specific issues. We therefore conclude that a) Incremental FCA algorithms are more suitable to handle stream data; b) There are few FCA algorithms able to run in a parallel or distributed environment, and c) Synthetic and meaningful representations for concept lattices are needed.

Chapter 3

Real-time Distributed Computation of Formal Concepts

3.1 Introduction

A data stream is an sequence of items that arrive in timely order. As opposed to data in static databases, data streams are unbounded, infinite and data distribution may vary with time (Chi et al., 2006). In BI, stream concept analysis help users both qualitatively (co-occurrence among items, classes and hierarchies) and quantitatively (number of co-occurrences, strength of the implication, distribution of sub-items, etc). Computing formal concepts can be seen as computing the *closed frequent itemsets* of their intents. Frequent itemsets can also be used to derive *Association Rules*, which provide a valuable information on how the frequent patterns are related to each other. There is a growing number of applications in association rule stream mining, for example, to predict frequency estimation of Internet packet streams (Demaine et al., 2002); to discover alarming incidents from data streams (Cai et al., 2004); or to estimate missing data in sensor networks (Halatchev and Gruenwald, 2005).

Although the field of data stream mining is being heavily investigated, there is still a lack of a holistic approach for mining closed itemsets from very large data streams. The bottleneck arises due to the limit in computational resources for stream mining, while most algorithms cannot run in parallel (Jiang and Gruenwald, 2006a).

This chapter addresses this problem by employing a *divide-and-conquer* approach: big problems are split into smaller parts that can be computed concurrently and partial results are merged at the end of the computation. This technique has been proven very effective to process large volumes of data when it is associated with a distributed environment, such as *Hadoop*[1].

A distributed environment brings extra challenges to the computation of formal concepts, for example, it must guarantee that data shared across nodes are consistent. Another issue is that the number of signals transmitted among nodes should be the minimum possible to prevent network overflow and latency, thus *caching* and *pipe-lining* of operations are desirable.

Our distributed approach is not limited to the concept miner algorithm. We propose an architecture covering the entire data stream mining overflow, from data filtering and transformation, to closed itemset processing, and real-time visualisation.

The remainder of the chapter is organised as follows. A review of the particular issues of data stream mining is provided in Section 3.2. Section 3.3 formalises the terminology and presents a review of the state of the art on frequent itemset mining algorithms. Our approach for distributed computation of concepts is presented in Section 3.4 along with conclusions in Section 3.5.

3.2 General Issues in Mining Data Streams

The primary goal of data stream mining is to compute all subsets of items, hereafter called "transactions", which occur in at least a fraction of the stream. A data stream is a unbounded sequence of tuples continuously generated at a rapid rate. In contrast with batch algorithms, stream mining algorithms should satisfy the requirements listed below.

Each transaction should take at most once to look up a data stream, that means that stream mining algorithms should have linear time $O(N)$ where N is the size of the stream. Anything above this constraint may fall into a ever increasing, unmanageable computational cost (Jiang and Gruenwald, 2006a).

[1]Apache Hadoop. http://hadoop.apache.org

Another issue is to maintain memory usage within a limited range, although new data elements are continuously arriving from a data stream. This indicates that, at some point, some data must be discarded. As we will see later, this is a non-trivial task and may affect either the correctness or completeness (or both) of the stream mining algorithm.

The stream mining algorithm should use the result of previous computation and make the changes incrementally at minimum cost. This is usually done using a synopsis data structure, typically a prefix-tree, to store and maintain itemsets and frequency count in a compact way.

Finally, a stream mining algorithm runs indefinitely when executed. However, the real-time analysis of a data stream requires that results be promptly available when requested, therefore partial results should be provided.

3.2.1 Data Processing Model

In most cases, it is not practical to store and compute the entire data from the stream. Instead, most algorithms select portions of the stream that should be processed to mine itemsets, while insignificant data is discarded. Not all itemsets have equal importance in the mining of new itemsets. For example, an itemset that has much less support than a predefined minimum support is not necessarily monitored since it cannot be a frequent itemset in the near future, nor can it generate closed itemsets. Similarly, old itemsets may not me as relevant as new itemsets because users are, in general, interested in the *recent* itemsets. There are three main processing model to cope with this problem: the *landmark* model, the *damped* model and *sliding window* model.

3.2.1.1 Landmark Model

The **Landmark** model mines all frequent itemsets over the entire history of stream data from a specific time point called landmark to the present (Figure 3.1). This is the case of most batch itemset mining algorithms, such as FP-Growth (Han et al., 2000a) and CHARM (Zaki and Hsiao, 1999) when used to mine streams. Notice that, because it is an accumulative model, the required amount of memory grows with the number of transactions.

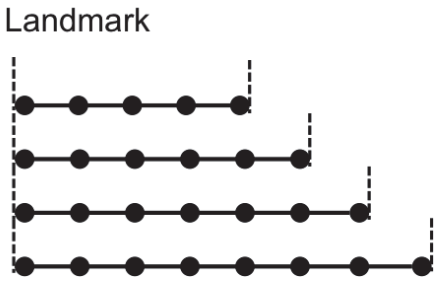


FIGURE 3.1: An illustration of the Landmark model (circle indicates a transaction). As a new transaction arrives it is added to the processing.

3.2.1.2 Damped Model

The **Damped** model mines frequent itemsets in stream data where each transaction has a weight, which decreases over time causing recent transactions to be more important than previous ones (Figure 3.2). A decay function is typically employed to assign weights to transactions. For example, in (Gupta et al., 2010), the decay function affects the support count of the transaction, with older transactions contributing less.

In contrast with the Landmark model, the Damped model may have newer transactions out of the window if its support, combined with the decay function, does not reach the minimum threshold (e.g. Figure 3.2 time span 3). This is a remarkable feature of this model *vis-à-vis* the Sliding Window model.

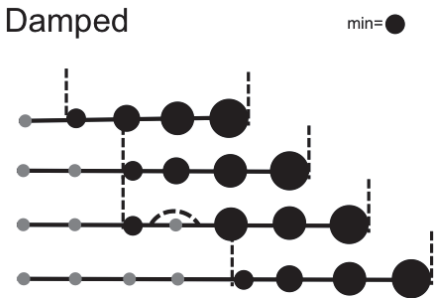


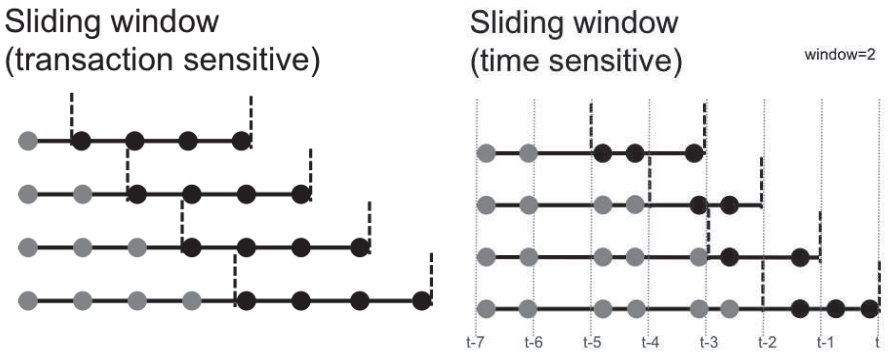
FIGURE 3.2: An illustration of a Damped model for stream processing. Circle indicates a transaction, diameter indicates its weight. Grey nodes represent the transactions that are out of the window.

3.2.1.3 Sliding Window Model

The **Sliding Window** model operates on a dynamically selected set of transactions, called *transaction window*. New transactions are added to the window while others are

removed. Sliding window models are further classified into two categories: *transaction-sensitive* and *time-sensitive* (Figures 3.3(a) and 3.3(b) respectively). The transaction-sensitive type is bounded by a fix number of transactions, in a FIFO data structure, while the time-sensitive window is dictated by fixed units of time, which may lead to a varying number of transactions in the window.

A strong feature in favour of this model is that *it guarantees that data can fit into the memory*, because the number of transactions to manage is constrained by the size of the sliding window. Nevertheless, all transactions in the window need to be maintained in order to remove their effects on the current mining results when they are out of range of the sliding window. This model is used by many algorithms (Chi et al., 2006, Chang and Lee, 2003, Li et al., 2009) including the algorithm proposed in this thesis.



(a) Sliding Window model (transaction-sensitive)

(b) Sliding Window model (time-sensitive)

FIGURE 3.3: An illustration of the Sliding Window model.

3.2.2 Data Structure

Having an appropriate data structure is fundamental for handling large amounts of data streams coming continuously. An inefficient data structure will largely affect the mining process, ultimately leading to a bottleneck in the system, if the stream rate is greater than what the data structure is able to handle in linear time.

One of the tasks of the data structure is to keep an overall state of the mining process in order to use the results of previous computation to make changes incrementally. Besides managing itemsets, the data structure should keep the state like frequency count, itemset statuses, transactions in the current window, etc. Frequent Itemset mining algorithms

typically employ some variant of a *Prefix-tree* (also known as *Trie*) to store transactions and itemsets. A prefix-tree can handle insertions and searching operations in linear time in worst case ($O(M)$ where M is the size of the transaction).

For example, in (Chang and Lee, 2003), the different combinations of items that appear in each transaction are maintained in a prefix-tree lattice structure. A node in the lattice contains an item and it denotes an itemset composed of items that are in the nodes of its path from the root. In (Manku and Motwani, 2002a) a lattice data structure is used to store itemsets, approximate frequencies of itemsets, and maximum possible errors in the approximate frequencies. In some cases more than one data structure is necessary. For example, Chi et. al. (Chi et al., 2006) propose a variant of a prefix tree called Closed Enumeration Tree (CET) to store closed frequent itemsets, while another prefix-tree, FP-Tree is used to store all transactions. Additionally, a hashset is used to store information about closed itemsets for checking in constant time ($O(1)$). We will describe some of these data structures in detail in the following sections.

3.3 Frequent Itemsets Mining Over Stream

3.3.1 Preliminaries

Before proceeding, let us formalise our terminology to avoid ambiguity. Let M $= \{i_1, i_2, \ldots, i_n\}$ be a set of n elements, called *items*. A subset $Y \subseteq M$ is called an *itemset*. Each transaction I is a set of items in M. Given a set of transactions T, the support of an itemset I is the *number of transactions* that contain I.

We assume that there is a lexicographical order among the items in M and we use $X \prec Y$ to denote that itemset I is lexicographically smaller than item J. For the sake of clarity, we may represent an itemset e.g. $\{A, B, C\}$ as ABC, given $A \prec B \prec C$.

Formal concepts are mathematically defined as *closed itemsets* for their intents. Although both terms differ in their philosophy[2], we may use both terms interchangeably. Recalling the definition of formal concept in Section 2.2, let T and Y be subsets of all the transactions and items appearing in a data stream D, respectively. A *closed itemset* can be defined by two operators, $\downarrow$ and $\uparrow$ where $T^{\uparrow} = \{j \in M | \forall t \in T, j \in t\}$ and

[2]FCA has a particular emphasis on the dual relationship of objects and attributes.

$Y^{\downarrow} = \{t \in D | \forall i \in Y, i \in t\}$. An itemset X is said to be closed if and only if $\uparrow$: $2^T \longrightarrow 2^Y$ and $\downarrow$: $2^Y \longrightarrow 2^T$ where the composite function $\uparrow\downarrow$ (") is called a closure operator.

3.3.2 Overview on Algorithms for Frequent Itemsets Mining

Frequent Itemsets Mining (FIM) over streams are not fundamentally different from those processing relational data. Indeed, most stream mining algorithms are variants of non-stream ones. Nevertheless, as explained in Section 3.2, the applicability of the former type of algorithms may be compromised for stream processing. For example, FIM algorithms that require several scans over the data may fall into exponential computational complexity as opposed to incremental ones, which build a intermediary structure and carry only the needed changes as new data arrives. The frequent itemsets mining algorithms can be either *deterministic* or *approximative*, i.e., itemsets are discovered if they score higher than some probability threshold. We are interested in the *deterministic* class of algorithms for this study. This section provides an overview of the most relevant algorithms, starting from frequent itemset mining algorithms for static data, closed itemsets mining, stream algorithms and lastly, closed itemsets mining algorithms for stream data. A timeline with the algorithms is provided in Appendix A.

3.3.2.1 Mining Frequent Itemsets with Candidate Generation: The Apriori algorithm

One of the most popular frequent itemset mining algorithm is the **Apriori** algorithm (Agrawal and Srikant, 1994). It uses a twofold approach to generate candidate itemsets and test if they are frequent. As intuition suggests, generating all possible candidate itemsets is a costly operation. The testing phase compare candidate's support with a given threshold.

Although the Apriori algorithm suffers from considerable overhead by generating potentially useless candidates, it is widely regarded as a pioneer algorithm for itemset mining and its simplicity serves as a good pedagogic example to contrast with other approaches.

3.3.2.2 Mining Frequent Itemsets without Candidate Generation

Another branch of frequent itemsets mining algorithms correspond to those without candidate itemset generation. The widely known algorithm **FP-Growth** (Han et al., 2000a) is also a two steps approach: First it builds a compact data structure called the *FP-Tree*, which is in fact a *prefix tree*. Then it extracts frequent itemsets directly from the FP-tree. An example of a FP-Tree is as follows. Nodes in the FP-Tree correspond to items and have a counter. FP-Growth reads 1 transaction at a time and maps it to a path in the tree (Figure 3.4).

A lexicographical ordering is applied to the itemsets, so paths can overlap when transactions share items (when they have the same prefix), in this case, counters are incremented. Pointers are maintained between nodes containing the same item, creating singly linked lists (dotted lines). The more paths that overlap, the higher the compression in this structure. This allows FP-tree to fit in memory, given a number of itemsets. It is worth noting that in the worst case scenario each itemset would produce a path in the tree. If this happens, the storage space is no better than the plain storage of itemsets cause a FP-Tree requires additional space for pointers and counters[3].

The second step consists in mining frequent itemsets extracted from the FP-Tree by traversing it in bottom-up fashion.

Mining frequent patterns can be viewed as first mining 1-itemset and progressively growing each 1-itemset by mining on its "conditional pattern" base recursively. A "conditional pattern" are the items which sufix is dependent of. For example, in Figure 3.4(d) one conditional pattern for {e} is {c,d,a:1} and the other is {a,d}. Then, for each conditional pattern, frequent itemsets are derived as permutations of the items, e.g. for {e} and {c,d,a} we have: {e},{e,a},{e,c},{e,d},{e,a,c},{e,a,d},{e,c,d} and {e,a,c,d}.

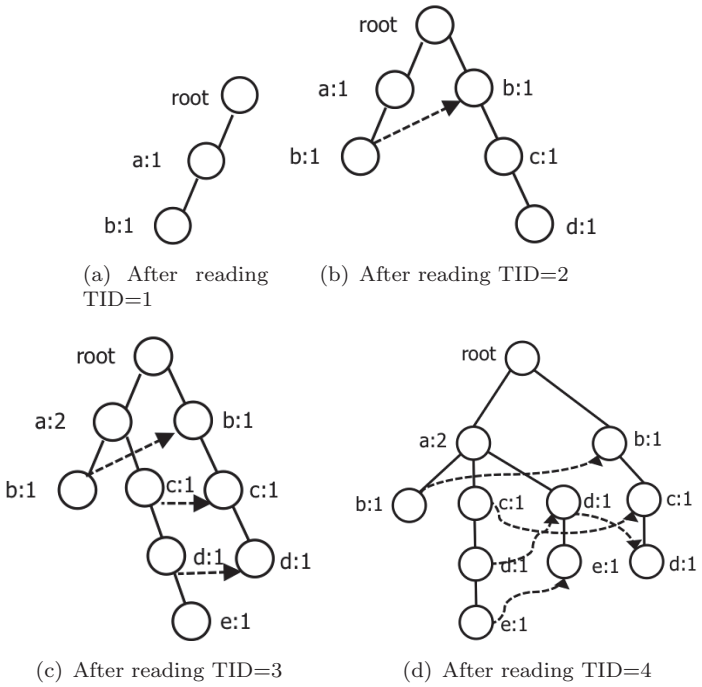
An item header table is used to record the occurrences of the same item across the tree (e.g. Figure 3.4(c)).

FP-Groth is not adapted for stream mining, because it scans data multiple times. For mining frequent itemsets over stream, Chang and Lee (2003) proposed a damped window based algorithm, called **estDec** for finding recent frequent itemsets adaptively over an

[3]Heuristically the worst case scenario is highly unlikely for most applications. In text mining, however, computation may lead to worst case scenario.

TABLE 3.1: Transactions table.

TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}



(a) After reading TID=1 (b) After reading TID=2

(c) After reading TID=3 (d) After reading TID=4

FIGURE 3.4: An example of a FP-Tree after reading each transaction.

online data stream. The effect of old transactions on the mining result of the data steam is diminished by decaying the old occurrences of each itemset as time goes by. Lower-weighted itemsets may be pruned from the prefix tree (called *monitoring lattice*).

A single pass algorithm was proposed to count frequency of data elements over a data stream in (Manku and Motwani, 2002b). In the **Lossy Counting** algorithm, the set of frequent itemsets in a data stream is found when a maximum allowable error rate and a minimum support is given. A set of newly generated transactions in a data stream is loaded together into a fixed-sized buffer in main memory and they are batch-processed[4]. The information about the previous mining result up to the latest batch operation is maintained in a linear data structure containing the information about the itemset, count and error.

[4]Also known as *offline* stream processing

3.3.2.3 Mining Closed Frequent Itemsets

Whilst mining all itemsets is desirable, it is not necessary to compute them all. It is sufficient to compute only *closed itemsets* which is a set much smaller than all and all other rules can be derived from them (Zaki and Hsiao, 1999). Several algorithms have been proposed to mine closed itemsets over static datasets: **Closet** (Pei et al., 2000), **CHARM** (Zaki and Hsiao, 1999), **Closet+** (Wang et al., 2003), **CHARM-L** (Zaki and Hsiao, 2005), **FP-Close** (Grahne and Zhu, 2003), **DCI-Closed** (Lucchese, 2004) (see (Duneja and Sachan, 2012) for a survey).

CHARM (Zaki and Hsiao, 1999) is among the first algorithms to mine closed frequent itemsets and the technique influenced a number of other modern algorithms. CHARM works by pruning candidates based not only on subset infrequency (i.e., no extensions of an infrequent itemset are tested), but also branches based on non-closure property, which means that any non-closed itemset is pruned. The algorithm uses no internal data structures like hash or prefix tree, and the procedure goes into recursion only if a node is a good candidate. This feature makes this algorithm unsuitable for stream data.

Among the algorithms for mining closed itemsets over a data stream, the **Moment** algorithm (Chi et al., 2006) is arguably the most referenced. This algorithm maintains a dynamically selected set of itemsets which can be classified as: infrequent gateway nodes, unpromising gateway nodes, intermediate nodes, and closed nodes. A summary data structure, called *Closed Enumeration Tree* (CET), maintain those itemsets over a transaction-sensitive sliding window. The selected itemsets consist of closed frequent itemsets and a boundary between closed frequent itemsets and the rest of the itemsets. When a new transaction arrives, it checks the closed frequent itemsets stored in a hash table with its support and tidsum information to decide its node type according to the node properties and incrementally updates the associated nodes information.

An improvement over the original algorithm is presented in (Li et al., 2009), called **New-Moment**. The NewMoment algorithm uses a bit-sequence representation for itemsets in order to meet the requirement of limited memory space of the maintenance of closed frequent itemsets generated. Experiments show that NewMoment outperforms the original Moment algorithm and has smaller memory footprint.

Jiang and Gruenwald (2006b) propose a new algorithm for mining closed itemsets in a sliding window, called **CFI-Stream**. When performing addition and deletion operations, the CFI-Stream algorithm checks each itemset in the transaction *on the fly* and updates the associated closed itemsets supports. Current closed itemsets are maintained and updated in real time in a prefix-tree called DIrect Update (DIU) tree. In contrast with Moment, CFI-Stream maintains only closed itemsets in the structure and checks closure as a new transaction arrives. Author's benchmark shows that CFI-Stream and Moment algorithm have comparable running times, but CFI-Stream requires less storage space.

In (Chen and Li, 2007) the authors propose an algorithm that exploits the lexicographic order relation combined with the closure climbing technique which obtains closure generators for itemsets. This algorithm uses an in memory data structure called **GC-Tree** (Generator and frequent Closed itemsets Tree) to store all the frequent closed itemsets in the current sliding window.

Gupta et al. (2010) proposed an algorithm, **CLICI** (Concept Lattice based Incremental Closed Itemset), for mining all recent closed itemsets in landmark window model. The algorithm uses a decay function to assign higher weights to recent itemsets. The synopsis data structure, called CILattice, stores all recent closed itemsets in the stream. Periodically, the lattice is traversed and all nodes having a support count less than threshold are are removed from the lattice.

Our contribution. In this chapter, we propose a novel distributed architecture for mining formal concepts over data streams. It computes and maintains closed itemsets online and incrementally, and can output the current closed frequent itemsets in real time based on users' query. Our approach is comprised of several components that carry the computation of concepts from a basic transaction, filter and transforms data as well as provides analytic features to visually explore data.

The core of our approach is the distributed concept mining algorithm. The algorithm is a variant of the Moment algorithm introduced in (Chi et al., 2006). We chose Moment because it has fairly decoupled procedures and, with some modifications, the CET operations may run in parallel. Other approaches seemed to require much more effort or were impractical. Nevertheless, when implementing and analysing the original Moment algorithm, we noticed the following issues:

- When deleting an transaction from the CET, a *closed* item set may become intermediate, however, there is no guarantee that the algorithm will carry out this change because only nodes added in the list F will be visited, and for a itemset be added in F it needs to become non-frequent (see *Explore* in Chi et al. (2006)). The algorithm ignores the fact that a *closed* frequent itemset may become *intermediate*.
- The *leftCheck( )* procedure might cause the pruning of a node that contains a *closed* itemset in its descendants (see *Deletion* in Chi et al. (2006)).

We tackle these issues in our approach and we introduce an extra verification step to reduce the amount of useless subsets generated (Section 3.4.3.2). To the best of our knowledge, this is the first *distributed* approach to compute and visualise formal concepts over a data stream.

3.4 A Distributed Approach to Compute Formal Concepts over Data Streams

In this section, we describe our distributed approach for mining concepts in real-time over a data stream. For this task, we used **Storm**[5], a distributed real-time data processing platform. Strom provides abstractions to carry on distributed computation similarly to "Map Reduce jobs" in Hadoop, however, when jobs in Hadoop eventually finish, in Storm a topology runs until the process is interrupted by the user. Storm is able to process over a million tuples per second per node. It is scalable, fault-tolerant, and uses persistent queuing to guarantee that the data will be processed. A brief introduction to Storm is given in the next section.

We may refer to "*node*" as an object in one of our data structures. "*Node*" is a term often used in distributed computation to refer to independent computation units. In this case we will refer to them hereafter as "*computation unit*".

3.4.1 Storm Topologies, Spouts and Bolts

There are three abstractions in Storm: spouts, bolts, and topologies. A **spout** is a source of streams in a computation. A **bolt** processes any number of input streams

[5]http://github.com/nathanmarz/storm

and produces any number of new output streams. Most of the logic of a computation is into bolts, such as functions, filters, streaming joins, database calls, etc. *Each bolt may have one or several instances running in parallel.* A **topology** is a network of spouts and bolts, with each edge in the network representing a bolt subscribing to the output stream of some other spout or bolt. In Storm, topologies run indefinitely when deployed. Figure 3.5 illustrates a Storm topology for the distributed computation of concepts. Each bolt is introduced below and explained in detail in the following sections. Each bolt runs sequentially with regard to a given transaction, but many transactions are processed in parallel in this schema. A memory dataset is employed to keep the data structures synchronized across the bolts. This adds an extra challenge to manage the state synchronized and to manage "race conditions"[6]. The orange colour indicate which bolts access the memory dataset.

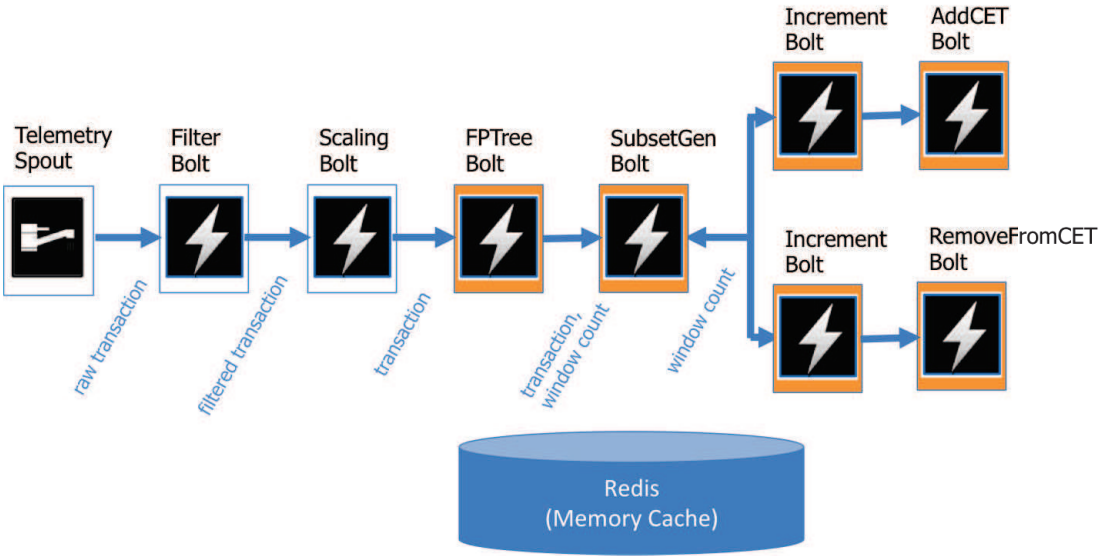


FIGURE 3.5: An illustration of a Storm topology with bolts and spouts.

TelemetrySpout. It is a stream generator for the telemetry use case we discuss in Chapter 5. This unit produces tuples from sensor data to the rest of the topology.

FilterBolt. It emits a transaction only if a condition is satisfied for the incoming transaction.

ScalingBolt. It processes a transaction and replaces continuous-valued attributes to multi-categorical values.

FPTreeBolt. It adds the transaction to the FP-Tree and emits the input transaction.

[6]When multiple nodes request the same resource.

SubsetGenBolt. It generates and stores the subsets of items of a given transaction by traversing the Closed Enumeration Tree (CET) described in Section 3.4.3.4. It emits the transaction.

IncrementBolt. It increments the support count and *tidsum* of the nodes in the FPTree and emits the current transaction to AddToCETBolt.

AddToCETBolt. Responsible for appending the transaction to CET and carrying the necessary changes. It may emit a tuple to SubsetGenBolt when necessary.

DecrementBolt. It decrements the support count and *tidsum* from the oldest transaction in the sliding window. It emits the *oldest* transaction in the sliding window to the RemoveFromCETBolt.

RemoveFromCETBolt. It deletes the oldest transaction in the sliding window from the CET tree. Notice that the deletion bolts run concurrently with the addition bolts.

In the following section we describe how the above Storm topology fits in the overall architecture we propose in this thesis.

3.4.2 System Architecture

The architecture of the real-time system is a client-server comprising mainly of *three* components: the concept mining algorithm that runs on top of a Storm topology; the user interface/analytics component, called **Cubix**; and a Node.js server that pushes updates to the client browsers in real-time (RT) (Figure 3.6).

The concept mining algorithm works *offline* in that no user interaction is required. It works independently of the other modules and outputs all formal concepts in the current window.

The user interaction is done in the **Cubix** analytics. It provides visualisations, filtering and searching capabilities, and a number of analytics tools (discussed in Section 4).

When the user makes a request, Cubix passes the request to the **Node.js server** which collects the current closed concepts from the CET and pushes updates to client browsers in real-time via *websockets*.

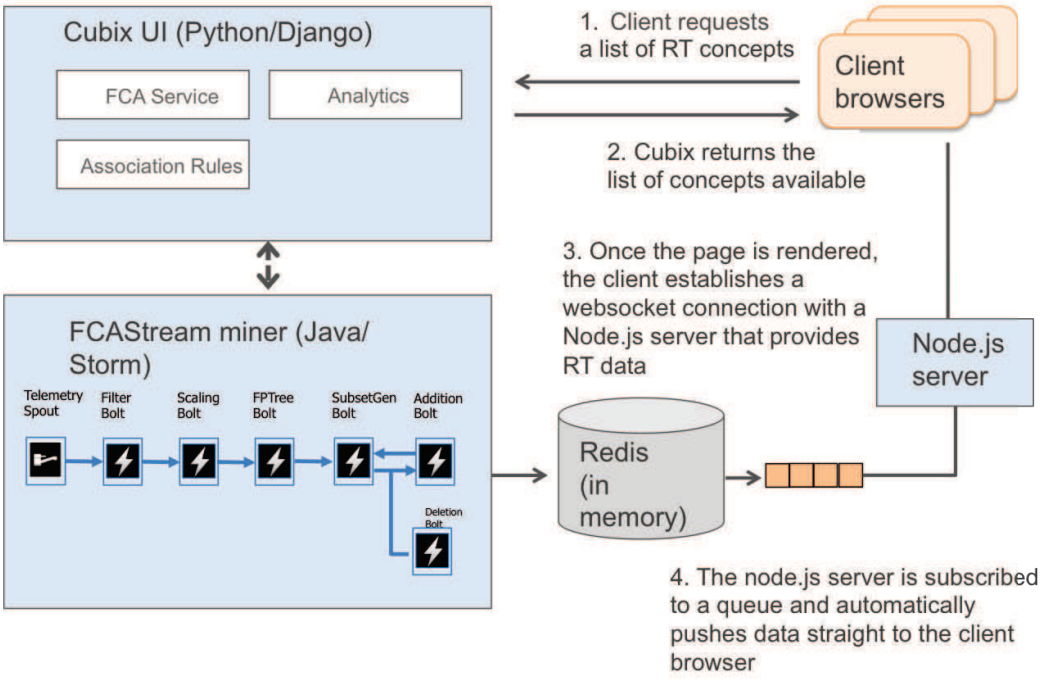


FIGURE 3.6: System architecture showing system's components (in blue) and the requests between them.

3.4.3 Storing Itemsets in the Window

Having an efficient way to store and retrieve itemsets and transactions is a key factor in FIM algorithms. In a distributed environment it adds an extra challenge: the structure is shared across multiple computing units, thus it is necessary to manage a consistent state and racing conditions. In the next sections, we describe the data structures we are using for storing transactions and itemsets.

3.4.3.1 CET Node Properties

We use a synopsis data structure called Closed Enumeration Tree - CET, described in Section 3.4.3.4, to monitor only the frequent itemsets that *may* become closed itemset or has a closed itemset in their descendants. We then prune all descendants of "unpromising" and "infrequent" itemsets. This approach was first proposed in (Chi et al., 2006) and it defines node properties to make assumptions about potential closed itemsets.

We introduce a new node property, "*Idle*" to indicate itemsets that are to be deleted from the CET *if* not updated after a number of transactions δ. The rationale for this property is as follows. After analysing common bottlenecks in the performance

of frequent itemset mining algorithms using the sliding window model, we discovered that the re-computation of previously deleted itemsets can be amortized by keeping the itemset in the synopsis. This way, if a node has become infrequent and then frequent in a interval $< \delta$ is will not need to be generated again. This situation occurs notably when the minimum support has a low value (which is often the case in anomaly detection systems we explain in Section 5.5).

The CET node properties are described as follows.

Infrequent gateway nodes. *All descedants of a infrequent gateway node are also infrequent.*

This definition is derived from the *apriori* property (Agrawal and Srikant, 1994). If an item set occurs N times, all its subsets occurs at least N times, in other words, if n_I is an infrequent gateway node, then any node n_J where $J \supset I$ represents an infrequent itemset. All descendants of infrequent gateway nodes are pruned from the CET.

Unpromising gateway nodes. *An unpromising gateway node has no closed itemset in its descendants.*

A node n_I is an unpromising gateway node if there exists a closed frequent itemset J such that $J \prec I$, $J \supset I$, and J occurs in the same transactions of a frequent itemset I. Descendants of an unpromising gateway node are pruned because no closed nodes can be found there.

Idle nodes. *An itemset marked as "idle" is to be removed from CET if it is not updated after δ transactions.* A node n_I is idle if it one of its antecedents is either "infrequent" or "unpromising". If a idle node is not updated after δ transactions it is then permanently deleted from the CET.

Intermediate nodes. *An intermediate node has at least one child with the same support.*

If n_I is a intermediate node, it has a child node n_J such that J has the same support as I does and n_I is not an unpromising gateway node. Intermediate nodes must not be pruned from the CET tree because it has at least one closed node among its descendants.

Closed nodes. *An itemset is closed if none of its immediate supersets has the same support.*

A node n_I is closed if there is no parent node n_J such that J has the same support as I. These nodes represent closed frequent itemsets in the current sliding window (see definition in Section 2.2).

3.4.3.2 An Extra Checking to Reduce the Number of Itemsets Generated

The original Moment algorithm judges closed itemsets not via closure checking as it is traditionally done, but through checking and updating the nodes properties as described in the previous section. A drawback of this approach is that it needs to store extra information besides the closed itemsets themselves. The CFI-Stream algorithm for instance, uses closure check on the fly through a series of conditions, and only the closed frequent itemsets need to be stored (Jiang and Gruenwald, 2006b). Nevertheless, benchmarks showed no significant performance difference between the two approaches (Jiang and Gruenwald, 2006b).

The bottleneck of Moment's algorithm is that the *explore* routine needs to generate subsets and check node properties. As we mention in Section 2.4.4, we can incorporate conditions to the algorithm in order to reduce the number of unnecessary comparisons. The extra checking we propose takes advantage of the fact that if an itemset I and a newly generated J such that $J \subset I$ have the same support, I cannot be closed because J is a smaller set containing all items of I, therefore I does not need to be generated. In the example of figure 3.7, the itemset AC do not need to be generated because $ABC \subset AC$ and $supp(ABC) = supp(AC)$.

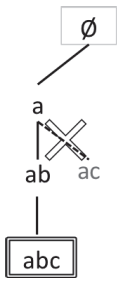


FIGURE 3.7: Itemset generation phase. AC is not generated because of ABC.

This checking is done in the SubsetGenBolt we describe in Section 3.4.6.

3.4.3.3 Storing Transactions in the FP-Tree

Each transaction in the stream is identified by an unique ID (we refer to it as TID), which is given by the current time in milliseconds in our case. The support of an itemset is the sum of the TIDs comprising all transactions in which the itemset takes part. We call it *tidsum*.

We need to store all transactions in order to calculate the *support* and the *tidsum* for transactions in the window. This can be done efficiently with a prefix tree like the FP-Tree (Han et al., 2000b), as introduced in Section 3.3.2.

In our implementation, we store the FP-Tree in a key-value memory cache where each node contains information about its items, support, children and the next item in the linked list. Table 3.2 illustrates the storage for the FP-Tree in Figure (3.8).

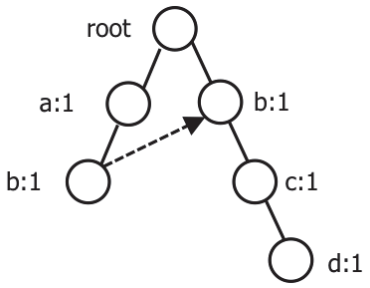


FIGURE 3.8: An example of a FP-Tree with itemsets {a,b} and {b,c,d}.

TABLE 3.2: Storage of a FP-Tree in a key-value database.

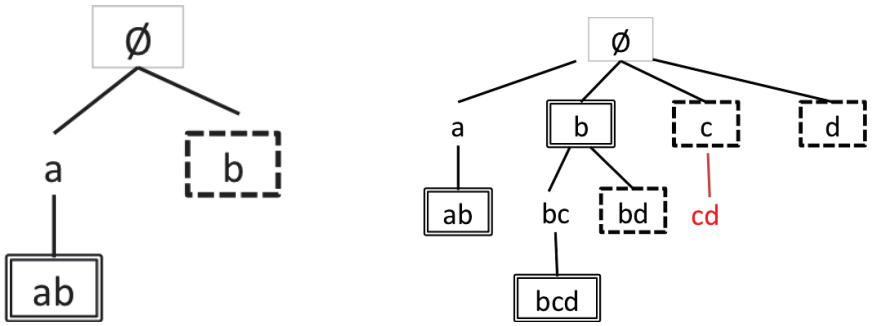
Key	**Value**
fpnode_1	items: [ ], supp: 1, children: [“fpnode_2”, “fpnode_4”]
fpnode_2	items: “a”, supp: 1, children: [“fpnode_3”], next: *null*
fpnode_3	items: “b”, supp: 1, children: [ ], next: “fpnode_4’
fpnode_4	items: “b”, supp: 1, children: [“fpnode_5”], next: *null*
fpnode_5	items: “c”, supp: 1, children: [“fpnode_6”], next: *null*
fpnode_6	items: “d”, supp: 1, children: [ ], next: *null*

To compute support and tidsum in this structure, we iterate through each item in the header table, navigating through the concerned nodes in the tree using the linked list (dotted arrows). The insertion and search in the tree have $O(logN)$ time complexity in average and $O(N)$ in the worst case (i.e. when every itemset form a separate branch in the tree).

3.4.3.4 Marking Boundaries with a Closed Enumeration Tree (CET)

A prefix-tree is used in several stream mining algorithms to store transactions and itemsets in a compact way (Chi et al., 2006, Jiang and Gruenwald, 2006a, Chang and Lee, 2003, Li et al., 2009, Grahne and Zhu, 2003, Lucchese, 2004). The Closed Enumeration Tree (CET) was introduced in the Moment algorithm (Chi et al., 2006) to store a dynamically selected group of itemsets (Figure 3.9). The itemsets in CET tree are used to determine "boundaries" to represent closed itemsets, itemsets that may become closed in the future, and itemsets that cannot generate closed itemsets.

As explained in Section 3.4.3.1, some heuristics are applied at each insertion to the CET to unecessary storage space and computation. In the example of Figure 3.9(b), node {c,d} is not generated because {c} is a unpromising gateway node.



(a) After reading {a, b}

(b) After reading {b, c, d}

FIGURE 3.9: An example of a Closed Enumeration Tree (CET). Dashed nodes: Unpromising gateway nodes; Borderless: Intermediate nodes; Solid rectangles: Closed Itemsets.

Like the FP-Tree previously mentioned, the CET is stored in a key-value memory database in a similar fashion, with one caveat: children ID's are stored in a lexicographically sorted collection[7]. This is because we are often exploring those itemsets using the $\prec$ and $\succ$ relations.

In addition to the CET, we use an additional hash table to enable quick lookup about which transactions a given closed itemset occurs. The key of the hash is comprised of both *tidsum* and *support*, and it stores a pointer to the corresponding closed node in the CET. We will refer to this hash table as **CET hash table**.

[7]Redis provides a built-in sorted collection, called ZSET, which insertion time is proportional to the logarithm of the number of elements.

3.4.4 Filtering and Scaling

Two major pre-processing tasks in FCA is to filter irrelevant attributes and objects, and distribute multi-valued attributes into ranges so that each value of a multi-valued attribute becomes nominally a one-valued attribute (see Discretisation and Booleanisation in Section 2.2).

This algorithm in the **FilterBolt** transmits a transaction only if pre-defined filter condition is satisfied. For example, some types of analysis may be focused on transactions containing particular attributes-values. The filtering conditions are pre-defined by a domain expert.

The **ScalingBolt** transforms multi-valued attributes (Algorithm 4). It takes as input the transaction I, a intervals mapping $\mathcal{M}$ and a dictionary $\mathcal{D}$. The intervals mapping is a dictionary containing the partitions for each attribute[8]. This is only possible if the maximum and minimum values of a given attribute is known *apriori*. For example, if a sensor is sending information about temperature in Celsius, the dictionary would contain intervals such as "[<-100]", "[-100:-50]", ..., "[>100]". For each multi-valued attribute a corresponding interval in $\mathcal{M}$ (lines 4 and 5).

One may argue in favour of a "floating" intervals approach, where the system automatically redistributes ranges according to the upcoming values of attributes. However, in the sliding window approach, keeping a floating interval would require the update of maximum and minimum values for each multi-valued attribute at each addition/deletion of a transaction from the window. This could be done efficiently with a *double-ended priority queue*, where the consultation time complexity is O(1) and insertion O(log N). Most importantly, it would require the re-computation of all itemsets with the new attribute scales.

In order to reduce the amount of memory used to store the string representation of attributes, the algorithm uses a second dictionary $\mathcal{D}$ to replace the name of each attribute by a more compact representation, typically a index number (line 9). A bit array representation for itemsets may be used, however, large bit arrays tend to have long streams of zeroes or ones. This phenomenon wastes storage and processing time. Each pair attribute-value is represented by a bit in the bit array, this representation would

[8]The interval mapping is typically defined by a domain expert.

not be suitable for sparse data such a text, where the number of 0's would account for most storage space. Thus, instead of compressing bit arrays as streams of bits, we might assign them a index from the dictionary $\mathcal{D}$.

Algorithm 4: ScalingBolt

Data: A transaction T, intervals mapping $\mathcal{M}$ and a dictionary $\mathcal{D}$.

```
1  begin
2      /* Scaling of multi-valued attributes                    */
3      foreach multi-valued item t ∈ T do
4          find the interval [a, b] in M where a ≤ t < b
5          replace t by the interval [a, b]
6      end
7      /* Replace string representation                         */
8      foreach item t ∈ T do
9          replace t by the corresponding string in the dictionary D
10     end
11     send modified transaction T to FPTreeBolt
12 end
```

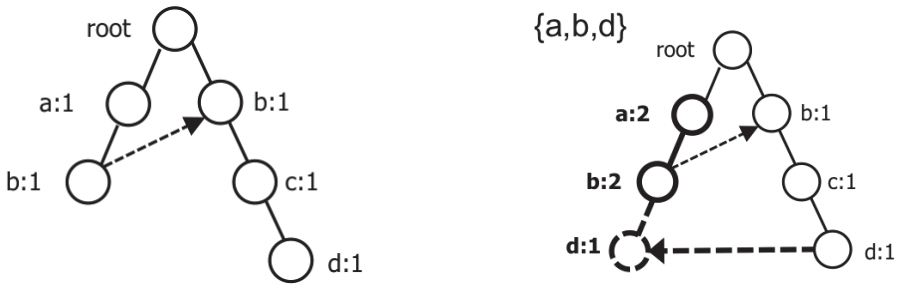
3.4.5 Update FPTree

After the transaction has been manipulated by the **ScalingBolt**, it is sent to **FPTreeBolt** which adds the itemset to the FP-Tree mentioned before. The algorithm iterates over each item in the transaction, finding its way through the paths of the FP-Tree in a top-down fashion. If the current item in the transaction exists in the FP-Free, it updates the tidsum and support for the node, otherwise, it appends a new node with the current item and proceeds to the next item in the transaction. Figure 3.10 illustrates the addition of a new transaction in the FP-Tree. Notice that the existing {d} node have its "*next*" pointer updated to the newly created node.

Experiments of Moment show that the boundary in CET is stable so the update cost is little.

3.4.6 Subset Generation

Once that **FPTreeBolt** added the transaction to the FP-Tree, we start the frequent itemset mining phase with the **SubsetGenBolt**. This module generates subsets of item-ids along with their frequencies in the current transaction in lexicographic order.



(a) Original FP-Tree (b) After reading {a, b, d}

FIGURE 3.10: An example of adding a new transaction {a,b,d} to the FP-Tree. Support and Tidsum are updated for existing nodes and a new node is appended.

Not all possible subsets need to be generated. As we have seen earlier, *unpromising and infrequent nodes* cannot generate closed itemsets and thus the subsets of these nodes do not need to be generated. Algorithm 5 shows the procedure.

First, it skips any unpromising or infrequent nodes (lines 2 and 3). Then, for each item in the current transaction, it goes recursively through the corresponding CET path incrementally appending new items (if they are not there already) until it completes the *powerset* of the transaction (lines 8-17).

The powerset generation has exponential time complexity $O(2^N)$, and this is the most costly operation in the entire system. Fortunately, this operation is not frequent since most itemsets should fit in some existing branch of the CET and most importantly, this is a non-blocking operation due to the parallelism the approach provides. For example, while the SubsetGenBolt is running the procedure, another thread (or machine) is consuming the incoming tuples. Alternatively, the system administrator can increase the number of threads for SubsetGenBolt without affecting the topology. Load balancing and fault-tolerance are managed by Storm.

3.4.7 Updating CET Tree

For each upcoming transaction we need to update the support in the FPTree and the parts of the CET related to the transaction. We also need push the transaction to the sliding window and pop the *oldest* one from it. These tasks are performed by the **IncrementBolt**, **AddToCETBolt**, **DecrementBolt** and **RemoveFromCETBolt**. In the following sections we explain each of them in detail.

Algorithm 5: SubsetGenBolt

```
Data: CET node n_I and a transaction T.
1  begin
2      if n_I is unpromising gateway or infrequent then
3          return
4      end
5      lastIdx ← index of an item t in T which is the last item of n_I
6      for j ← lastIdx to |T| do
7          curItem ← T[j]
8          if n_I has no children containing curItem then
9              newNode ← n_I ∪ curItem
10             /* Store newNode and add it as child of n_I                    */
11             Store(newNode)
12             AddChild(n_I, newNode)
13             /* Continue the recursion with the newly generated itemset     */
14             SubsetGenBolt(newNode, T)
15         else
16             /* n'_I is the child node of n_I containing curItem            */
17             SubsetGenBolt(n'_I, T)
18
19     end
20 end
```

3.4.7.1 Adding a Transaction to the Sliding Window

To add a transaction in the CET, the **IncrementBolt** updates the *tidsum* and *support* of the corresponding nodes (Algorithm 6 - line 3). If a node becomes frequent, its powerset needs to be generated by the **SubsetGenBolt** (line 5).

Algorithm 6: IncrementBolt

```
Data: Itemset n_I and transaction I.
1  begin
2      foreach node n'_I child of n_I do
3          update support and tidsum of n'_I
4          if n'_I is newly frequent then
5              emit to generateSubsetsBolt(n'_I)
6          end
7      end
8      AddToCETBolt(I)
9  end
```

The **AddToCETBolt** is a depth-first procedure that visits itemsets in lexicographical order and update node properties. It goes recursively through the leaves of the tree until the top. The function *hashCheck( )* checks if if there exists a previously discovered

closed itemset n_J that has the same support as n_I and $J \supset I$ (line 4). If so, mark it as *unpromising gateway* node (line 5)[9]. Otherwise, if a child node n'_I does not contain all items of n_I but has the same support, mark n_I as a intermediate node (lines 6 and 7). If none of the above conditions are satisfied, n_I is a closed node. If n_I is currently marked as an unpromising gateway node, we need to generate its powerset (line 11) and check if any of its children is a closed node by looking at the hash table (line 15). If so, mark n_I as unpromising gateway and return. The worst-case time complexity of the AddToCETBolt is $O(N)$ (without powerset generation).

Algorithm 7: AddToCETBolt

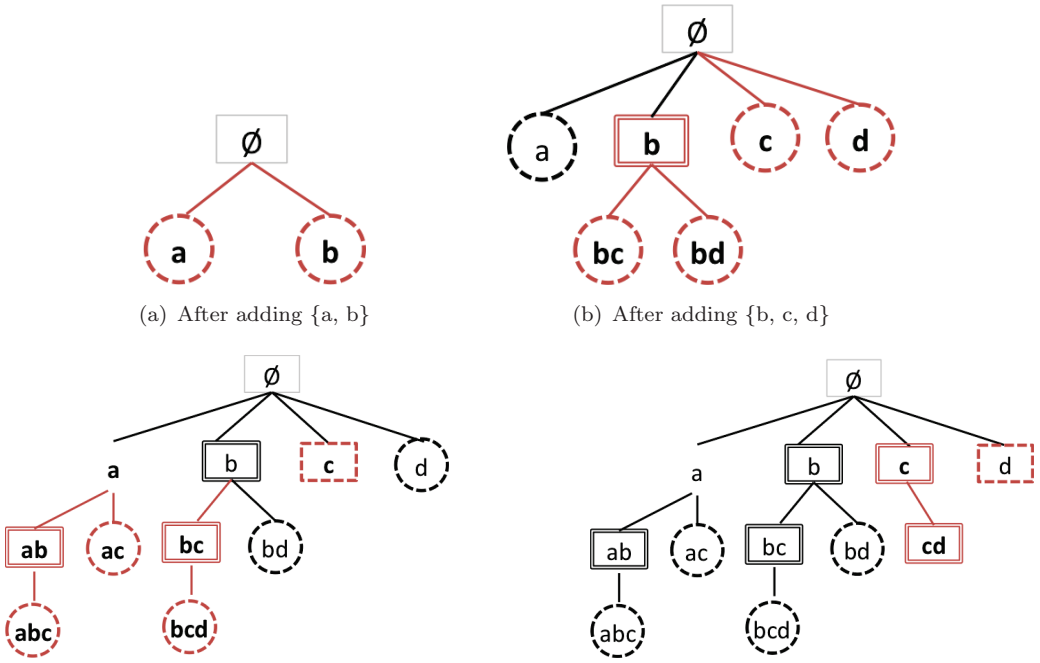
```
Data: Itemset n_I and transaction I.
1  begin
2      foreach node n'_I child of n_I do
3          AddToCETBolt(n'_I)
4          if hashCheck(n'_I) = true then
5              mark n'_I as unpromising gateway
6          else if Exists a child node n''_I with same support as n'_I then
7              mark n'_I as intermediate
8          else
9              if n'_I is unpromising gateway then
10                 /* This branch was prunned, needs to be generated again.   */
11                 emit to SubsetGenBolt(n'_I)
12             end
13             mark n'_I as closed
14             /* Check if a new child is closed                               */
15             if hashCheck(n'_I) = true then
16                 mark n'_I as unpromising gateway
17             end
18
19     end
20 end
```

Figure 3.11 illustrates the updates in the CET after reading each transaction.

3.4.7.2 Removing a Transaction from the Sliding Window

When deleting a transaction from the window, we decrement the support count and tidsum for each node (**RemoveFromCETBolt** 8 line 3). If the node has become infrequent its descendants must be pruned and the node marked as infrequent (lines 5 and 6).

[9] see **Unpromising gateway nodes** property in Section3.4.3.1



(a) After adding {a, b}

(b) After adding {b, c, d}

(c) After adding {a, b, c}

(d) After adding {c, d}

FIGURE 3.11: Changes on the CET when adding transactions. Consider minimum support = 2. Dashed circles: Infrequent gateway nodes; Dashed squares: Unpromising gateway nodes; Borderless: Intermediate nodes; Solid square: Closed nodes.

Algorithm 8: DecrementBolt

Data: Itemset n_I and transaction I.

```
1  begin
2      foreach node n'_I child of n_I do
3          update support and tidsum of n'_I
4          if n'_I is newly infrequent then
5              mark n'_I as infrequent gateway node
6              mark all descedants of n'_I as Idle
7          end
8      end
9      RemoveFromCETBolt(I)
10 end
```

In the **RemoveFromCETBolt**, for each "promising' 'child n'_I of n_I it checks if there is a node with the same support and containing all items of the current node using *hashCheck( )* (line 5). If so, n'_I is unpromising and has its descendants pruned from CET (lines 6 and 7). Otherwise, the procedure goes recursively until it reaches the leaves of the CET (line 10). Then, the procedure verifies if closed nodes have changed by looking at their immediate children. If the current node n'_I is closed and there is a child n''_I with the same support as n'_I, mark n'_I as intermediate and remove its entry in

the hashtable (lines 12-15). If there is no child with the same support as n'_I, it remains closed and its entry in the hashtable is updated (line 17).

Algorithm 9: RemoveFromCETBolt

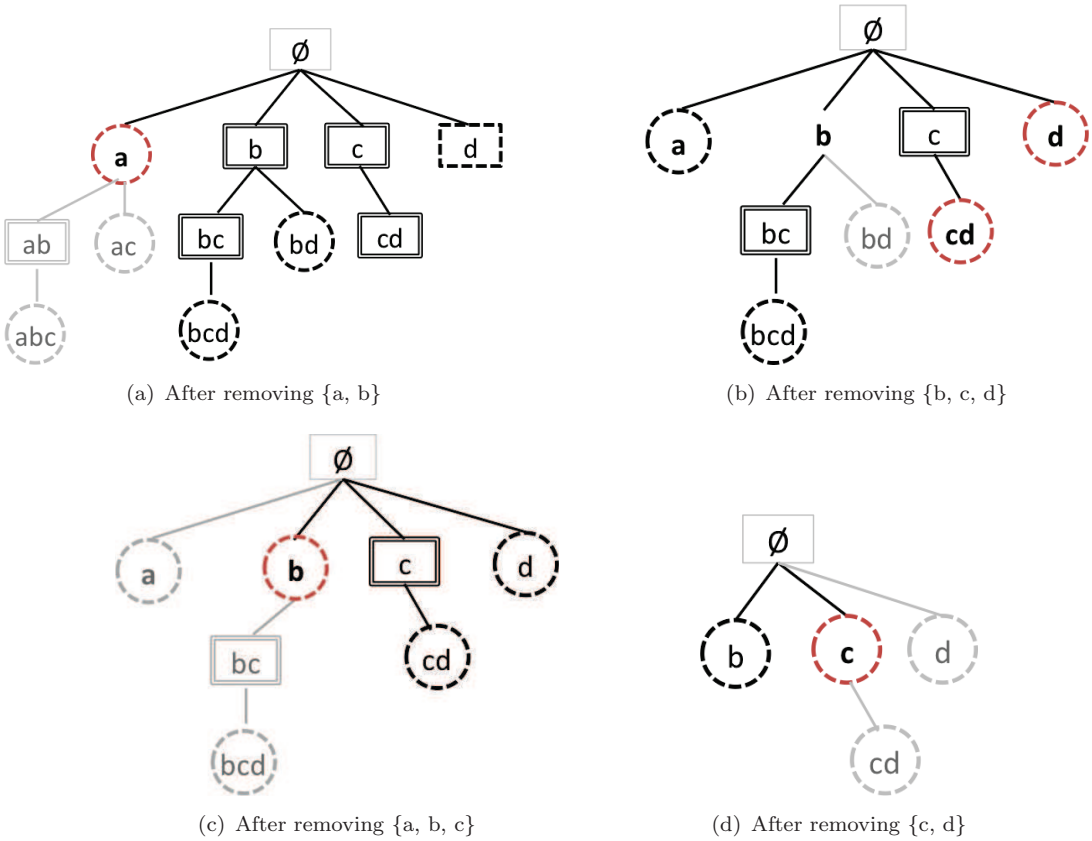
```
   Data: Itemset n_I and transaction I.
 1 begin
 2     foreach node n'_I child of n_I do
 3         if n'_I is an unpromising gateway or infrequent gateway node then
 4             continue
 5         else if hashCheck(n'_I) = true then
 6             mark n'_I as unpromising gateway
 7             mark all descedants of n'_I as Idle
 8         else
 9             foreach node n''_I child of n'_I do
10                 RemoveFromCETBolt(n''_I, I)
11             end
12             if n'_I is closed then
13                 if there is child node n''_I of n'_I with the same support as n'_I then
14                     mark n'_I as intermediate
15                     remove n'_I tidsum and support from hashtable
16                 else
17                     update n'_I tidsum and support on hashtable
18
19             end
20
21     end
22 end
```

Figure 3.12 illustrates the updates in the CET when deleting each transaction.

3.4.8 The Distributed Caching Problem

In a distributed environment, locally caching of data may become a problem when nodes request the data stored in the cache of another node. Consider the following scenario. A node "A" generates a new itemset I, stores it in its cache and proceeds generating the children of I. When the powerset of the itemset I is complete, the cache flushes data to the database. Another node "B" generating an itemset J needs to know if J or any of its children was already generated. However, J may have been generated by node "A" but it is stored in its cache.

In our experiments this situation occurred frequently. There should be a trade-off between the amount of data in the cache and the frequency in which it is flushed to the



(a) After removing {a, b}

(b) After removing {b, c, d}

(c) After removing {a, b, c}

(d) After removing {c, d}

FIGURE 3.12: Changes on the CET when deleting transactions. Minimum support = 2. Dashed circles: Infrequent gateway nodes; Dashed squares: Unpromising gateway nodes; Borderless: Intermediate nodes; Solid square: Closed nodes.

database. A workaround is to assign a *hash key* for each itemset processed so that a newly created itemset will be stored in the same storage address of the existing one, thus avoiding duplicates. That does not solve, however, the fact that multiple nodes might be computing the same itemset. As future work, we will investigate a strategy based on broadcasting across bolts in order to communicate which itemsets are being processed by which node.

3.5 Chapter Summary

This chapter described a novel distributed architecture to mine formal concepts over a data stream. A review on the main algorithms for itemset mining over stream was provided in Section 3.3.2. Using a synopsis data structure, typically a prefix-tree, stream

mining algorithms use the result of previous computation and make changes incrementally as new data arrives in the stream.

In most cases, it is not practical to store and compute the entire data from the stream. Instead, most algorithms select portions of the stream that should be processed to mine itemsets. The main processing models are: the *Landmark* model, the *Damped* model and *Sliding Window* model.

Our proposed distributed approach is described in Section 3.4. The distributed mining algorithm checks and maintains closed itemsets in an incrementally and in parallel. Information are shared across the bolts using a remote memory database. An extra checking is done in order to reduce the number of itemsets generated (Section 3.4.3.2).

The next chapter is dedicated to the Visual Analytics component of the architecture presented in Section 3.4.2.

Chapter 4

Visual Analytics for Formal Concept Analysis

4.1 Introduction

When FCA is applied to Business Intelligence, data is typically large, complex and dynamic - new questions are raised everyday by business experts and managers alike. The analysis is performed on a more frequent basis than traditional applications of FCA, therefore interactive features are needed. Because the nature of the task is not linear, which is the case of information retrieval systems, it is necessary to provide a number of tools. For most tasks, only a few of these tools are used, but they are all important in the sense that they can be combined to address different analysis needs. This is the "Photoshop" paradigm.

FCA provides semantic groupings of objects and attributes based on their co-occurrence. In real-world datasets the resulting concept lattices are often too large to allow users to answer their analysis questions. Through a series of participatory design sessions with our user groups, we assessed the benefits of FCA in their analysis goals and investigated alternative ways to improve the visual representation and exploration of the lattice structure.

This chapter discusses our proposal for visual analytics for concept lattices, where new visualisations are proposed together with novel visual analytics features that enhance

data representation and support different analysis tasks (Section 4.3). We extended the visual analytics to Association Rules, as we explain in Section 4.4. Finally, in Section 4.5, we propose a tree extraction algorithm to simplify lattice browsing and visualisation while preserving the most essential features of the original structure. As we will discuss in Section 4.5, trees are common and have easily understandable visual representations. We consider them as a visualisation alternative to large cluttered concept lattices, which preserves all lattice entities and some of its structure. Finally, Section 4.6 introduces **Cubix**, an analytics tool for Formal Concept Analysis that implements all proposed visual analytics features. It runs in the distributed architecture described in Chapter 3.

4.2 Visual Representation of Concept Lattices

Concept lattices carry meaningful information on how formal concepts are related to each other and their properties are well defined in the FCA literature (Gerd et al., 2002a, Eklund and Villerd, 2010, Kuznetsov and Obiedkov, 2001, Wille, 1989). As R. Wille noted in his 1989 paper (Wille, 1989):

> *"Lattices in data analysis are more than just mathematical structures: They carry meaning. Therefore, drawings of such lattices should not only reflect the mathematical structure but also give a meaningful presentation for the data." (p.2)*

The *Hasse* diagrams used in FCA are usually layered graphs, where concept vertices are assigned to horizontal layers according of the number of common attributes in each concept, and are ordered within each layer to reduce edge crossings. FCA lattices in particular suffer from considerable edge crossings, especially if the number of concepts exceeds a few dozen; this is unfortunately the case in most real word applications (Roth et al., 2008), which leads to reduced graph readability and aesthetics (Ware et al., 2002).

A good lattice representation will provide a clear overview of the structure, allowing a clear understanding of specialization and generalization for a given concept. Eades and Tamassia (1988) listed the following qualities for a graph drawing algorithm:

- display of symmetry;

- avoidance of edge crossings;
- avoidance of bends in edges;
- uniformity of edge lengths;
- uniformity of distribution of vertices.

For general graphs, the problem of a determining a planar layout of a graph with least edges crossing (the *Crossing Number*) is NP-hard. Therefore some heuristic methods are used, like the **force based** layout algorithms where initial vertex placement by continuously moving the vertices according to a system of physical forces.

Layout algorithms like the **additive line** diagram (Ganter and Wille, 1999) have the advantage of being configurable through the choice of the representation set, and the technique produces a high number of parallel edges, which in turn improves readability.

4.2.1 New Alternative Visualisations for Concept Lattices

We propose alternative visualisations for the concept lattice, based on the Hasse diagram and tree-like visualisations. In order to display a tree visualisation the concept lattice has some of its edges pruned, as we explain in Section 4.5. Each visual component of the interface is seen as a *view*, i.e. a "perspective" on the underlying data, in our design pattern model. This is important to keep the visualisation states synchronized across the analytic features, as we explain in Section 4.4.1.

In the following sections we discuss the three proposed visualisations for the Hasse diagram: matrix, sankey and heat map.

4.2.1.1 Matrix

A **Matrix** is a traditional and compact way to visualise dense relationships relationships among entities. Figure 4.1 shows a new matrix visualisation for concept lattices, where objects are displayed in rows and attributes in columns. Each formal concept is depicted as coloured, overlapping layers in the relation. This visualisation also reacts to mouse over and highlights the same concept in other object $\times$ attribute squares.

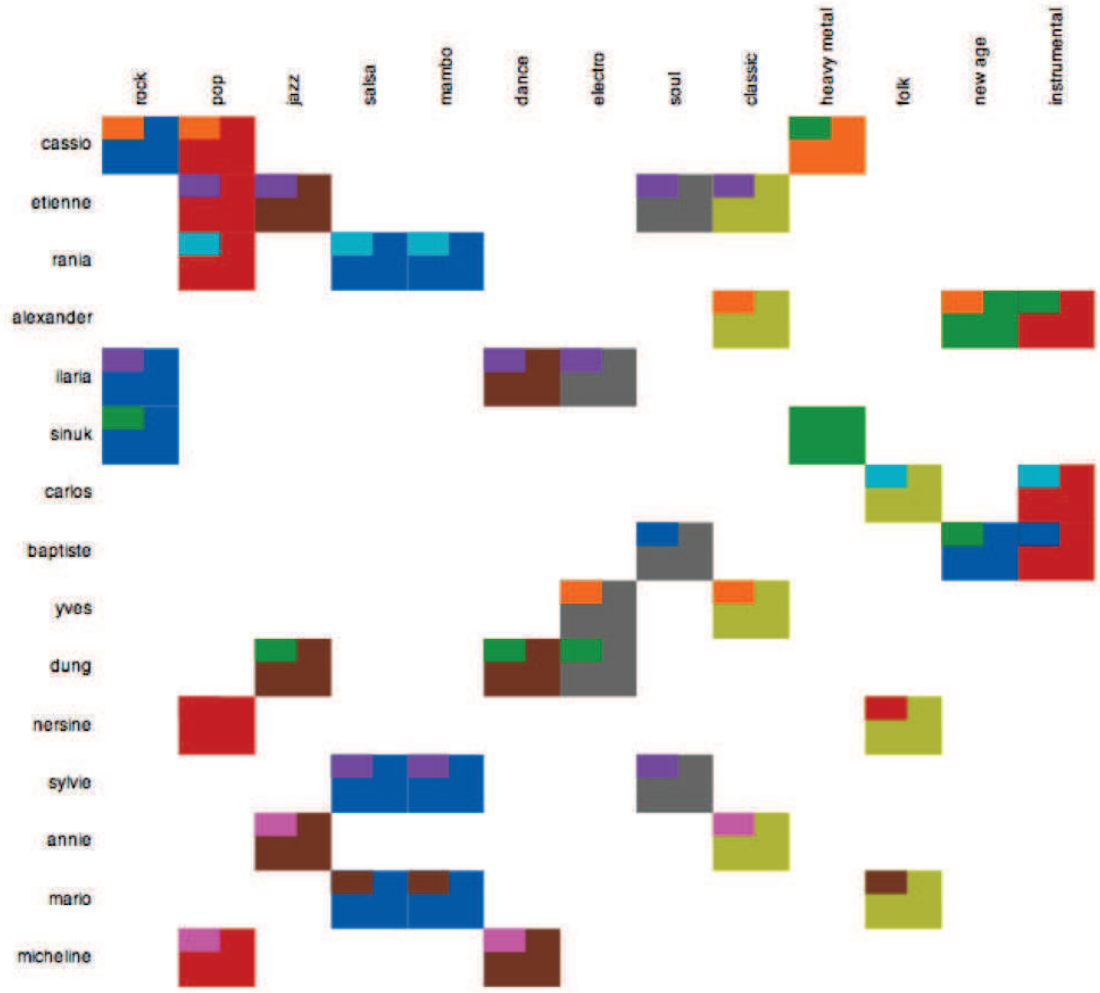


FIGURE 4.1: Visualisation of a concept lattice using matrix.

The relevant part of the drawing algorithm is illustrated in Algorithm 11 in Appendix B. First, it iterates over each relation I in the context $\mathbb{K}$. Then it iterates over the cells of a object-attribute matrix drawing rectangles on the cells for each concept found in that relation (lines 4-17). To allow user visualisation and selection, the inner rectangles have smaller widths and heights than its predecessor (lines 15 and 16).

This visualisation is particularly interesting for BI experts because of their familiarity with tabular data. Additionally, it facilitates the understanding of the relations among entities through a conceptual point of view, i.e, noticing which sets of objects and sets of attributes may form a concept. Whilst the matrix visualisation for concepts was well received by our users, it is limited to certain applications. It only makes sense with a small set of objects and attributes that can fit on a screen, since scrolling could potentially cause the user to loose reference of the same occurring concept on the cells out of

the visible screen. Second and most importantly, it is not able to display hierarchical features of a concept lattice[1]. To tackle specifically that issue, an alternative to the matrix visualisation is a variant of a diagonally-filled matrix, called **GeneaQuilts** (Bezerianos et al., 2010), which can represent hierarchies through links between the different layers in the matrix. A major drawback is, however, that the drawing of links between non-subsequent layers may result in multiple edges crossing.

4.2.1.2 Sankey

Another proposed visualisation for concept lattices is the **Sankey** diagrams (Schmidt, 2006). This class of diagrams are usually employed to depict flow, in which the width of the arrows is shown proportionally to the flow quantity. In the case of a concept lattice, this flow can correspond to the actual links between concepts. There is a strong focus on how entities are connected, therefore, it can be used to guide the exploration of links in the concept lattice. One advantage of this visualisation is that it makes easy to see, for instance, how objects are boiled down in throughout the hierarchy of concepts. Figure 4.2 shows an example of this visualisation for a concept lattice.

Any Hasse drawing algorithm can be used as a basis for the Sankey diagram for concept lattices, provided the following changes: The diagram is rotated 90 counter-clockwise, so it is read from left to right. The width of a node is equal to the sum of the thickness of the edges connecting to it. Edges follow a Bézier path to give an idea of flow. The space between the layers are noticeably larger than the usual Hasse diagram.

4.2.2 A "Heat map" Visualisation for Multi-Valued Concepts

Similarity Formal Concept Analysis (SFCA) compute formal concepts without the need of data scaling, multi-valued attributes are grouped together with respect to a similarity threshold (Messai et al., 2008). The result of SFCA are *multi-valued concept lattices*, in which concepts may have attributes with continuous values.

To help the analyst to quickly identify intervals in the concepts and compare with other concepts in the lattice, we designed a new visualisation based on "heat maps". In this visualisation, each concept is depicted as an array of rectangles (Figure 4.3). Each

[1]Unless we establish a drawing order to the overlapping rectangles, for example.

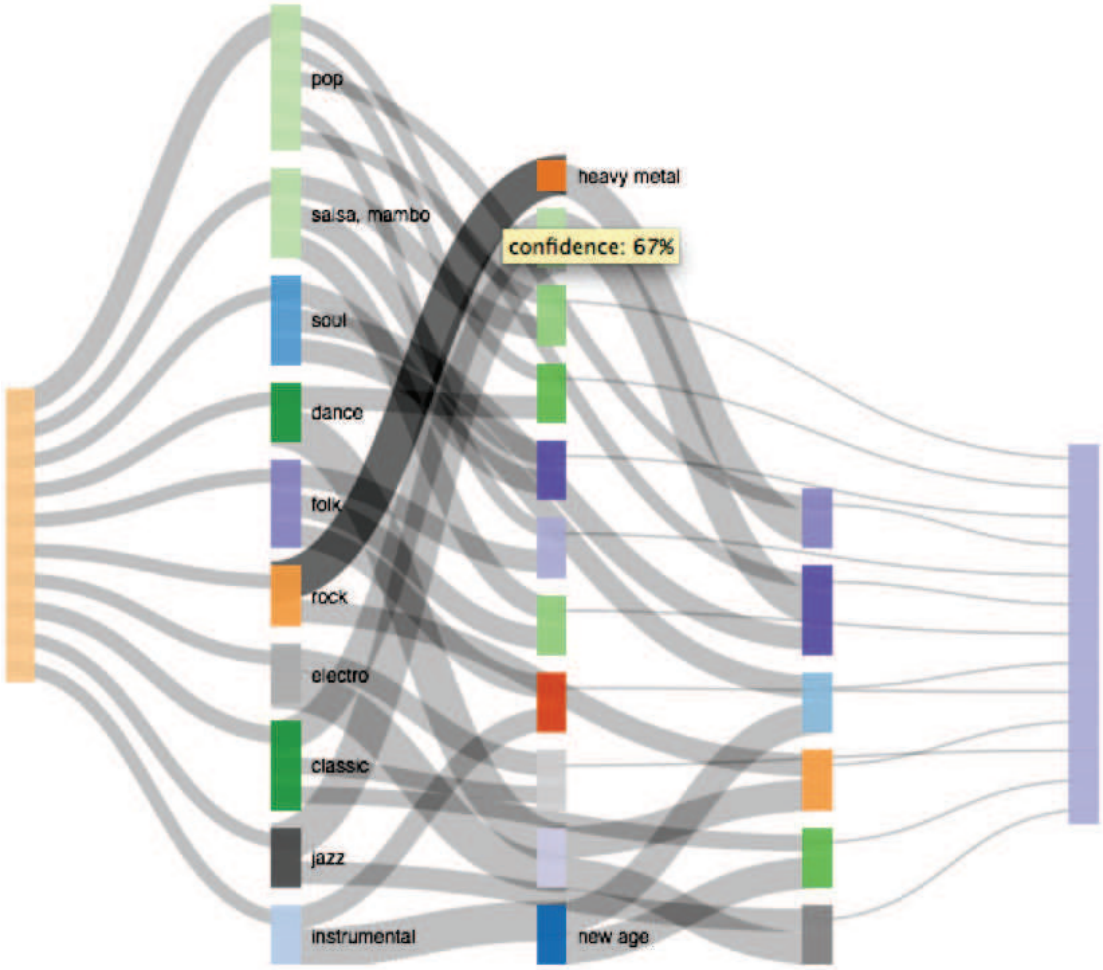


FIGURE 4.2: Visualisation of a concept lattice using the Sankey diagram. Edges thickness is given by the confidence value of the implication between two concepts.

rectangle represents an attribute, its colour indicates the interval of the attribute value in a continuous colour scale from blue to red. Its width is proportional to the size of the range. If an attribute is not present in the concept the corresponding rectangle is empty in order to keep the position of the rectangles consistent.

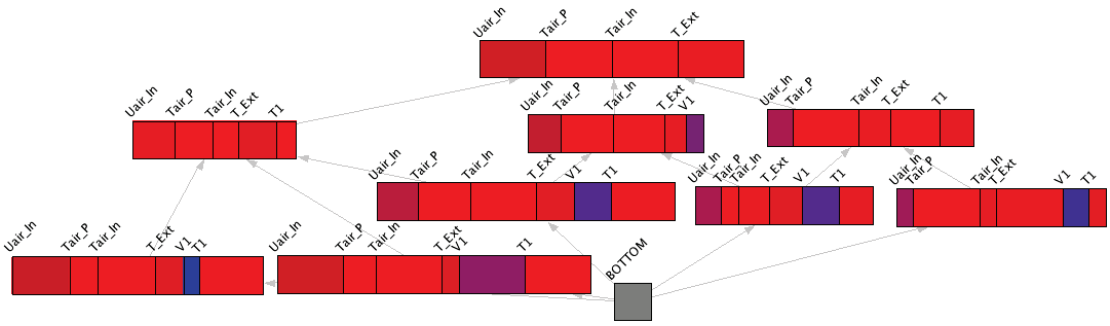


FIGURE 4.3: A "heat map" visualisation for concept lattice. Colour indicates position in the range (from blue to red), width shows the length of the interval.

4.3 Visualisation of Association Rules

It is possible to compute association rules from formal concepts via subset generation. Association Rules are of the form $\{premise\} \implies \{conclusion\}$, e.g. {"fly", "lay eggs"} $\implies$ {"bird"} and are known to establish implication among categorical variables. One of the most known use case is the *market basket analysis*, which seeks to discover products that are often brought together, for example, *butter* is often bought together with *milk*. Whilst the example is obvious, there are cases where association rules can highlight interesting relationships. For example is known that *babydiapers* and *beer* are often bought together (fathers going to buy nappies for their babies, would buy beer as well).

Because most items are categorical, association rules carry very little information about the way they can be visualised. Data mining software typically represent them as a list of logical sentences, impractical for a large number of rules.

We implemented two new visualisations combined with statistics and charts to enable progressive exploration of the *ruleset*. A **Matrix** view display each rule in a row and the concerned pairs of attribute-value in columns (Figure 4.4). In this visualisation, the colour of the cells indicates whether the attribute is in the premise (purple) or in the conclusion (yellow) of a rule. The confidence of each rule is represented by the opacity of each cell: the brighter the colour, the higher the confidence is. Rules are sorted from premise to conclusion and in lexicographic order for convenience.

In the example of Figure 4.4, rule **id15** has the highest confidence. It represents the implication between $\{dance\}$ and $\{electro\}$ music. The matrix visualisation provides a good overview of the ruleset and the integration with the analytics features allows filtering and drilling down of rules.

The second visualisation is a **Radial** graph showing how attribute-value pairs imply one to the other (Figure 4.5). The confidence of a rule is represented by the thickness of the connecting line. The connecting line is a gradient from purple to yellow, to indicate premise and conclusion respectively. The algorithm to generate this visualisation has a special treatment to position labels. The X, Y coordinates of a label and the angle of rotation is given by the following equations:

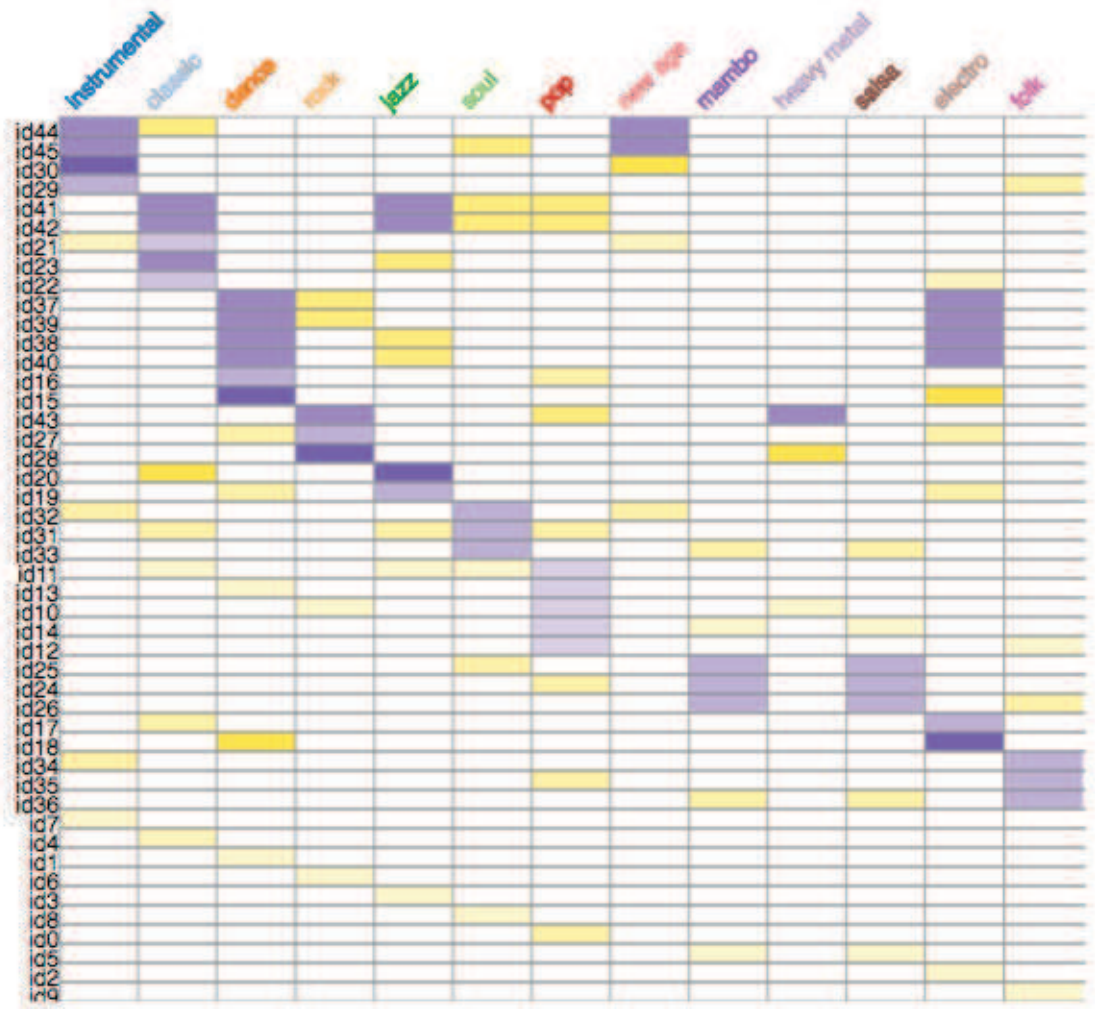


FIGURE 4.4: Visualisation of Association Rules using the Matrix visualisation.

$$posX = \sum_{i=0}^{|M|} \cos(2i\pi) \tag{4.1}$$

$$posY = \sum_{i=0}^{|M|} \sin(2i\pi) \tag{4.2}$$

$$\theta = \arctan\left(\frac{posY}{posX}\right) \times \frac{180}{\pi} \tag{4.3}$$

The positions follow a circular path given by $PosX$ and $PosY$. The label is rotated θ degrees to "flip" the label to be read conveniently, i.e., avoid placing labels up-side down as illustrated in Figure 4.6.

FIGURE 4.5: Visualisation of a concept lattice using the Radial graph visualisation.

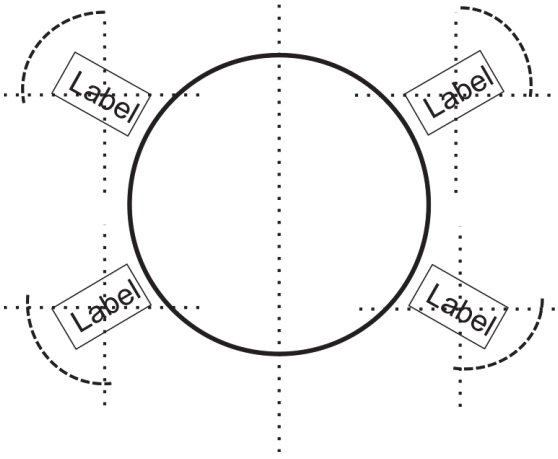


FIGURE 4.6: Illustration of labels positioning in the Radial AR Visualisation.

In the next section, we describe the proposed interactive features that play together with the lattice visualisation.

4.4 Visual Analytics of Concept Lattices

Visual analytics can be seen as an integral approach combining visualisation, human factors and data analysis (Keim et al., 2008). According to Thomas et al., *"Visual analytics is the science of analytical reasoning facilitated by interactive visual interfaces"* (Thomas and Cook, 2006). It is the tight integration of computational analysis by algorithms and highly interactive visualisations. The human intuition and background knowledge are key aspects in the process, orchestrating the interface elements to produce (or induce) a desired state that will ultimately lead to some knowledge gain. The system, on the other hand, has to provide meaningful representations and exploration methods on the data. Interactions that affect the state of the interface should provide smooth transitions to help users cognitive memory and traceability.

The potential of using analytics features to enhance FCA was acknowledged as early as in 1995, when Carpineto & Romano implemented a visual interface for large lattice called **Ulysses** (Carpineto and Romano, 1995). They suggested a *"Fish eye"* representation of concept lattices so that the focus on one concept node would expand its similar neighbours proportionally. Ulysses allows users to reduce the search result space by adding constraints to the lattice. Later with **CREDo** (Carpineto and Romano, 2004a) only parts of the lattices are displayed similar to file/folder displays, where a second level of the hierarchy is indented and can be expanded or collapsed interactively by users. Priss (2000) used the lattice representation to show the concepts hierarchy in thesauri. Each concept is viewed as a facet in an information retrieval system. Akand et al. (2010) propose an algorithm that generates a browse-able concept lattice designed for biology applications. Villerd et al. (2007) used visual analytics in an indexed document collection to display concept lattices in a global view and concept details in a local view. The system expands/collapses information according to selections made in the concept lattice. More recently, Bach (2010) proposed an interface, called **Facettice**, for faceted navigation and data analysis using interactive Hasse diagrams. In the diagram, nodes have additional information encoded as bar charts to depict the distribution of values for a given attribute in the concept. Additional *attribute-lattices*[2] are visualised and allows filtering by node clicking.

[2]Concept lattices generated for each attribute and its values.

In the next sections we describe each of the proposed analytic features for concept lattice: a dashboard, the use of visual variables to enhance lattice exploration, search and selection, visual filters and clustering. All the proposed analytics work analogously for concept lattices and association rules. The techniques were implemented in **Cubix**, which will be discussed in detail in Section 4.6.

4.4.1 The Model-View-ViewModel (MVVM) Pattern

Typical visual analytic tasks include exploration, filtering, selection and data transformation. All interface elements must be consistent at each operation. For example, when the user searches for "*all retail stores with revenue above X*" he or she will expect that the visualisations change to match with his or her query.

Maintaining a consistent state across multiple visualisations and triggering updates only to concerned parts is a challenge faced by modern analytics systems. To cope with this, we implemented a variant of the Model-View-Controller pattern, called **Model-View-ViewModel (MVVM)**[3]. A ViewModel is an abstraction of the view that serves to mediate between a view (e.g. Hasse diagram) and a model (e.g. concept lattice). The application maintains a synchronized state through event handlers (for a detailed explanation on MVVM, see Hall (2010)). For example, when the user click on a filter, the action is sent to the data model which in turn triggers the corresponding event listeners causing the visual components to be updated. It also allows switching visual components without affecting the underlying data structure.

4.4.2 Dashboard

In addition to the main concept lattice visualisation, several charts display different aspects of the underlying conceptual structure such as co-occurrence of attributes, concepts distribution, stability *versus* support, etc. (Figure 4.7). Some charts are updated when the user points the mouse over a concept, highlighting details of the concept. Similarly, a selection of a point/series in the chart will highlight the concerned concepts in the lattice. This technique is called *Linking and Brushing* (Ward, 2009).

[3] "ViewModel" is usually spelled without spaces, supposedly to refer to a combined entity.

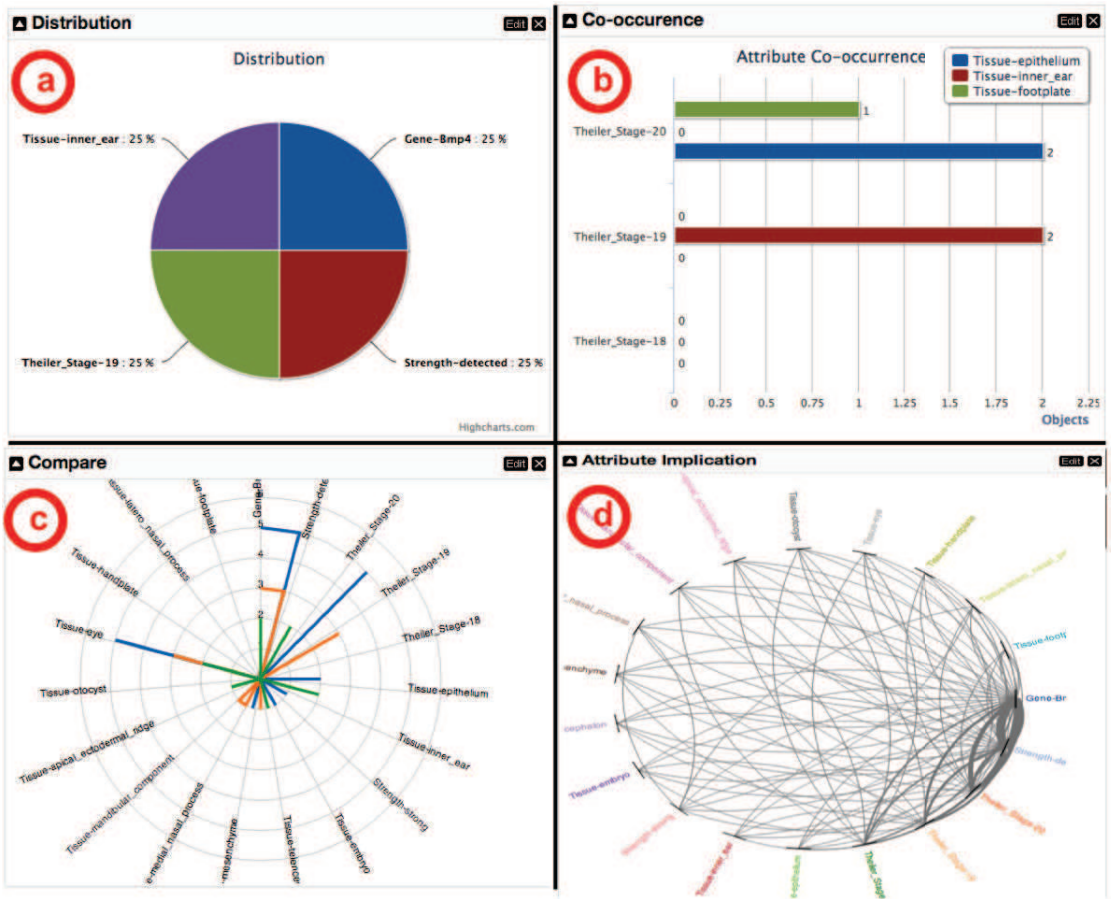


FIGURE 4.7: Dashboard for FCA a) Distribution chart, b) Co-occurrence chart, c) Comparison chart and d) Attribute implication graph.

Distribution chart. This chart show the distribution of attributes/values of a given concept when the user places the mouse pointer over it (Figure 4.7 - a).

Co-occurrence chart. Some FCA tasks involve discovering co-occurrence patterns among two or more attribute values. This can be done with the co-occurrence chart (Figure 4.7 - b).

Comparison chart. This polar diagram is activated when the user selects two or more concepts in the lattice for comparison. Each attribute is represented by an axis and each concept value is draw as a line of a particular colour. In the example of Figure 4.7 - c, three concepts are selected for comparison.

Attribute Implication graph. The attribute implication chart displays the implication relationship among attributes. The stronger the implication between two attributes, the thicker the line connecting them both (Figure 4.7 - d).

4.4.3 Visual Variables

Common analytic techniques include the assignment of different colours, shapes and sizes to nodes and edges, according to different dimensions or properties. This approach is underused in traditional lattice visualisations, where the main visual variable used is node/link colour to reflect user selections or node size to indicate the immediate presence of an extent or intent as displayed in *ConExp*[4].

We use these as well as other visual variables in a Hasse diagram to enhance the understanding of conceptual data. Prominent features of the lattice like specialization and generalization can be better understood, for instance, the confidence of implications of different concepts can be rendered by edge thickness. Figure 4.8 shows a concept lattice where the size of the node is proportional to its *support*, the confidence by edge thickness and colour to represent the cluster the concept belongs to.

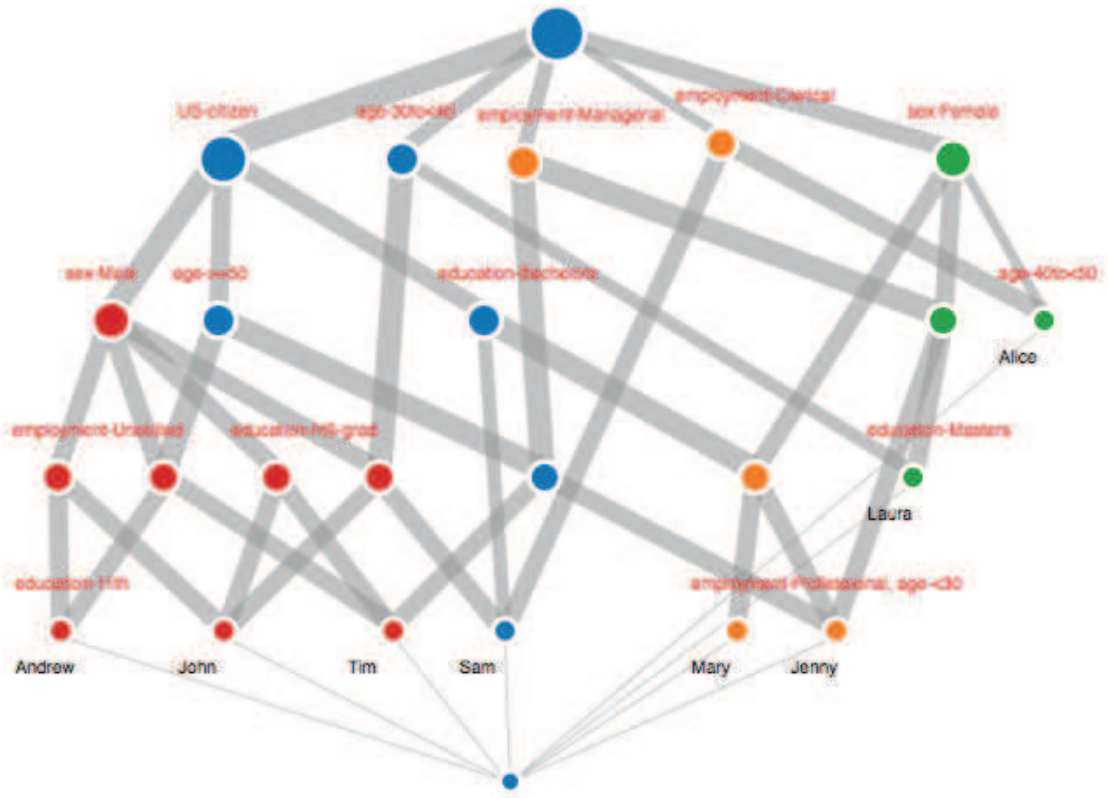


FIGURE 4.8: Concept lattice enhanced with visual variables depicting different properties.

[4]Concept Explorer (ConExp) - www.conexp.org

4.4.4 Search and Selection

Search and selection are two of the most common operations in decision-support systems.In our analytic tool, a concept can be selected manually (by clicking) or by searching for properties in the concept lattice.

A search bar provides auto-completion of text and highlights the selected concepts in lattice (Figure 4.9).

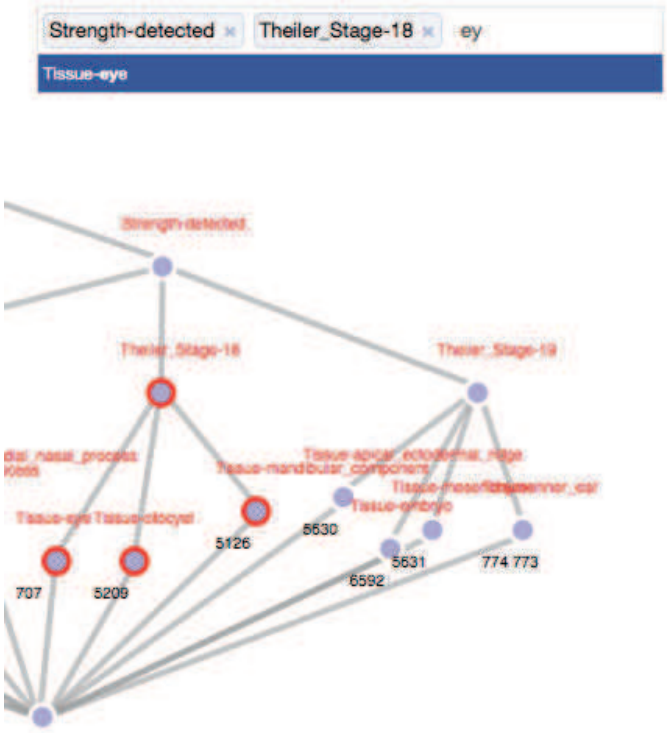


FIGURE 4.9: Search in the concept lattice.

4.4.5 Visual Filters

The number of concepts can grow dramatically with the number of objects and attributes, yielding poorly readable concept lattices. The filter bar displays the current distribution of attributes in the lattice and allows the user to visually select and filter concepts (Figure 4.10). The filter bar has an additional visualisation, which is a *lattice view* for each attributes (Figure 4.11). This visualisation allows identification of attribute-value correlations and it is particularly interesting for complex multi-valued attributes. A history of activated filters is displayed at the bottom and it allows the user to remove a given filter non-sequentially.

It is worth noticing that the filters act on the concept lattice rather than on the context itself. Current FCA tools such as ToscanaJ and Conexp, allow filtering on the context, and the lattice needs to be computed again.

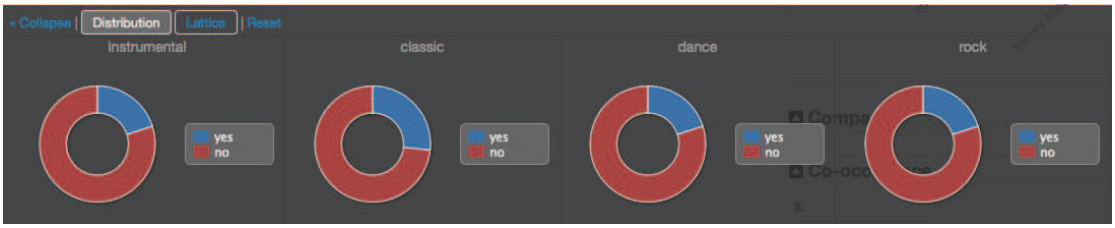


FIGURE 4.10: Visual filter bar.

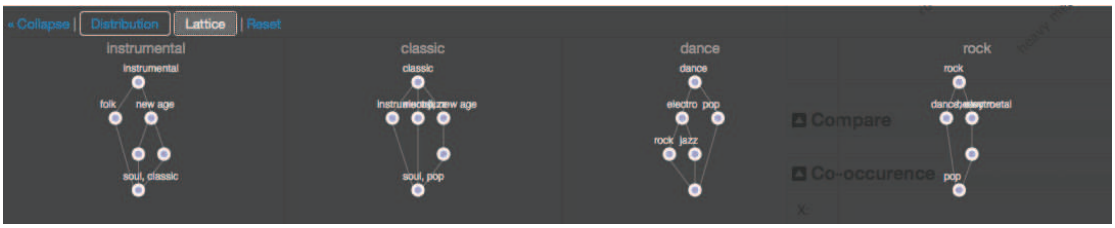


FIGURE 4.11: Lattice filter bar.

4.4.6 Concept Clustering

Clustering of concepts can be useful to facilitate the analysis and to identify zones of interest. Some similarity measures are based on the concept lattice topology (e.g. counting the number of links between two concepts); Intent/extent similarity (e.g. *Jaccard*); or confidence between two pairs of concepts.

Concept similarity (Jaccard). It is a coefficient for calculating the ratio of shared attributes between concepts. We define concept similarity as:

$$CSim(A,B) = \frac{|m_a \cap m_b|}{|m_a| + |m_b|} + \frac{|g_a \cap g_b|}{|g_a| + |g_b|} \tag{4.4}$$

Proximity. Conceptual proximity is the topological distance between concepts A and B in the concept lattice.

$$prox(A,B) = 1 - \frac{shortestDistance(A,B)}{diameter(Lattice)} \tag{4.5}$$

Strength. It is the average concept similarity value (CSim) along the shortest path between a pair of concepts.

In the example of Figure 4.8, a *k-means* clustering algorithm (Hartigan and Wong, 1979) was used to identify clusters (the number of clusters is defined by the user).

4.5 Tree Extraction and Visualisation from Concept Lattices

Browsing concept lattices becomes a problem as the number of clusters grows significantly with the number of objects and attributes. Interpreting the lattice through a direct visualisation of the Hasse diagram rapidly becomes difficult and more synthetic representations are needed. A common approach is to show or hide parts of the lattice via interactive exploration of subsets of terms or neighbours of a focus concept (Ducrou et al., 2008). Carpineto and Romano (2005) defined constraints to be applied to the concept lattice in order to simplify lattice querying and navigation.

Trees are good alternatives to represent concept lattices in this context, because they do not suffer from edges crossings and they are natural metaphors for navigation history since there is only one parent per child. Additionally, users are familiar with that structure as trees are used to navigate through folders and files in most operating systems. In (Carpineto and Romano, 2005) it was noted that trees are particularly interesting structures to represent concept lattices for browsing, however, authors pointed out that their main disadvantage is the amount of replicated information when concepts have multiple parents. In the current work we avoid duplication by selecting only one parent for each concept in the lattice. One inherent challenge is to formally define the notion of best parent among the potentially numerous parents of a concept. A previous work by Le Grand et al. (2009) used an approach to extract trees from lattices based on the assignment of weights to attributes. Concepts with higher average weights were selected as parents while the other concepts were removed from the resulting tree.

Our approach consists in representing lattices not as Hasse diagrams, but as trees. As we explain in the next section, we propose different criteria to extract trees from lattices and visualise the resulting trees. Trees are inherently simpler hierarchical structures

than Hasse diagrams and due to their applicability in many domains, there is a plethora of tree representations. These include: *indented outline trees*, sometimes called "tree lists" (common in file browsers such as windows Explorer), traditional *layered node-link diagrams* in 2D or 3D (e.g. **ConeTrees** - Robertson et al. (1991)), *spatially transformed tree diagrams* (e.g. **Radial trees** Bachmaier et al. (2005)) as well as several *space optimization* (**Space Optimized trees** Nguyen and Huang (2002)) and *space-filling tree visualisation* techniques (e.g. **TreeMaps** Johnson and Shneiderman (1991)).

4.5.1 Tree Extraction based on the Selection of One Parent per Layer

In a previous work by Le Grand et al. (2001), authors presented an approach to extract trees from concept lattices based on the selection of one parent concept per hierarchical layer in the lattice. The process starts by collecting all most specific concepts (i.e. the lower bound parent concepts at the second lowest layer of the concept lattice) so all objects are present in the new structure. A single parent concept is then selected for each of these concepts, and the process continues recursively until reaching the top of the lattice. The choice of a single parent for each concept therefore removes links and eventually concepts[5].

The goal is to select parents according to their relevance from user's point of view, for example, taking into account the attributes that are most significant to him or her. The challenge is to find the set of criteria that best suits users' expectations. Trees extracted from the same concept lattice can be very different, as shown in Figures 4.13 and 4.13, which represent two distinct trees extracted from the lattice of Figure 4.12. Notice that the concept C6 is lost in the tree extraction process illustrated in Figure 4.14.

Trees are displayed in three different forms: the left one (Figure 4.13(a)) represents the tree as extracted from the lattice, the one in the middle (Figure 4.13(b)) is the traditional representation of a tree without crossing links, and the visualisation on the right (Figure 4.13(c)) shows the layers of the tree as concentric arcs, from the most generic (outer layers) to the most specific (inner layers). The size of each arc is proportional to the number of children in the corresponding sub-tree.

[5]If a concept is not selected as a single parent by any of its children concepts, it is removed from the lattice

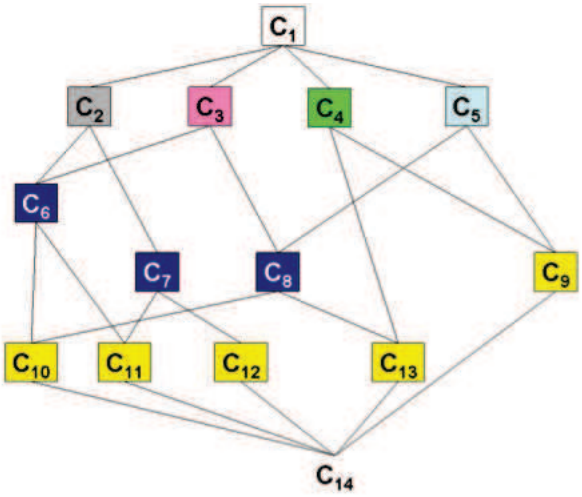
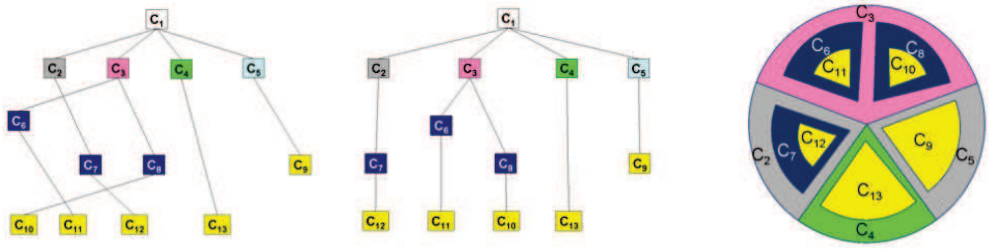
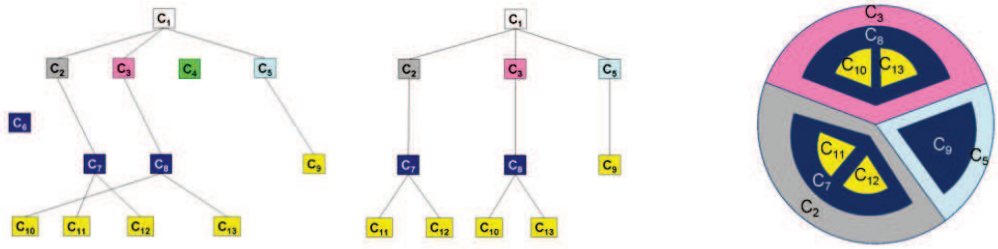


FIGURE 4.12: Original concept lattice.



(a) Original lattice (b) Tree from the lattice (c) Radial visualisation

FIGURE 4.13: An example of tree extraction without concept loss.



(a) Original lattice (b) Tree from lattice (with concept loss) (c) Radial visualisation

FIGURE 4.14: An example of tree extraction with concept loss.

A hierarchy of parent selection criteria was defined in (Riadh et al., 2010) to extract a tree from the concept lattice generated from on a corpus of 126 web pages dedicated to tourism. The algorithm works as follows:

- For each concept, called the current concept, it selects the parent concept which is located at the shortest distance from the upper bound of the lattice (in order to

minimize the number of links);

- if several parent concepts satisfy this condition, it favours those which have already been selected as a single parent by other concepts of the same level as the current concept;
- If the last condition is not sufficient, it selects the concept with higher average of attribute weights, previously assigned by the user.

Figure 4.15 shows the conceptual tree extracted according to these criteria from the lattice generated from the tourism data set. Outer arcs represent most general concepts, i.e., those situated on the top in *Hasse* diagrams. To facilitate the interpretation of the tree, the intent of each of the concepts in the top layer is displayed. The surface of each arc is proportional to the number of leaves of the corresponding sub-tree to provide a glimpse of the most significant topics in the data set, in this case pages devoted to France, food, camping, and the region Loire. This visualisation has some limitations. While it is easy to see the top-most categories, children nodes have smaller drawing space than their parents, even though they are more numerous. The nested nodes in the tree are thus harder to discern. We addressed this issue by inverting the drawing order of parents and children, as we explain in Section 4.5.3.

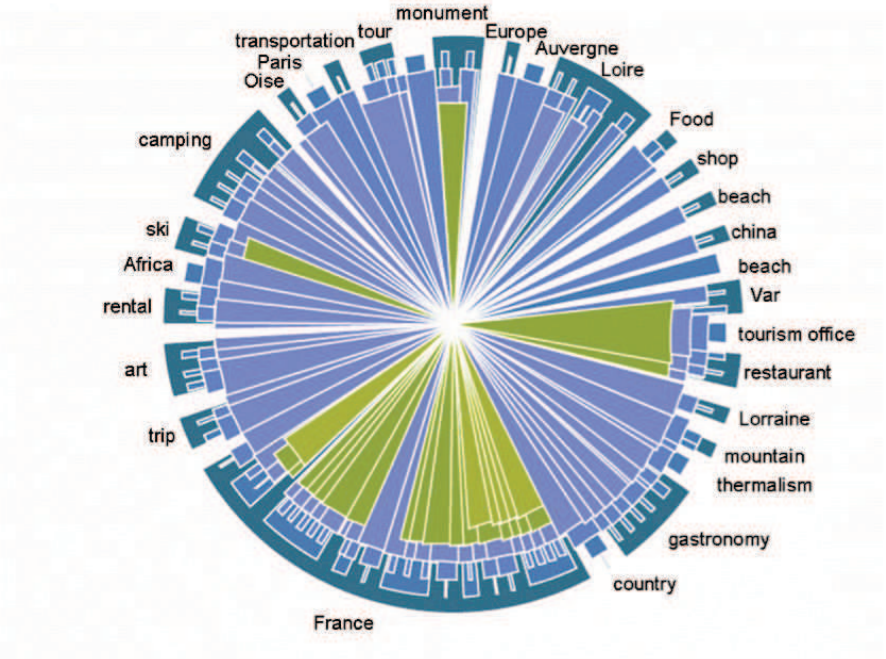


FIGURE 4.15: Tree extracted using the *one parent selection per layer* approach.

4.5.2 Tree-extraction based on Conceptual Indexes

The initial tree extraction approach presented above has one main disadvantage: the elimination of concepts which can lead to a loss of information. The present approach leverages conceptual measures as criteria to select parent concepts, keeping the link with the parent that scored the best with the considered index. As with the previous approach, the process starts with the most specific concepts i.e., at the bottom of the *Hasse* diagram and recursively computes the index for each of the candidates and eliminate links to all parent concepts except the one with the selected parent. The pseudo-code of the algorithm is provided in Algorithm 10.

Choosing a single parent concept at each step leads to some information loss, although in this case only links are removed. Our goal is to minimize this loss by selecting parents using the most relevant criteria according to the kind of analysis performed by the analyst. In the following sections we consider various strategies for selecting parent concepts, including the *stability* (Kuznetsov et al., 2007) and *support* (Gerd et al., 2002a) indexes from FCA literature, as well as the *confidence* and *similarity* measures.

Algorithm 10: ExtractTree

```
Data: A concept lattice ℒ.
1  begin
2      foreach concept ⟨A, B⟩ in Concepts(ℒ) do
3          Parents[⟨A, B⟩] ⟵ list of parent concepts of ⟨A, B⟩
4          if |Parents[⟨A, B⟩]| > 1 then
5              max_score ⟵ -1
6              foreach concept ⟨C, D⟩ in Parents[⟨A, B⟩] do
7                  Score[⟨C, D⟩] ⟵ score for concept ⟨C, D⟩
8                  if Score[⟨C, D⟩] > max_score then
9                      SelectedParent[⟨A, B⟩] ⟵ ⟨C, D⟩
10                     max_score ⟵ Score[⟨C, D⟩]
11                 end
12             end
13         end
14     end
15 end
```

4.5.2.1 Parent Selection based on Stability and Support

The stability index measures the proportion of subsets of objects of a given concept whose derivation is equal to the intent of this concept (Kuznetsov et al., 2007). In other words, stability indicates the probability of preserving a concepts intent while removing some objects of its extent. A more stable concept is less dependent on individual members in the extension. We recall the definition of stability:

Definition. Let $\mathbb{K} = (G, M, I)$ be a formal context and $\langle A, B \rangle$ be a formal concept of $\mathbb{K}$. The stability index of $\langle A, B \rangle$ is defined as:

$$stability(A, B) = \frac{|C \subseteq A | C' = B|}{2^{|M|}} \tag{4.6}$$

Using the context in table 1 as an example, we calculate the *stability* for concepts 3 and 5 in order to select a parent for concept 6 (0.25 and 0.5 respectively); we keep the one with highest *stability*, in this case we therefore remove the edge between concepts 3 and 6. The idea behind the choice of the parent concept with the highest *stability* is that we expect to keep parent concept's meaning even if some of the objects or attributes are removed. On the other hand, the support measure is the relation between the *intent closure* and the total number of objects (Gerd et al., 2002a):

Definition. Let $B \subset M$. The support count of the attribute set B in K is:

$$support(B) = \frac{|B'|}{|G|} \tag{4.7}$$

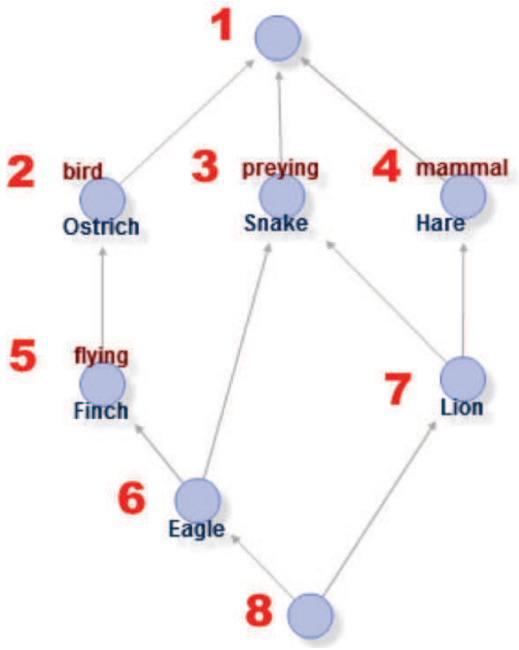
The use of support as parent selection criterion may lead to trees containing concepts with fewer specialization levels since generic concepts generally have higher support values than their most specific counterparts.

4.5.2.2 Parent Selection Based on Similarity

This measure relies on clustering parent and child concepts that share most of their attributes or objects. A parent and a child concept having a great number of attributes

Animals	Preying	Mammal	Flying	Bird
Lion	×	×		
Finch			×	×
Eagle	×		×	×
Hare		×		
Ostrich				×

(a) The formal context of Animals



(b) The concept lattice of Animals

FIGURE 4.16: An example of Concept Lattice.

in common are supposed to be grouped together following the principle of similarity clustering and local predictability (Hannan and Pogel, 2006). It can be defined as:

Definition. Let a concept $\langle A, B\rangle$ be such that $A \in G$ and $B \in M$. Let concept $\langle C, D\rangle$ be a child (specialization) of $\langle A, B\rangle$. The shared attribute index of an edge $E = \langle C, D\rangle \rightarrow \langle A, B\rangle$:

$$similarity(E) = \frac{|B \cap D|}{|M|} \tag{4.8}$$

In the same animals' context of Figure 4.16(a), we have potential parent concepts 3 and 5 sharing the same number of objects with concept 6, but concept 5 has more attributes in common with concept 6, so it should be chosen as the unique parent of concept 6 if the *similarity* criterion is considered.

4.5.2.3 Parent Selection Based on Confidence

The *confidence* value of a concept estimates how likely it is that an object with an attribute set A, also has an attribute set C (Ganter and Wille, 1999). In other words,

it tries to measure how strong the *implication* of the parent concept to a child concept is. For instance, considering the formal context in Table 1, what is the probability of a given object that is {*Bird, Flying*} to be also {*Bird, Flying, Preying*}? The following paragraph formalizes its definition.

Definition. Let a concept $\langle A, B\rangle$ be such that $A \in G$ and $B \in M$. Let concept $\langle C, D\rangle$ be a child (specialization) of $\langle A, B\rangle$. The shared attribute index of an edge $E = \langle C, D\rangle \rightarrow \langle A, B\rangle$:

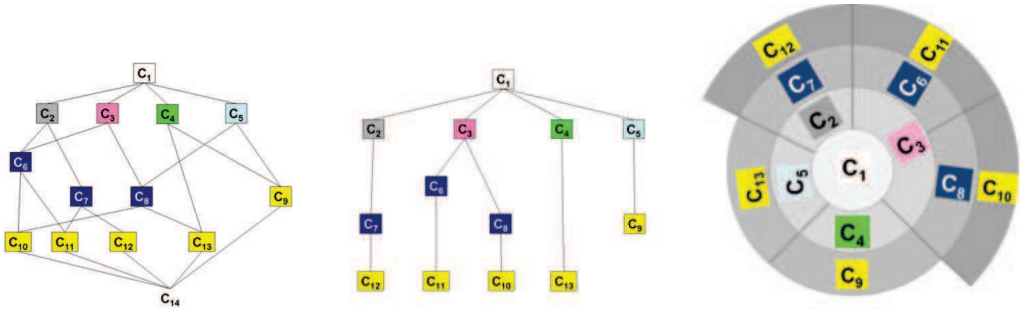
$$confidence(E) = \frac{|C|}{|M|} \tag{4.9}$$

An advantage of this method is its consistency with the interpretation of concept lattices. Taking our context as example (Figure 4.16), there is a 50% probability that an animal that is a {*Flying, Bird*} be also a {*Flying, Preying, Bird*}. By contrast, a {*Preying, Mammal*} has only 33% of chance of being also a {*Flying, Bird*}.

4.5.3 A "Sunburst" Visualisation for Concept Lattice

We improved the radial-filling visualisation shown Figure 4.12. Because the number of concepts tends to increase in the most specific levels, the previous visualisation suffered from decreasing inner space. We decided to invert the way the hierarchy is displayed, top nodes are now in inner layers while more specific concepts are located in outer, more spacious layers. This visualisation layout is called "Sunburst" (Stasko, 2000) and provides a overview of the distribution (depicted as the size of the arcs, Figure 4.17(c)).

One challenge is to assign label to concepts in a tree visualisation like the sunburst. Concepts are labelled using elements from its *intent* and/or *extent*. It may become an issue if the number of attributes or objects is too long to be displayed, causing text overlaps. In the sunburst for example, layers with a large number of concepts tend to have smaller arc size. A common technique to cope with this problem is to *only assign labels that were not previously assigned to any of its parents*, i.e., avoid repeating labels that already appeared in parent concepts. We used this technique combined with a logic that hides the label of a concept if the arc size is too small to fit the text. Optionally, the user can zoom in the areas of interest and hidden labels will appear accordingly.



(a) Original lattice (b) Tree extracted from lattice (c) Sunburst visualisation

FIGURE 4.17: Tree-extraction process and Sunburst visualisation.

4.5.4 Case Study of the Tree Extraction Process on the Tourism Web Pages Data Set

In this section, we discuss a case study of a concept lattice to qualitatively examine the nature of the trees resulting from different criteria. Figure 4.18 shows the Hasse diagram of a lattice containing 2,214 concepts and 7,758 links, constructed from a corpus of 126 web pages dedicated to tourism, characterized by their most common words from 60 words extracted from thesaurus.

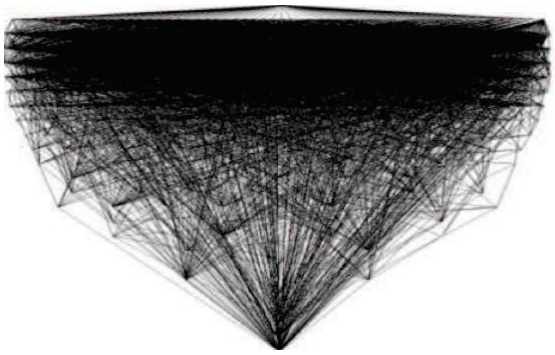


FIGURE 4.18: Concept lattice for the tourism web pages data set.

For the sake of the example, we selected a sub-context from the dataset containing only 87 concepts. In this particular context, the original concept lattice depicted by the Hasse diagram contained 216 edges between concepts, reduced to 82 after the tree extraction (38% of the original edges). Each of the proposed measures revealed particular aspects on the analysis of a lattice, as illustrated in Figures 4.19 and 4.20.

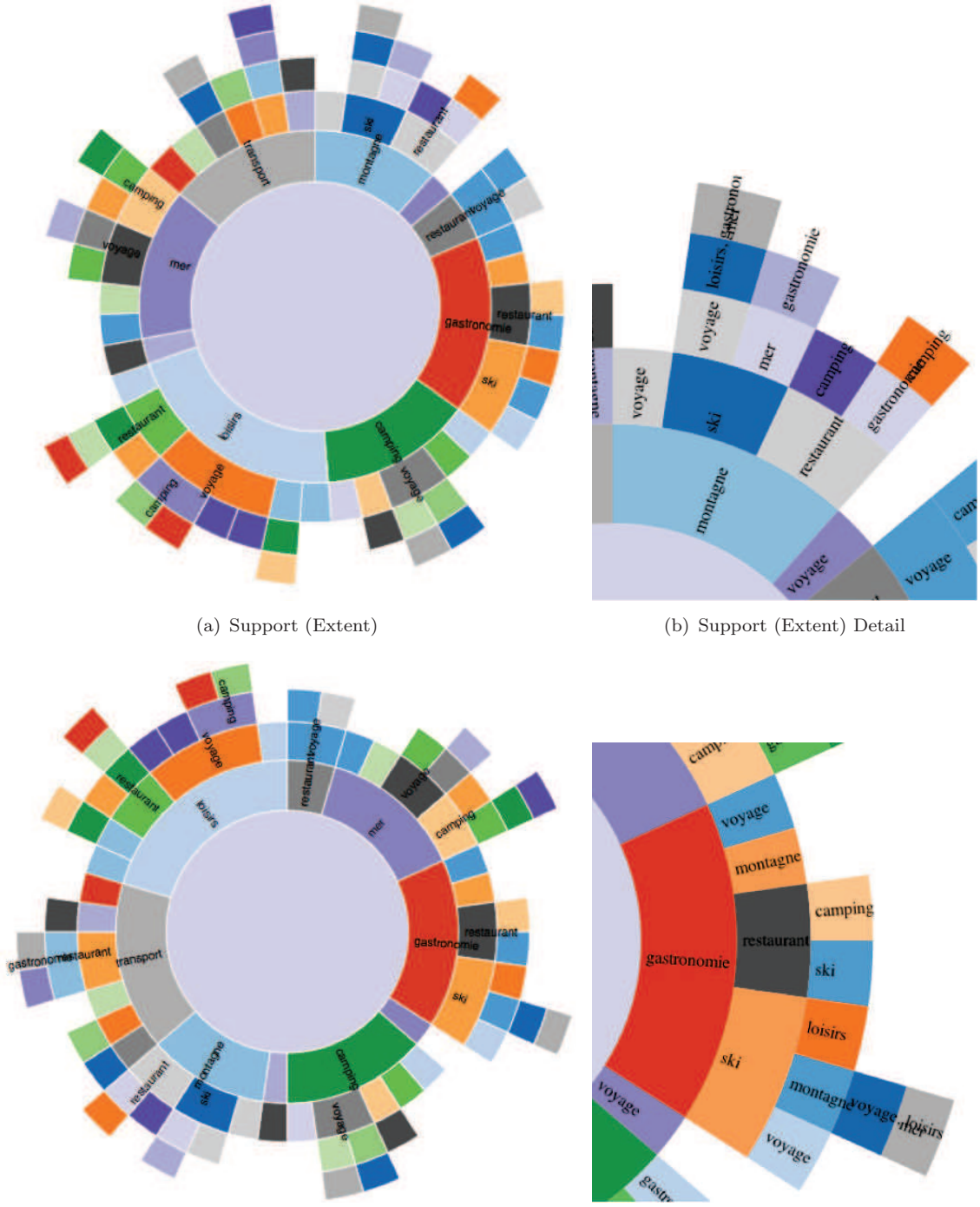
The left column in Figures 4.19 and 4.20 shows the complete tree extracted from the lattice using the corresponding criterion, whereas the column on the right displays a

particular part of the tree that we discuss. As explained previously, each concept is labelled in a non-repetitive way, i.e., labels assigned to parents are not be displayed in the respective children. Since labels can change from one visualisation to the other depending on the parent selected, colours were applied to each concept for easy identification (all concepts have the same colour in all visualisations e.g. "loisirs/voyage/camping").

Figure 4.19(a) shows the tree generated with *extent support* as parent selection criterion. This criterion tends to group concepts that have frequent objects in their extent. Notice that the concept containing {"montagne"} (mountain) accounted for many other concepts containing that attribute in contrast with confidence and similarity, for instance. This occurs because there are many web pages relating the keywords {"montagne", "ski"} and {"montagne", "restaurant"}, thus making the corresponding concepts more likely to be selected as parent in this criterion. On the other hand, the concept which contains {"ski"} has only one sub-concept {"voyage"} in contrast with the trees extracted using *confidence* and *similarity* (Figures 4.20(a) and 4.20(c)). We can infer that pages containing {"ski"} alone are not as popular as other keywords, because they are commonly associated with either {"montagne", "transport"} or {"loisir", "voyage"} as the corresponding concepts show.

The *stability* (intent) index was the criterion that generated the tree in Figure 4.19(c). With this index, concepts that retain the same set of web pages when some of their keywords are removed are the best candidates for parent selection. In the detail of Figure 4.19(d) the concept containing {"gastronomie", "ski", "montagne"} was the preferred parent to append the child concept {"gastronomie", "ski", "montagne", **"voyage", "loisir"**} contrarily to the selection made in the tree using support (Figure 4.19(b)) where the parent is {"montagne", "ski", "voyage"}. This suggests that despite having fewer occurrences (web pages), the concept {"gastronomie", "ski", "montagne"} has a smaller set of keywords that together correspond to the same pages. For instance, pages that talk about "gastronomy in the mountains" can cite "ski" as one activity to do in the mountains but the absence of that keyword is not crucial to describe the page.

Figure 4.20(a) depicts the tree generated by the *similarity* criterion. Parent concepts sharing most objects with a particular child concept were the best candidates. As an example, the concepts involving {"gastronomie", "ski"} were appended to the concept {"ski"} because there are more pages in common with the later than there are pages



(a) Support (Extent)

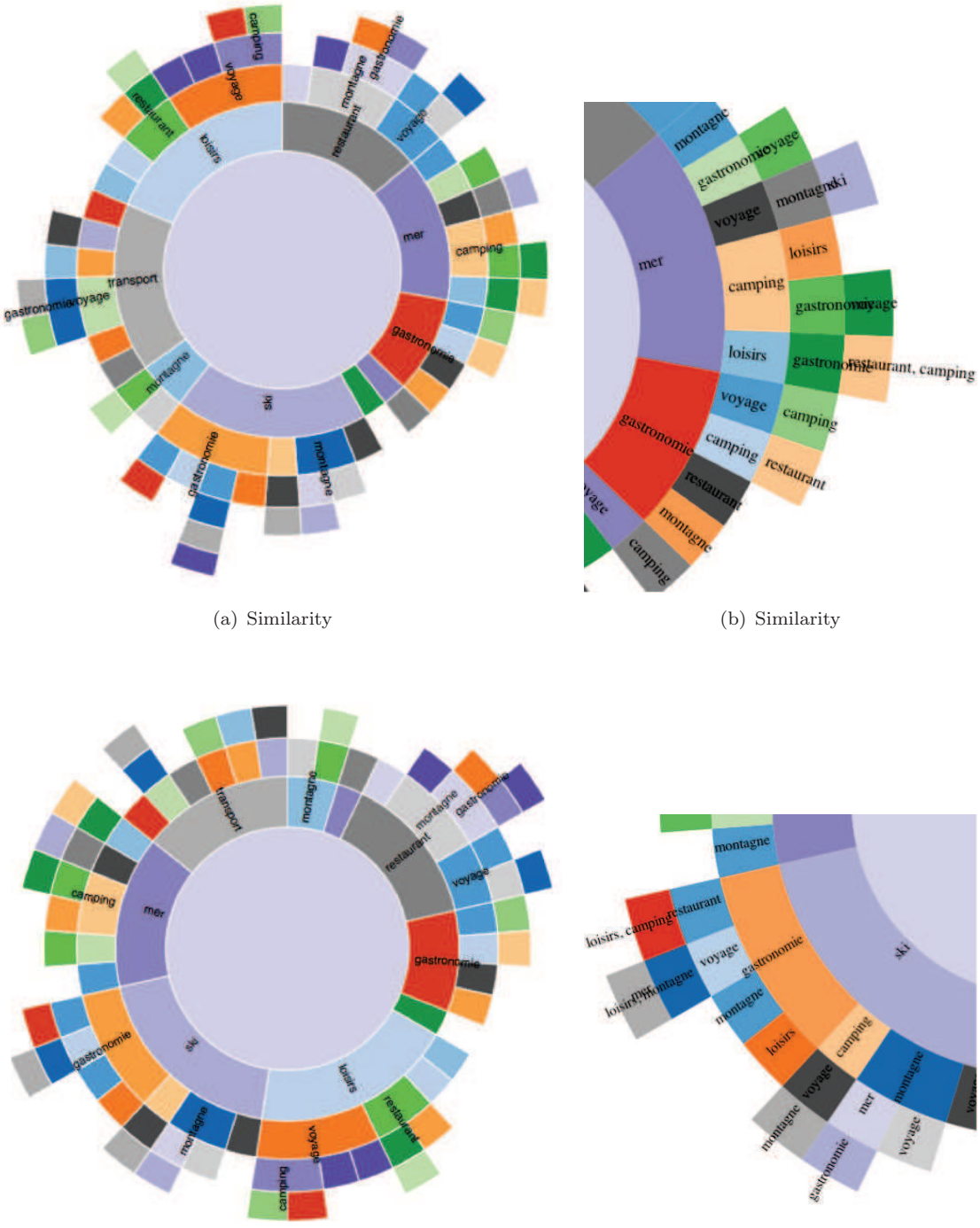
(b) Support (Extent) Detail

(c) Stability (Intent)

(d) Stability (Intent) - Detail

FIGURE 4.19: Trees generated from the lattice in Figure 4.18 using Support and Stability as parent selection criteria. Colour is assigned for each concept for comparison.

shared with the concept {"gastronomie"}. The same rationale explains the same parent selection for the confidence criteria: a page containing both keywords "gastronomie" and "ski" is more likely to be about "ski' than about "gastronomie". These choices contrast with those using *stability* and *support* (Figures 4.19(a) and 4.19(c)).



(a) Similarity (b) Similarity

(c) Confidence (d) Confidence

FIGURE 4.20: Trees generated from the lattice in figure 4 for each proposed measure.

In Figures 4.20(c) the tree was generated using the *confidence* criterion, i.e. children concepts are associated with the parent to which the relationship of confidence is the highest among the candidates. As a consequence, the concept {"voyage"} was the parent selected for the concept containing {"camping", "voyage"} as opposed to the concept

{"camping"}, because usually camping implies going on a trip whereas going on a trip does not necessarily mean to go camping.

4.5.5 Guidelines on How to Choose the Parent Selection Criteria

The extraction of a tree from the original lattice implies that some links are removed and in some cases this can be misleading, depending on the selection criteria. Nevertheless, the process is able to keep some of the most essential conceptual features and provides different perspectives to the analysis, for example, on *frequent patterns* or on the *implications* between the concepts. It is also possible to display different trees at once in order to get a more comprehensive view.

When two parent concepts have the same score, the parent with the first computed score is selected by the algorithm. This could lead to an undesired non-deterministic result for the same input parameters. We stress that even non-deterministically in those cases, the process yields results consistent with the chosen metric. One way of tackling the issue is to define "tie-break" scores, i.e., other indexes to be used to define the best parent. For instance, if two concepts have the same support, the parent selection second criteria can be stability.

The choice of parent selection criteria for tree transformation corresponds to a classification problem. Deciding if a "Lion" is more "mammal" than it is "preying" it is not always straightforward, hence we rely on the measures that attempt to keep the context semantics when looking at the entire concept lattice. For instance, if we have more objects described by the "mammal" set which are semantically closer to "Lion" than other concepts, then it may reasonable to be chosen as its parent. To guide the user in the choice of the selection criteria, the visual variables presented in Section 4.4.3 may be used to highlight the effect a given parent selection criteria yields. Table 4.1 presents a general recommendation for the parent selection criteria that are more suitable for some analysis tasks.

Table 4.1: General recommendation for parent selection criteria.

Criteria	**Description**	**Rationale**	**Suitable for**
Stability	Measures how likely a concept is to change if some of their attributes or objects are removed.	Stable concepts are less impacted by noise and usually represent strong correlation with real world entities (e.g.: a concept that encapsulates our notion of "mammal").	Observing real world analogies
Support	Measures the frequency of the concept *itemset*.	Frequent concepts are usually generic concepts since they aggregate a larger number of objects than the specialized ones.	Frequent pattern analysis
Similarity	Represents the degree of similarity between parent and child nodes.	Concepts that share most attributes or objects should be linked together because they are similar.	Similarity analysis
Confidence	Measures how strong the implication is between a parent concept in a child concept.	Implication is one of the most intuitive interpretation of a concept lattice.	Implication analysis

4.6 Cubix: A Visual Analytics Tool for FCA

Cubix is a Visual Analytics tool that implements all proposed techniques described in this chapter. Cubix is part of a three-year project CUBIST[6] ad result of a series of participatory design with our end users. As a first step we conducted interviews with three types of use cases attempting to find patterns within their data: a company conducting market intelligence, computational biology, and a space control centre operation monitoring. All users had in common large amounts of data of which they wanted to answer mainly the following types of questions: frequent pattern detection, anomaly detection and pattern comparison. An example of frequent pattern detection is "during the first stage of a mouse embryo development what are the genes expressed together most often?". An example of an anomaly detection is "what are the sensor and telemetry logs of a space load on the International Space Station that may be related to a specific instrument malfunction?". Finally, a pattern comparison question would be "Are the jobs available in Liverpool similar to those in Manchester?" FCA allowed them to

[6] http://www.cubist-project.eu

create semantic groupings of objects and attributes based on their co-occurrence and progressively explore the lattice to look for answers to their questions.

4.6.1 System Overview

FIGURE 4.21: Cubix user interface displaying the adult data set. Its main components: 1) Toolbar; 2) Visualisation canvas; 3) Dashboard; 4) Selection & entities bar and; 5) Filter bar.

Figure 4.21 shows the layout organisation of Cubix and its main components. All panels are collapsible allowing a novice user to view only the concept lattice or, to an expert user, several tools can be displayed on the screen.

The header contains functions to open a new formal context in the CXT file format (when deployed with FCA service) and to analyse the current context. The concept lattice is shown in the canvas region. On the left, a toolbar provides options related to the concept lattice, such as alternative visualisations, size and label assignments. The search bar on top highlights the selected concepts. On the right, the dashboard displays different aspects of the conceptual structure. The selection and entities bar show the selected concepts (by search or manual selection) and attribute/object filtering, respectively. Finally, on the bottom, the filter bar allows the user to drill down on the concepts by selecting attribute-values.

4.6.1.1 Workspaces and Data Sources

Cubix allows grouping of different data such as files, databases and a data streaming API, on a single workspace to perform analysis. Each workspace provides options for filtering and merging data from different sources.

4.6.1.2 Data Scaling

In FCA, scaling of data refers to the process of discretizing data in order to generate formal concepts. Cubix automatically identifies categorical, location and date/time fields, and it allows the filtering of attributes/objects and selection of sub-contexts. It automatic identifies and scales single attributes into boolean values, e.g. "mammal" → "mammal-yes" and "mammal-no".

4.6.1.3 Concept Lattice Analytics

Cubix combines visualization techniques with data mining, allowing users to interactively analyse the conceptual data, by filtering and selecting, transforming and clustering concepts (see Section 4.4 for details). Figure 4.21 - 2 displays the selected visualization for the concept lattice (sunburst). The other visualizations available are: Hasse diagram, Matrix, Sankey, Icicle and Tree.

A filter bar (Figure 4.21 - 5) has two functions, first it allows the filtering of concepts through the visual selection of attributes; second, it displays the current conceptual distribution for each attribute.

As seen in Section 4.4.4, it is possible to perform text searches of attributes in concepts, with an auto-completion feature to help users easily search for both attributes and values in concepts. The results are dynamically highlighted as the user searches in the different concept nodes.

Based on recommendations from our users, we use the notions of support, stability and association (explained in Section 4.5.2.1) as filters, since these strategies help our users answer questions of the form most frequent pattern. But we also represent these notions visually on the lattice to enhance data understanding. For instance, the power of implications of different concepts can be rendered by edge thickness. In this way users

can be guided in understanding and choosing criteria for reducing their lattices. Using these criteria we can also extract a tree structure to simplify the lattice representation (Section 4.5).

4.6.1.4 Association Rule Analytics

Similar to the concept lattice analytics, all interface controls work analogously with association rules (Figure 4.22). The semantic dashboard adapts to the analysis of association rules, for instance, one chart displays support, stability and confidence on a *scatterplot matrix*. If an item from the chart is selected it filters the association rules to be displayed.

FIGURE 4.22: Association Rules Analytics in Cubix.

4.6.1.5 Ontology Integration

Cubix can be integrated with a RDF/OWL triple store which enables the conceptual analysis on top of SPARQL queries. The conceptual analysis of ontologies provides unique information on the semantic data, by grouping entities belonging to particular properties in a hierarchical fashion. The process starts when the user enters a SPARQL query, which is then sent to the triple store via Cubix. The results are automatically scaled to a formal context, formal concepts are then mined and visualized in the analytics view. Currently, only Sesame and Owlim are supported in Cubix.

4.6.2 Similar Tools

Over the past decade a number of FCA analysis and visualization tools have appeared. Their purpose is to generate the concept lattice of a given formal context and the corresponding association rules. ToscanaJ[7], Galicia[8] and Concept Explorer (ConExp)[9] are among the most popular ones. They can compute and visualize concept lattices but are not designed to do so for large numbers of concepts and the support for interactive analysis is limited.

Recently, OpenFCA[10], an open source FCA-based web application has drawn attention in the area for its ability to create formal contexts, mine and visualize concepts and explore association rules. It is one of the first tools with a highly interactive layout for concept analysis. There are a few limitations however, for instance, the only method for reducing large lattices is based on defining a maximum tree depth.

Specialized tools like FCA Bedrock[11] can scale data into formal contexts and has become a essential step in order to employ FCA in a Business Intelligence context. Facettice, a tool for visualizing and navigating in concept lattices demonstrated that visualization and interaction techniques can greatly enhance the understanding of conceptual data. However, the tool was not designed to support interactive analysis.

Cubix extends the features of existing tools, from data scaling to the visual exploration of the concept lattice and association rules.

4.7 Chapter Summary

In this chapter, we described techniques for the visual exploration of concept lattices through searching, filtering and sub-selection of concepts and attributes; clustering and transforming concepts, and encoding lattice's properties using visual variables. We presented novel visualisations for concept lattice and association rules. Additional information can be found in (Melo et al., 2012) and (Melo et al., 2011b).

[7] toscanaj.sourceforge.net
[8] iro.umontreal.ca/~galicia
[9] www.source-forge.net/projects/conexp
[10] www.code.google.com/p/openfca
[11] www.source-forge.net/projects/fcabedrock

In order to facilitate navigation in large concept lattices, we propose a transformation approach to extract trees from concept lattices, attempting to minimize both semantic and conceptual loss in favour of readability and interpretation. This is an important step in the visual analysis of conceptual structures, as the resulting tree structures are visually easier to understand than cluttered lattice graphs. Each of the tree construction measures proposed in our work provides particular insights valuable to different analysis tasks, identified in our work as recommendations.

These contributions represent an important step in the visual analysis of conceptual structures, as domain experts can get semantically rich insights that traditional FCA tools do not provide.

Chapter 5

Case Studies

5.1 Introduction

This chapter provide details of each of the three use cases that demonstrate the application of the proposed techniques in a real-world setting. The three use cases are: a) Aircraft cabin design (Complex Systems Design Laboratory, CSDL), where new visualisations for continuous data helped analysts to quickly identify classes of comfort for passengers; b) Genes co-expression analysis using a combination of both analytics features and semantic integration (Heriot-Watt University, HWU); and c) Prediction of possible failures from telemetry data in real-time (Space Application Services, SAS).

The first part of the chapter describes the design and evaluation methodology carried throughout the thesis. It outlines the procedures, test users and evaluation results. The second part of this chapter discusses the idiosyncrasies of each use case and presents further discussion from an user-centered perspective.

5.2 Methodology

It is widely recognized that understanding social and organisational context is critical for the success of many systems today. *Human Computer Interaction* (HCI) literature provides methods that help the system designer to collect and analyse users and their surrounding environment, and further enhancing the interface (Lazar et al., 2010). In the *User-Centered Design* (UCD) paradigm end users have an active role during all

phases of the development process, as opposed to traditional software engineering where user evaluations are usually conducted in later stages (Fitton et al., 2005).

Users are only a part of the puzzle. They carry information, perform tasks, collaborate, adapt to new circumstances, set goals, in a complex and dynamic environment. It is therefore, necessary to have a holistic approach to study users and their environment where the system can be seen as a medium in which the user will serve to accomplish his or her goals. One of the most important aspects of system modelling is the task analysis (Hackos and Redish, 1998). For this reason, users from the three use cases were encouraged to participate in the design and prototyping sessions in a task-oriented way, besides the usual evaluation at the end of the process. The methodology is described in the following sections.

5.2.1 Design and Prototyping

As a first step we conducted interviews with three types of users attempting to find patterns within their data: users conducting market intelligence, computational biology, and space control centre operation monitoring. All users had in common large amounts of data within which they wanted to answer 3 types of questions: frequent pattern detection, anomaly detection and pattern comparison. An example of frequent pattern detection is "during the first stage of a mouse embryo development what are the genes expressed together most often?". An example of anomaly detection is 'what are the sensor and telemetry logs of a space load on the International Space Station that may be related to a specific instrument malfunction". And a pattern comparison question would be "are the jobs available in Liverpool similar to those in Manchester?".

Open questionnaires were used to gather information about user's background, common tasks and goals (Appendix C. The interviews were complemented with observations in order to get insights on implicit tasks[1]. All users had previous experience with analytics systems at different levels and this aspect was taken into account.

[1]The *implicit* tasks are those that the user cannot easily express in an interview because they are not the focus of attention, e.g. checking if there are new notifications in the monitoring system.

5.2.1.1 Low-Level Tasks Taxonomy for Formal Concept Analysis

To guide our analysis, the main low-level tasks for Formal Concept Analysis were identified. These are common tasks performed by users on a concept lattice in order to match with the use cases aforementioned. This is based on works of Amar et al. (2005) and Lee et al. (2006) on graph analysis complemented by user interviews. Table 5.1 outlines the most important tasks for FCA.

TABLE 5.1: Low-Level Analytic Tasks for Formal Concept Analysis.

Concept Lattice Tasks	**Description**
Follow Path	Display concept hierarchy for a given concept
Find specialization/-generalization	Find antecedents or descendants for a given concepts
Filter	Find concepts satisfying some conditions on objects and attributes
Find Classes	Given a set of items, find the frequent co-occurrence patterns
Characterize Distribution	Given a concept and a quantitative attribute, find the distribution of the items in the concept
Cluster	Find similar concepts
Correlate Items	Determine useful relationships between concepts
Scan	Quickly review a set of items
Set Operation	Find the intersection of two or more concepts

Although concept lattice analysis share some similarities with graph analysis, they differ on a few important points. For example, graphs have much more topological importance than lattices, so finding neighbours or central nodes or clusters does not make much sense in concept lattice analysis. Concept lattice analysis in turn, has much more browsing and correlation tasks requirements. A comprehensive study on low-level task for FCA remains to be done in the literature.

5.2.1.2 Participatory Design

Once the subset of tasks the interface should act upon are identified, the first prototypes can be designed. In the *participatory design* sessions users are encouraged to create their own version of the interface and communicate its behaviour. Although this technique is not intended to literally design the interface, it provides a good understanding on how users expect the interface to behave, what are the expected inputs and outcomes. Figure

5.1 shows some low fidelity prototypes designed by users from the biological use case (HWU). Notice that Figure 5.1(c) influenced our dashboard / FCA views described in Section 4.4.2.

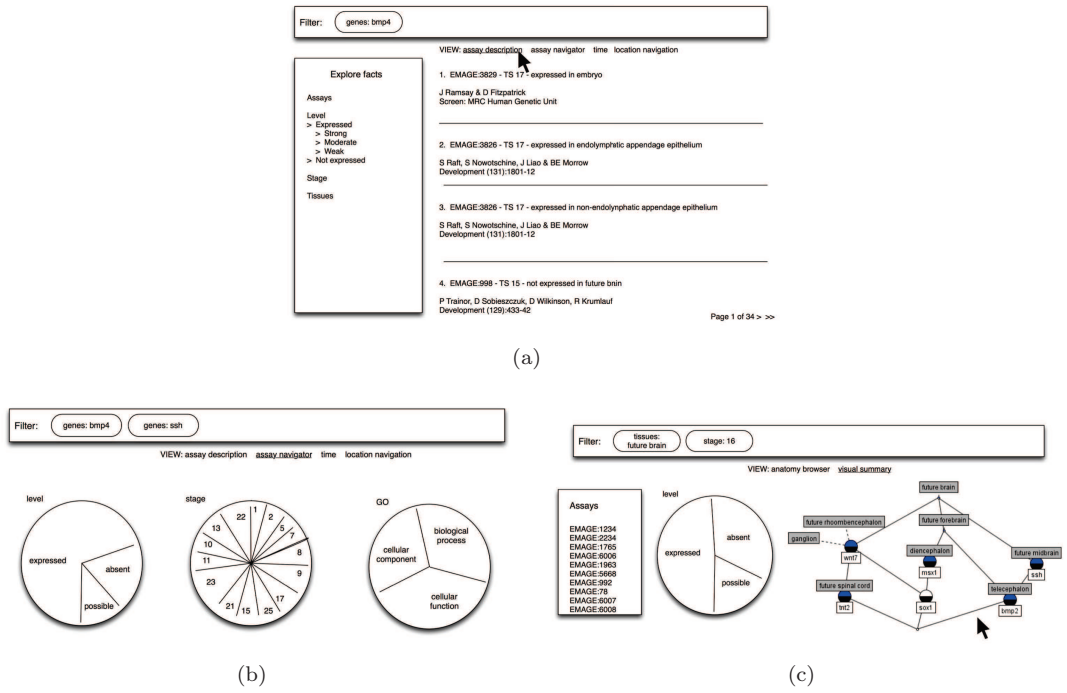


FIGURE 5.1: Prototypes designed by the users of the biological use case (HWU).

The main features were identified across the three use cases, in a way that the prototyping focused in the commonalities rather than individual aspects. The following table (Table 5.2) summarises the differences between the three use cases given five high-level features: Data browsing, lattice exploration, multiple facets, visual analytics and search.

TABLE 5.2: General recommendation for parent selection criteria.

Use Case	**Data Browsing**	**Lattice Explo-ration**	**Multiple facets**	**Visual Analytics**	**Search**
CSDL	×			×	×
HWU	×	×	×	×	×
SAS	×			×	

5.2.2 Early Evaluations

This section describes the evaluation methodology in the early stages of the development process. Sections 5.3, 5.4 and 5.5 provide an in-depth discussion on each use case with

further evaluation results.

5.2.2.1 Test Users

Operators from the Space Control Center at SAS, biologists and bio-informaticians at Heriot-Watt University took part in the evaluation. These two populations have different background experience with analytic systems as operators are far more experienced than both biologists and bio-informaticians. This poses a challenge to the design: How can the interface be simple enough so that less experienced users make good use of it and at the same time allowing experienced users to use its advanced functions? The solution we propose, as we will see later in this chapter, consists on a set of configurable views for each use case and user skill.

5.2.2.2 Procedure

In an evaluation, data is gathered from users which reflects their judgement of their usage of the evaluated product (Hackos and Redish, 1998). The first evaluation was carried with the aid of a leader user for each use case to conduct the evaluation in place. Before the evaluation, we provided some material about Cubix, including a short tutorial describing its main functionalities.

To evaluate the features of our first version of the prototype, we have used a sample of a traffic accident dataset[2] and conduced an analysis for most common causes between different combinations of attributes (e.g. road surface, weather, accident severity, etc). FCA allowed us to answer pattern identification questions such as "What can be considered the main causes of accident severity: serious'?" or "how many accidents are 'light conditions: darkness' and 'accident severity: serious'?".

The evaluation consisted of three parts: 1) A leader user from each use case provided three tasks that should be performed using Cubix with the dataset provided; 2) The users performed the assigned tasks while communicating their thought process[3] as we record audio and screen capture; 3) A survey (Appendix D) is collected from users and conduct an interview with the leader user.

[2]Traffic accidents dataset released by the Department for Transport: http://data.gov.uk/dataset/road-accidents-safety-data

[3]Also known as the "Think Aloud" method (Fonteyn et al., 1993)

Some examples of tasks are:

- How many accidents are “light conditions: darkness” and “accident severity: serious” ? Do you think this should be another way of doing this task? How?
- What can be considered the main causes of accident severity: serious”?
- How likely is an and “accident severity: serious” occur during weekends (Sat-Sun)?
- What about the surface conditions of the fatal accidents?
- What the probability of causing serious or fatal accidents when it snows?
- what are the differences between accidents on Saturday or Sunday?

The choice of a common, unfamiliar dataset for the evaluation was intended to avoid domain-specific bias when performing the tasks in the prototypes.

5.2.2.3 Evaluation Results

Below, we outline the results of the quantitative evaluation for each component.

For the “Search and Select” Component	
The purpose and function of the component is clear.	0.75
The component is easy to understand and use.	0.58
The interface is appealing and attractive.	0.42
The component is useful.	0.83
For some kinds of information needs or queries, particularly this component is useful.	0.75
I have similar functionalities in the tools I usually use.	-0.11

FIGURE 5.2: UI Component Questions.

For the „Navigate in Data”-Component	
The purpose and function of the component is clear.	0.33
The component is easy to understand and use.	0.33
The interface is appealing and attractive.	0.50
The component is useful.	0.33
For some kinds of information needs or queries, particularly this component is useful.	0.33
I have similar functionalities in the tools I usually use.	0.67

FIGURE 5.3: UI Component Questions.

For the “Explore Selection” Component	
The purpose and function of the component is clear.	0.67
The component is easy to understand and use.	0.67
The interface is appealing and attractive.	0.67
The component is useful.	0.67
For some kinds of information needs or queries, particularly this component is useful.	0.67
I have similar functionalities in the tools I usually use.	-1.00

FIGURE 5.4: UI Component Questions.

The detailed evaluation for each use case is described in the following sections.

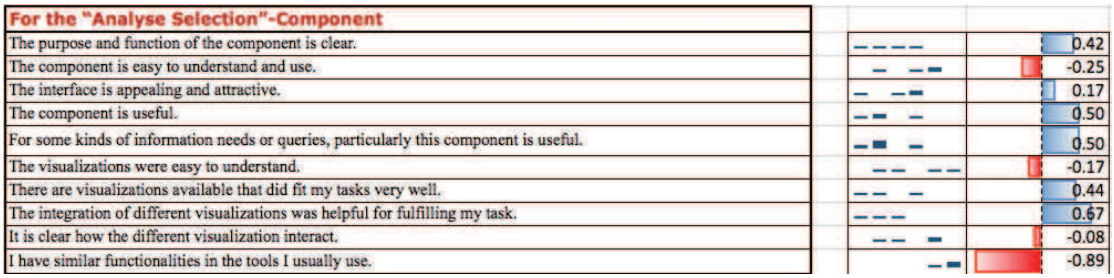


FIGURE 5.5: UI Component Questions.

5.3 Case Study: Genes Expression Analysis Enhanced by Visual Analytics

A gene is a unit of instructions that directs the body how to do one essential task, i.e. create a protein. Gene expression information describes whether or not a gene is expressed (active) in a location. There are many types of gene expression experiment. This case study focused on a technology called *in situ* hybridisation (ISH) gene expression. Completed ISH experiments are published online. For the mouse, one of the main resources in this field is EMAGE[4].

EMAGE documents the result of an experiment using a series of textual annotations. Each annotation is a triple: *gene tissue-level* of expression. For the sake of brevity, both genes are referred to by short names or identifiers rather than their full name. For example, the gene *bone morphogenetic protein 4* is referred to as *Bmp4*. Tissues are often present in several *Theiler Stages* and there can be multiple instances of a tissue in a single Theiler Stage. Thus each TS has its own anatomy called EMAP and each tissue is uniquely identified by an *EMAP* number. For example the *orbito-sphenoid bone* in *TS 23* is *EMAP:8385*.

A number of computational methods have been proposed to help biologists discover unexpected patterns and formulate interesting hypotheses. Popular techniques employ *unsupervised classification* methods such as clustering, to group and visualise co-expressed genes (see Prelić et al. (2006) for a survey). The main limitation of most clustering algorithms is that they do not allow clusters to overlap, a counter-intuitive idea in this domain as genes are not restricted to a specific function and usually take part in several biological processes when interacting with other genes.

[4]http://www.emouseatals.org/emage/

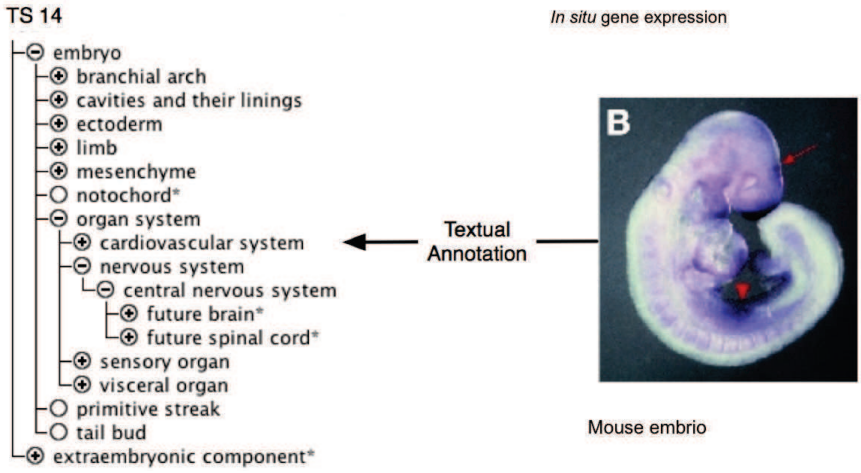


FIGURE 5.6: An Illustration of an Annotated Schema for *In situ* Gene Expression Data in *Theiler Stage* 14.

One of the main motivations for the use of FCA in life sciences comes from the idea that FCA provides an intuitive understanding of generalisation and specialisation relationships among objects and their attributes. In (Kaytoue-Uberall et al., 2008) for example, FCA was used to extract groups of genes with similar expressions profiles from data of the fungus *Laccaria bicolor*. In (Blachon et al., 2007) authors developed a tool to query a set of extracted formal concepts in a human gene expression data set according to various criteria (e.g. presence of a keyword in a gene description) and then to cluster concepts according to similarity, in terms of the attributes (samples) and the objects (genes above a threshold of expression) constituting the concepts. They called these clusters *quasi-synexpression-groups* (QSGs).

In (Andrews and M^{c}Leod, 2011) authors presented an approach to use FCA to analyse large clusters of gene co-expressions. The approach makes use of the formal context creator FcaBedrock[5] and the formal concept miner In-Close[6] (the aforementioned tools have been developed in Cubix) to convert and simplify formal contexts. The workflow is explained as follows. The user supplies metadata for conversion in FcaBedrock, such as the names of the genes or tissues and their values, and with decisions as to what to convert and how to convert it. These metadata are used to create a formal context. User-defined constraints, such as object exclusion, attribute exclusion and attribute restriction, applied to the data, allow different analyses to be carried out and the creation of sub-contexts which only focus on particular portions of the data. Next, In-Close

[5]http://sourceforge.net/projects/fcabedrock
[6]http://sourceforge.net/projects/inclose

is applied to the context file generated in the previous step. Using a trial and error approach, the user has to find an appropriate minimum size to mine a small number of large concepts, i.e., find the largest co-expressions of genes within the data. Finally, a third tool, Concept Explorer[7] was used to visually display the corresponding concept lattice.

Whilst the above workflow required three standalone tools and FCA expertise, the same workflow can be achieved through a combination of different techniques provided by Cubix. Most of the complexity has been hidden, empowering the biologists to run the entire workflow themselves. Additionally, whilst FCA could only be visualised via a static lattice, Cubix provides a series of analytical features and is able to deal with the implicit relationships and inconsistencies in the EMAGE data.

In this case study we investigated how Cubix can address the challenges of gene expression analysis, through filtering and clustering of large amounts of data, interactive exploration of co-expressed genes and display of relevant statistics.

5.3.1 Querying and Converting the Gene Expression Ontology Data to Formal Contexts

In contrast with traditional FCA, which takes as input a binary table of objects and attributes, our approach is based on the querying of ontology data which is then converted to a formal context in a process transparent to the user. The conceptual analysis of ontologies provides unique information of gene expression data, by grouping entities belonging to particular properties in a hierarchical fashion and highlighting patterns of co-occurrence for those groups. Because biologists have little or no knowledge of SPARQL, the language we use to query ontologies, a set of pre-defined queries are available.

The procedure consists in translating each object o, attribute a and their incidence relation I in the result table such as $o \in G$ and $a \in M$, to create the formal context $\mathbb{K} = (G, M, I)$. As the number of variables in a query can be arbitrary, Cubix has an option that allows users to select whether a given column in the result table is an object column, attribute or none. Contrarily to traditional FCA tools, in Cubix the formal

[7] http://sourceforge.net/projects/conexp

context is transparent to the user. The data is displayed in a table where its rows contain attributes, followed by attribute values and objects, in the columns (Figure 5.7). We found that this structure is more accessible to non-FCA experts. Filtering, sub-selection and conversion operations are possible through functionalities that came from the aforementioned tool FcaBedrock.

	Attribute	Values	Objects
☑	embryo	level_detected , level_not_detected , level_weak ,	Arc, Foxa2, Smad2, Otx2, Smad5, Smad1
☑	endoderm	level_detected , level_not_detected ,	Foxa2, Zic2, Zic3
☑	definitive endoderm	level_detected ,	Hhex, Cer1
☑	parietal endoderm	level_strong ,	Fst
☑	extraembryonic ectoderm	level_detected ,	Bmp4, Zic2
☑	ectoderm	level_detected ,	Wnt3, Smad5, Otx2, Eomes, Zic2, Tdgf1, Zic3, T
☑	mesoderm	level_detected ,	Wnt3, Smad5, Lefty2, Eomes, T, Mesp1, Tdgf1
☑	primitive streak	level_detected , level_strong , level_not_detected ,	Lefty2, Eomes, Gsc, Foxa2, T, Fgf8, Mesp1, Fst, Smad1, Otx2
☑	primitive endoderm	level_detected , level_not_detected ,	Hhex, Wnt3, Fgf8, Smad2, Otx2, Gsc, Foxa2, Cer1, Sox17
☑	visceral endoderm	level_strong , level_detected ,	Smad1, Foxa2, Sox17
☑	extraembryonic component	level_weak , level_strong , level_detected ,	Smad5, Bmp4, Arc, Smad2

FIGURE 5.7: Genes, tissues and level of expression in Theiler Stage 9.

The formal context, once created, is passed to the concept miner, which returns the number of formal concepts to the user. If the number of formal concepts is too high, the user can exclude from the computation concepts with fewer than a user-specified number of attributes and objects (so-called *minimum support*), to simplify the context further. Apart from the user being able to manually define minimum-support criteria, in Cubix, the minimum-support feature is being reconfigured to be automatically calculated and applied to the formal context without user intervention.

5.3.2 Visual Analysis of Expression Clusters

Figure 5.8 visualises a tree as a Sunburst (Stasko, 2000) extracted from the concept lattice using support as parent selection criteria. The context in Figure 5.7 was filtered to focus on gene *Bpm4*. In this visualisation, each concept is an arc and the hierarchy is represented from the innermost to the outermost layer.

Clustering of concepts (as opposed to clustering of objects or attributes) can be useful to facilitate the browsing of concepts and to identify zones of interest in the gene expression data. In our experiment (Figure 5.8), biologists used Cubix's *K-means* clustering algorithm to identify 5 clusters of gene expression information. The zones identified can be roughly described as follows. The blue cluster represents the concepts that occurrences of the gene *Bmp4* in similar tissues. The orange cluster contains the concepts related to the "Theiler_Stage-20", whereas in the red cluster we find the concepts comprising both "Theiler_Stage-18" and "Theiler_Stage-19".

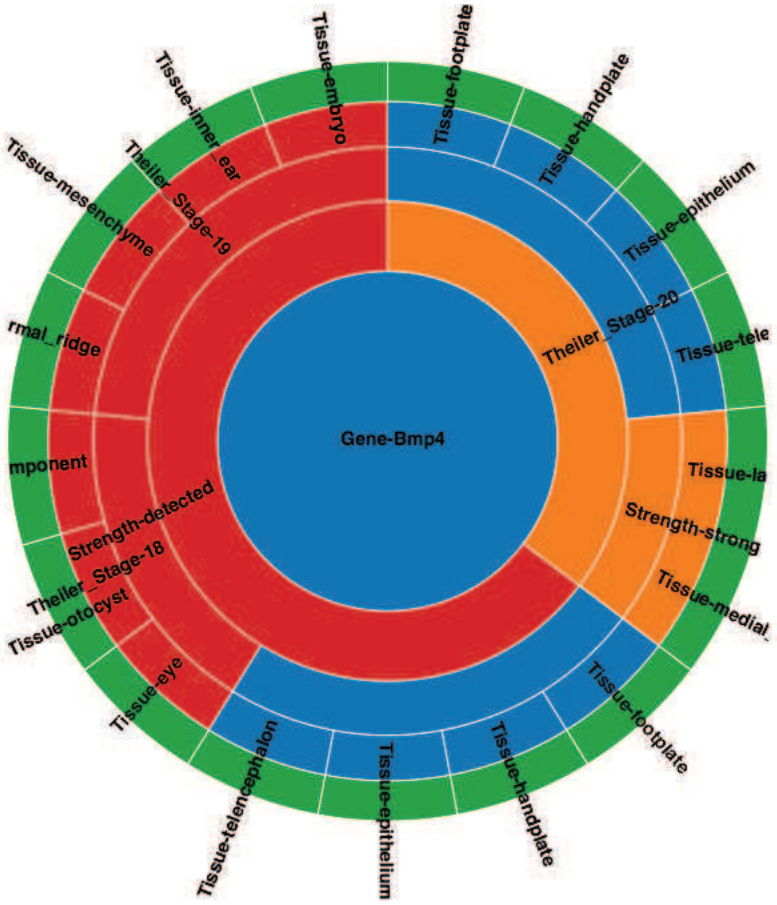


FIGURE 5.8: A radial-filling "Sunburst" visualisation of the gene expression data with colours depicting clusters.

Cubix has a dashboard where several charts display different aspects of the underlying conceptual structure such as co-occurrence of attributes, concepts distribution, stability vs. support, etc. Interestingly, by looking at the "attribute implication" chart, biologist were able to see how properties are related to gene Bmp4, in this case, "strength-detected" and "Theiler_Stage-20" are highly related to the occurrence of this gene (Figure 5.9).

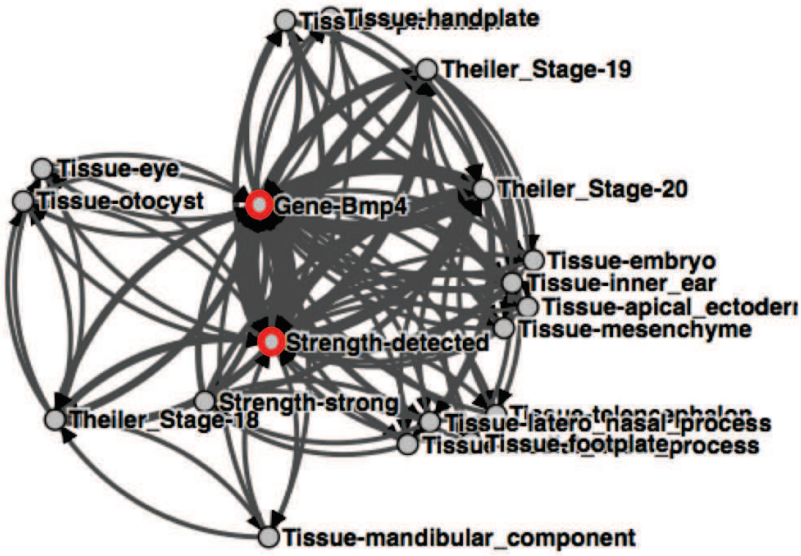


FIGURE 5.9: A radial-filling "Sunburst" visualisation of the gene expression data with colours depicting clusters.

5.3.3 Using Association Rules to Highlight Gene Expression Occurrence Patterns

The use of association rules analytics brought complementary insights to the analysis of gene co-expression clusters. For example, after filtering the rules generated for the context in Figure 5.7 by confidence down to a manageable size, results revealed a few interesting facts. A significant amount (75%) of the genes detected in the embryo were also detected in the primitive *endoderm* (Figure 5.10) in *TS9*. This is not surprising, since the later tissue is part of the former in the anatomy hierarchy. On the other hand, another rule showed that 71% of the genes detected in the *mesoderm* were detected also in the *ectoderm* (against 62% the other way around) in *TS9*. This follows an intuitive reasoning since both tissues are part of the same organ.

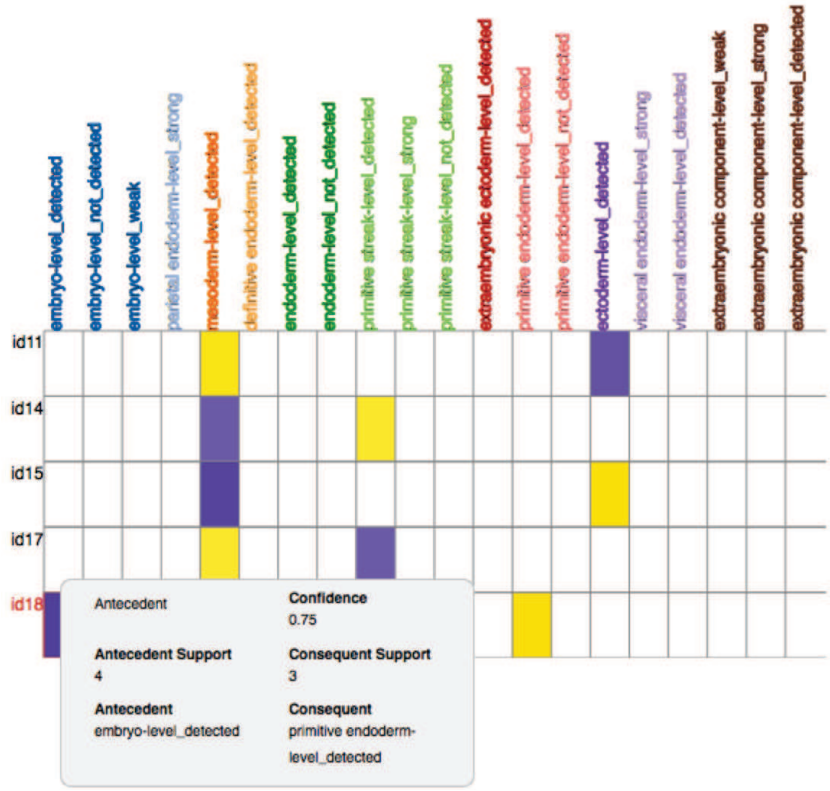


FIGURE 5.10: Genes, tissues and level of expression in Theiler Stage 9.

5.3.4 Discussion

This use case demonstrated the use of Cubix to the analysis of gene expression data where overlapping groupings are generated by Formal Concept Analysis and interactively analysed in Cubix. Cubix's workflow allows users to carry out an analysis starting from querying a semantic database, converting it into a formal context, simplifying the context to make it manageable, and visualising the result as a concept lattice and associated relevant statistics.

Existing tools for genes expression analysis, such as Cytoscape[8] and Orange[9] have specific advantages, e.g. Cytoscape can run efficient analysis in network data and Orange is a machine learning tool with some predictive features. In contrast, Cubix operates at a conceptual level and it is less data-mining centric and more analytics oriented (i.e, dashboards, drill-down, selection and filtering, etc). Besides, those tools are designed to run locally whereas Cubix is being designed to run on a cluster of computers eventually

[8]http://www.cytoscape.org
[9]http://orange.biolab.si

in the cloud. This will allow scalable analysis of data of orders of magnitude higher than the mentioned tools.

Although most of the functionalities in Cubix can be used with other data than EMAGE (with the corresponding scaling of data), as future work we will extend our experiments to other genes expression data sets like cancer and brain development. We also intend to provide a public web service API to allow interoperability with other platforms.

For further information about this use case see (Melo et al., 2013).

5.4 Case Study: A Visual Analytics Approach for Aircraft Cabin Design

This case study is part of the Complex System Design Lab (CSDL)[10] project which involves 27 industrial and academic partners and aims at providing a collaborative environment for complex system design. Since the simulation of design choices usually outputs large and high dimensional datasets, the CSDL platform should allow efficient analysis of such datasets to identify the right conception choices.

CSDL industrial partners have provided a use case which corresponds to a commercial aircraft cabin air control system. In this use case, the goal is to identify relevant design configurations which ensure comfort conditions in terms of *air temperature* and *velocity* inside the cabin. Typical fields of temperature and velocity are obtained using the same fluid model as in (Bui et al.) and the comfort design problem is parametrized by 13 continuous parameters each evolving in a range interval of possible values.

These design parameters are: angles of air injection at 4 passengers' personal fan (Alpha_1..4), blown air speed at 4 passengers' personal fan (Uair_1..4), temperature of blown air at main inlet (Tair_In), temperature of blown air at 4 passengers' personal fan (Tair_P), blown air speed at main inlet (Uair_In), external temperature (T_ext), and fuselage thermal conductivity (Kappa_F). The mean values of temperature and velocity for each of the four passengers' seats (see Figure 5.11) have been computed to assess the passengers' comfort, which resulted in eight output criteria (two per passenger) related

[10] http://www.systematic-paris-region.org/fr/projets/csdl

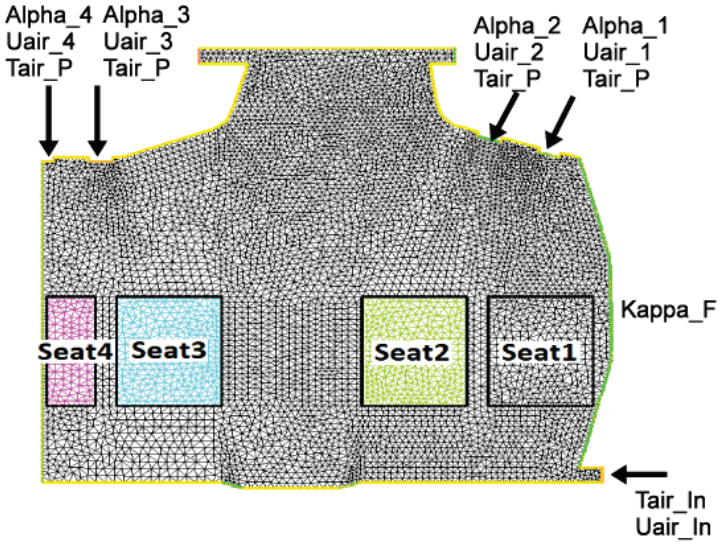


FIGURE 5.11: Aircraft cabin schema and the main simulated input parameters.

to the comfort. Moreover, a measure of the energy consumed by the air-conditioning system is also considered to estimate the price at which this comfort comes.

As we discussed previously, in FCA, complex data needs to be transformed into a boolean matrix in the scaling process. In order to provide flexible scaling we used a concept miner algorithm based on Similarity Formal Concept Analysis (SFCA) introduced by Messai et al. (2008). SFCA considers similarity to directly classify non-binary data into lattice structures called Many-Valued Concept Lattices (MV lattices) using a similarity threshold θ for each multi-valued attribute.

Besides extending FCA to complex data and avoiding the loss of information due to truncation in FCA scaling, SFCA produces MV lattices with different granularity levels (i.e. *similarity* levels) to allow progressive data exploration.

In this use case we investigated the usefulness of Cubix to provide a support for combining numerical values (quantitative) analysis together with qualitative analysis to aid decision makers in the process of complex system design. In particular, we show how Cubix can be used to highlight crucial information a designer may need to validate design choices. The analysis is facilitated by a novel visualization for MV concepts.

5.4.1 Simulation Dataset of the Aircraft Cabin Test Case

The use case dataset corresponds to the simulation results of 100 randomly chosen configurations of design parameters (the 13 input parameters). 9 output criteria have been defined to assess the quality of each configuration in terms of passengers comfort and energy cost. The mean values of temperature and velocity of each of the four seats are computed which resulted in 8 criteria associated with the comfort of the passengers. The dissipated energy is computed based on the velocity as a measure of the loss of energy due to the fluid viscosity.

In order to quickly appreciate the comfort in each seat and hence simplify the dataset exploration and the experiments evaluation, comfort scores were computed for the values of the comfort output criteria (temperature and velocity). The scores are in a three points scale (0: uncomfortable, 1: acceptable, 2: comfortable) computed according to ANSI/ASHRAE Standards (ASHRAE, 2004) as follows:

$$\text{score(T)} = \begin{cases} 0 \text{ if } T < 21 \text{ or } T > 24 \\ 1 \text{ if } 21 \leq T < 22.5 \text{ or } 23.5 < T \leq 24 \\ 2 \text{ if } 22.5 \leq T \leq 23.5 \end{cases} \quad \text{score(V)} = \begin{cases} 0 \text{ if } V > 1 \\ 1 \text{ if } 0.2 < V \leq 1 \\ 2 \text{ if } V \leq 0.2 \end{cases}$$

These rules can be translated as expressions for a given score in Cubix (Figure 5.12). The score is then computed for each concept in the lattice and it may be used for filtering or assigning colours according to the resulting value. If two or more expressions are satisfied for a given concept, the last added rule has the priority.

5.4.2 Extracting Comfort Classes and Their Corresponding Design Parameters' Ranges

Our objective is to determine the design parameters that are important to qualify the experiments such that all of the four passengers seats are satisfied. We make the assumption that the temperature is more important than the velocity to define the thermal comfort of the passengers. Therefore, we focus our analysis on the experiments that offer the maximum comfort for the passengers from the temperature point of view only: we

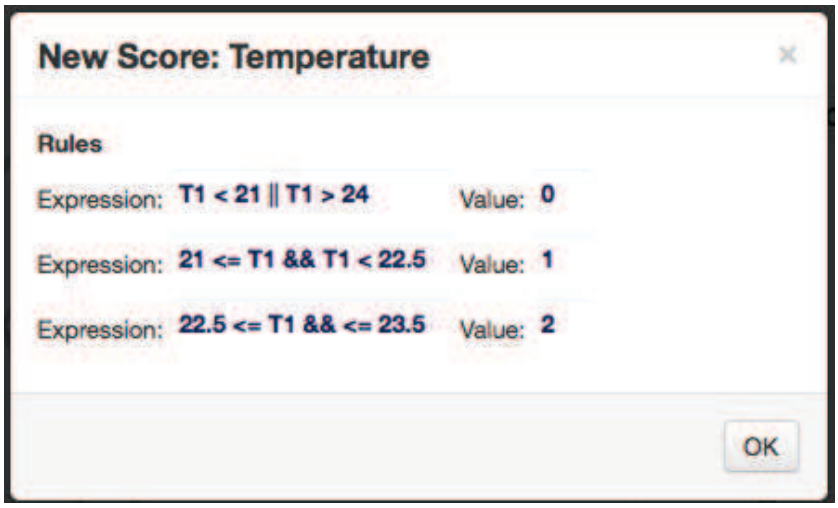


FIGURE 5.12: Creating conceptual scores in Cubix.

keep only the concepts such that the score for the temperature is 2 and we called this subset S_{silver}.

Then, following a strategy for choosing the similarity thresholds as explained in (Messai et al., 2012), we applied SFCA to build the corresponding MV lattice shown on Figure 5.13. Notice that colours were assigned according to the score we defined in Figure 5.12.

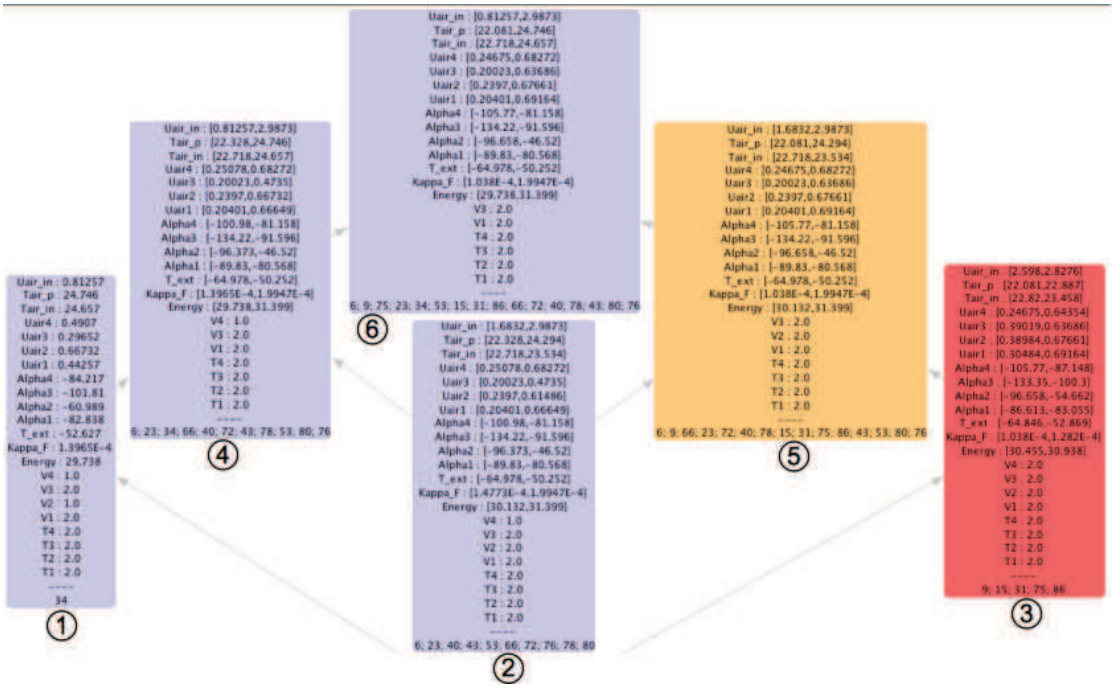


FIGURE 5.13: MV lattice generated on S_{silver}.

By analysing this MV lattice, it turns out that S_{silver} can be described using three

TABLE 5.3: Extracted ranges of the 13 comfort design problem parameters for different classes of comfort.

Parameter	Range	"Maximum comfort"	"Intermediate comfort"	"Poor comfort"
Alpha_1 (°)	[-90, -80]	[-86.61, -83.05]	[-89.83, -80.56]	[-89.83, -80.56]
Alpha_2 (°)	[-100, -45]	[-96.65, -54.66]	[-96.65, -46.52]	[-96.65, -46.52]
Alpha_3 (°)	[-135, -90]	[-133.35, -100.3]	[-134.22, -91.59]	[-134.22, -91.59]
Alpha_4 (°)	[-120, -80]	[-105.77, -87.14]	[-105.77, -81.15]	[-105.77, -81.15]
Uair_1 (m/s)	[0.2, 0.7]	[0.30, 0.69]	[0.20, 0.69]	[0.20, 0.69]
Uair_2 (m/s)	[0.2, 0.7]	[0.38, 0.67]	[0.23, 0.67]	[0.23, 0.67]
Uair_3 (m/s)	[0.2, 0.7]	[0.39, 0.63]	[0.20, 0.63]	[0.20, 0.63]
Uair_4 (m/s)	[0.2, 0.7]	[0.24, 0.64]	[0.24, 0.68]	[0.24, 0.68]
Uair_In (m/s)	[0.6, 3]	[2.59, 2.82]	[1.68, 2.98]	[0.81, 2.98]
Tair_In $(°C)$	[22, 25]	[22.82, 23.45]	[22.71, 23.53]	[22.71, 24.65]
Tair_P $(°C)$	[22, 25]	[22.08, 22.88]	[22.08, 24.29]	[22.08, 24.74]
T_ext $(°C)$	[-65, -50]	[-64.84, -52.86]	[-64.97, -50.25]	[-64.97, -50.25]
Kappa_F (W/K)	$[1e^{-4}, 4e^{-4}]$	$[1.03e^{-4}, 1.28e^{-4}]$	$[1.03e^{-4}, 1.99e^{-4}]$	$[1.03e^{-4}, 1.99e^{-4}]$

distinct main comfort classes: (i) "*Maximum comfort*" (concept $n°3$): maximum score for both temperature and velocity for the 4 seats, (ii) "*Intermediate comfort*" (concepts $n°2$ and 5): the score for velocity for seat 4 is not maximum, and (iii) "*Poor comfort*" concept (concept $n°2$, 4, and 6): the score for velocity for seats 2 and 4 is not maximum.

These three comfort classes can be directly read on Figure 5.13. Each class is given by one MV concept in the lattice and the order defined on the MV concepts also holds for the comfort classes : Concepts 3, 5 and 6 corresponding respectively to maximum, intermediate, and poor comfort classes. The colors linked to concept scores in Figures 5.13 and 5.14 allow to quickly identify the concepts corresponding to the most relevant design configurations and to extract ranges of variation of the corresponding design parameters. These extracted ranges are given in Table 5.3.

5.4.3 Identifying Comfort Classes Using the "Heat map" Visualization

When the relevant concepts are identified, i.e. the comfort classes in our case, the analyst will take decisions based on concepts that fit best his or her goals.

The filtering capabilities provided by Cubix helped the analytics to boil down the large number of concepts to a handful of concepts. The task of the analyst is then to identify which concepts contains the parameters within the intervals he or she is looking for. For example, for the parameters of *air speed* and *temperature*, large ranges mean that comfort may be compromised. On the other hand, smaller ranges indicate stronger guarantees on comfort. The heat map visualisation described in Section 4.2.2 has been useful in this context. Figure 5.14 shows the concepts as an array of square rectangles where each rectangle represents an attribute, its width is proportional to the size of the interval and the colour indicates the average (median) value of the interval, varying from blue to red.

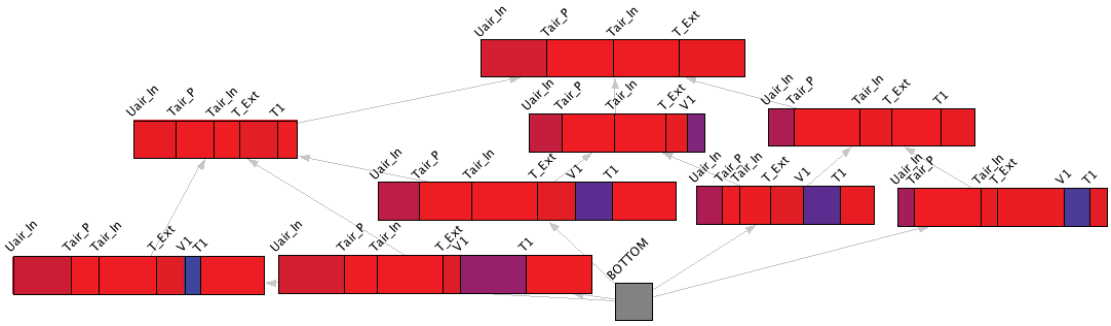


FIGURE 5.14: The MV concept view for the lattice in Figure 2. Colour indicates position in the range (from blue to red), width shows the length of the interval.

This representation allows to straightforwardly read the maximal ranges of variation of all the design parameters together with the corresponding comfort score for temperature and air speed in each concept.

5.4.4 Discussion

In this use case we presented a visual and conceptual approach for decision support applied in a collaborative complex system design project. The approach takes advantage of the use of Similarity-based Formal Concept Analysis (SFCA) to classify, visualize, and explore simulation data in order to help system designers to identify relevant design choices.

The approach is illustrated on an aircraft cabin design case study which concerns the simulation of different configurations of the ventilation system to study the passengers comfort in the cabin. The classification of simulation data with their corresponding comfort scores using Cubix allows to derive for each simulated input parameter the maximal interval of values which guarantee an acceptable comfort level.

We implemented a new visualisation for MV lattices that consists on the filtering and assigning colours to concepts based on user-defined scores. A colour gradient is assigned according to the "global score" of a concept. For instance, the score of "maximum comfort" is attributed when the scores of velocity and and temperature are equal to 2 and the colour. This simplification is important because in Complex System Design the number of parameters are usually overwhelming to the analyst to overview and compare. Alternatively, users can filter concepts that are below a score threshold. This is particularly important during the extraction of comfort classes.

The obtained results have been evaluated by new simulations which converged to the same solutions in terms of passengers comfort as well as in terms of input parameters ranges. Details of this study can be found in (Messai et al., 2012).

5.5 Case Study: Real-Time Concept Analysis for Anomaly Detection in Space Telemetry Data

Three instruments make up the solar observatory (SOLAR, Figure 5.15) payload in the International Space Station (ISS). These instruments, SOVIM, SOLSPEC and SOLACES, measure with a remarkable degree of accuracy the spectral energy irradiance of the Sun, over virtually the entire spectrum. These studies are of primary importance for atmospheric modelling, atmospheric chemistry and climatology. Over the last three years, the SOLAR science teams have gathered precious data during the minimum of solar activity. It provided the most comprehensive measurements available about the solar minima of 2008-2010 and at the time this thesis is written, it has been recording information about the solar maxima of 2012-2013.

It may happen that the payload manifests an unforeseen thermal situation. For example, when the temperature of one or several sensors changes in an unusual way, albeit within

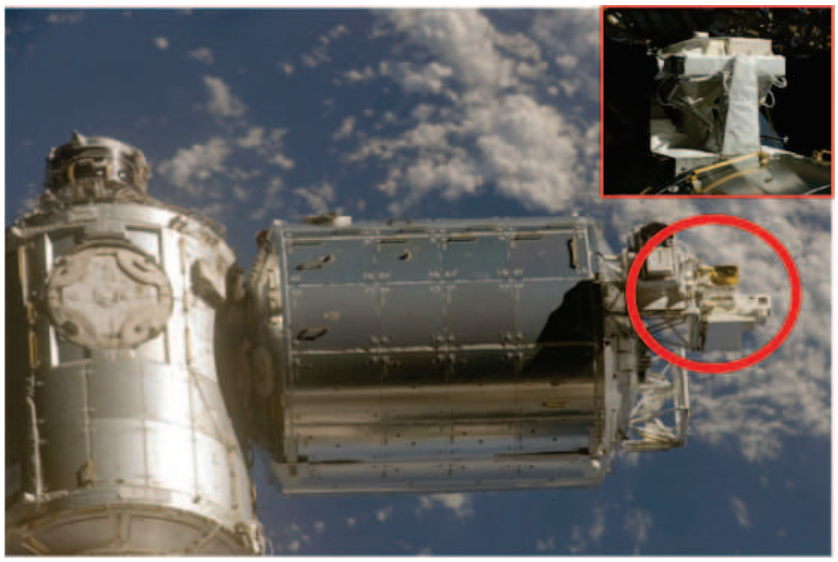


FIGURE 5.15: Solar observatory device (SOLAR) attached to the International Space Station.

the nominal limits. The operator is then charged with finding similar occurrences inside the telemetry archive and with the determination of typical thermal and power profiles. Analysing such a large number parameters is a tough task even to the most experienced operator. Figure 5.16 shows the user interface of the SOLAR monitoring system.

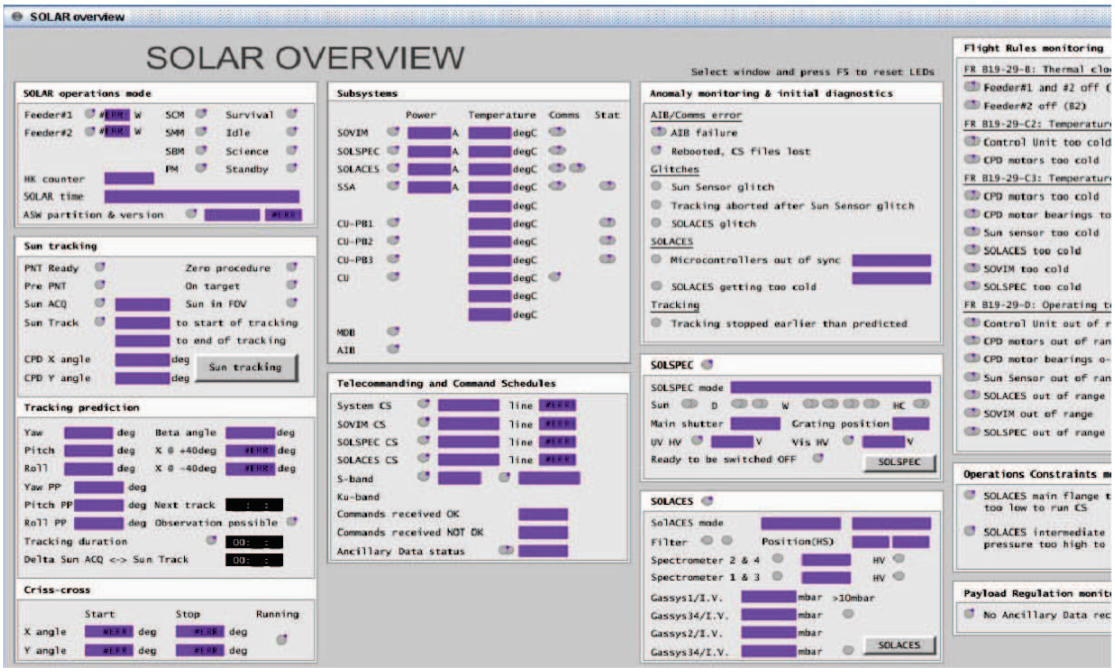


FIGURE 5.16: SOLAR monitoring system.

Whenever there is an anomaly, it is important for the operators to quickly retrieve information about the telecommands sent during a specific time period and the results

of those commands. Other examples of unexpected behaviour include:

- A telemetry parameter was out of its predefined soft values;
- An error code was issued by SOLAR;

A clear need is to find patterns of failure in the flow of telemetry parameters with the aim to transpose these to the prediction of future failures (forensic analysis). This is a common problem in many stream mining systems such as aeronautics, military surveillance, biochemical analysis and fraud analysis.

Several anomaly detection techniques have been proposed in literature. Some are based on machine learning algorithms, such as neural networks (Zhang et al., 2001), distance-based algorithms to detect outliers (Lazarevic et al., 2003) or adapted clustering algorithms (Wu and Zhang, 2006) (See (Ahmed et al., 2007) for a review).

In this use case, we apply Formal Concept Analysis to retrieve the combination of parameters that indicates an anomalous behaviour. The analysis of all possible combinations of parameters is boiled down to a handful of formal concepts.

5.5.1 Telemetry Dataset

SOLAR has been operational for more than four years, already, sending one telemetry packet every second or so. Over a year, this represents approximately 3.10e7 packets. Each telemetry packet contains 343 parameters. 44 parameters do not change at all or very rarely. Among the others, 135 have binary readings, such as ON and OFF. Others are continuous valued attributes. Example of parameters include: temperature, power supply, pointing device telemetry and system variables for the different parts of the device.

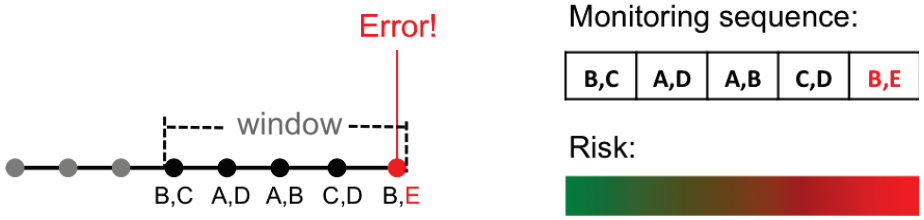
Telemetry data from the SOLAR payload is managed by Space Applications Services (SAS). A subset of the data, covering the 30-day period (26-Sep- 2008 / 25-Oct-2008) related to the particular SOVIM failure, has been provided to the CUBIST project.

5.5.2 Anomaly Detection with Real-Time Distributed Computation of Concepts

Our approach consists in simulating the telemetry stream to identify which pairs of parameters-values are related to an anomaly. Anomalies are characterized by a particular attribute in the transaction. We also investigate the sequence of concepts that preceded a anomaly, in an attempt to predict new anomalies. Because anomalies occur very rarely, we filter concepts according to *maximum* support rather than minimum support.

The anomaly detection mechanism works as follows. We applied the approach described in Chapter 3 to mine formal concepts in the current sliding window.

Whenever an anomaly occurs, all current concepts in the window are stored in a monitor data structure (a linked list) in the sequence they appeared prior to the error. Then, whenever a new transaction appears in the window, the algorithm checks for the occurrences in the monitoring table *sequentially*. The assumption is that if an anomaly occurs, looking at the sequence of previous concepts may be a good indicator of when the next anomaly may occur. Figures 5.17(a) and 5.17(b) illustrate the process.



(a) A transaction with a error (A,E) enters in the window.

(b) All concepts preceding the anomaly are monitored.

FIGURE 5.17: Monitoring concepts in the current window prior to an anomaly.

Concepts {B,C}, {A,D}, {A,B}, {C,D}, in this order, are currently in the window when a transaction {B,E} arrives (E is a flag to indicate error, Figure 5.17(a)). All concepts in the current window are then stored in the monitor list (Figure 5.17(b)). New transactions in the stream are matched with those in the monitoring table. If a sequence of transactions in the stream matches with any subsequence of concepts in the monitoring table, a signal is send to the interface to indicate increase in the risk of a new anomaly. Otherwise, if the order of concepts arriving in the stream does not match with the one in the monitoring list, the risk signal set to 0.

Experiments showed that the real-time analysis of telemetry data can be useful to restrict the number of parameters that need to be analysed when an anomaly happens. In one experiment, operators discovered that variation in voltage in two connected circuits may lead to an increase in the temperature in another device, which eventually causes it to overheat and trigger an error.

We implemented a user interface to show the current concepts in the window, the stored sequence of concepts when the last anomaly happened and the risk status (Figure 5.18). As explained before, if any upcoming transaction matches with the sequence, the risk signal is incremented and displayed in form of a *gauge*.

In this first version of the UI, operators had difficulty in tracking every change in the concepts within the current window, as concepts may change too frequently[11]. As future work, we plan to use a *time-sensitive* sliding window that updates the window every hour, so operators can carefully investigate each concept in the current window.

We are currently investigating the *minimum* number of attributes that are most likely to cause an anomaly. In the experiments, concepts related to an anomaly had between 50 and 200 pairs parameter-value. Heuristically, operators can discard most of the a attributes. For instance, "SOLAR_PB1_Ovtemp" is a flag to indicate if the temperature of the internal PB1 device is above normal conditions. It is therefore expected that the parameter "SOLAR_PB1_temp" used to indicate PB1's temperature, displays a high temperature value. Reducing the number of attributes that are not involved in the occurrence of an anomaly can greatly improve operators' productivity in discovering the causes of it.

5.5.3 Discussion

Our real-time approach helped operators to identify the combination of parameters that could potentially cause a breakdown. In most cases there is no explicit correlation between the parameters and the error; it is assumed that the co-occurrence of certain pairs of parameters-values is most likely to cause an anomaly. Operators can use Cubix for further analysis on static telemetry data, once the monitoring parameters are known.

[11] In some periods many concepts were created and removed within a few seconds

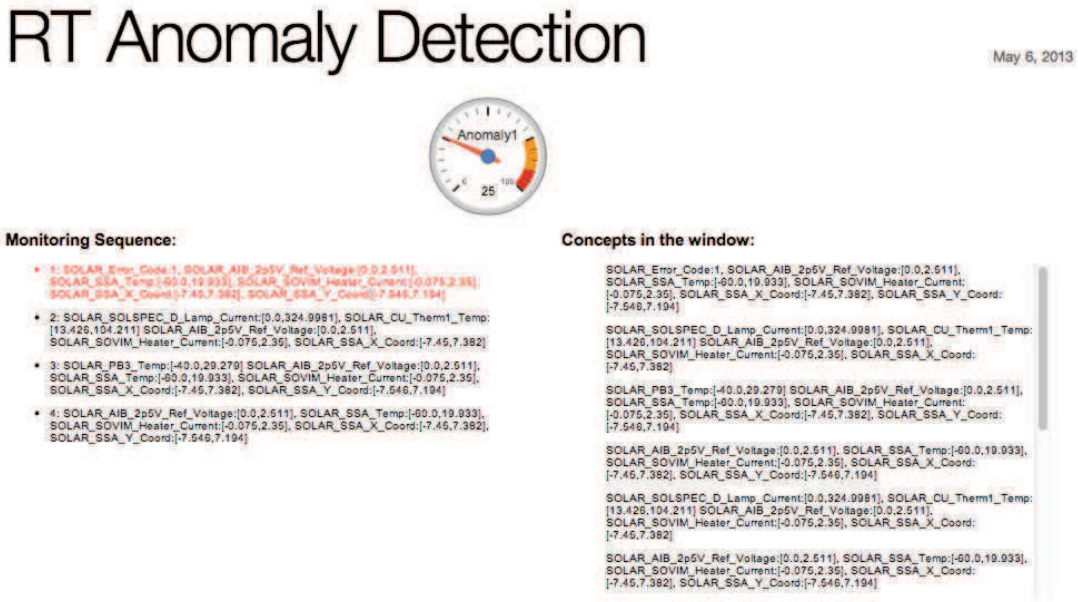


FIGURE 5.18: Anomaly Detection UI.

Although it is hard to claim that the sequence insight worked due to the limited amount of data available for the experiments, the preliminary results are encouraging. It provided insights on how some anomalies are related to a sequence of concepts, e.g. the variation off the voltage of a particular device may lead to its malfunction.

One important issue that appeared during our experiments concerns the scaling of continuous attributes. Some sequences of transactions did not match those in the monitoring structure because of a small difference in the way the scaling was done. For example, a given temperature attribute is scaled in equals intervals of 10 degrees, e.g., "Temperature:[20-30]". A new transaction containing "Temperature=31" will be scaled to the interval "Temperature:[30-40]", and two concepts are considered unrelated even if their difference is only one degree in temperature (among 342 other attributes). We believe that a more flexible approach would be more effective in comparing different, albeit similar concepts. Similarity Formal Concept Analysis (SFCA, Messai et al. (2012)), used in the aircraft cabin design use case in Section 5.4 is a alternative to consider. SFCA is able to mine concepts over continuous attributes using a similarity threshold instead of scaling data.

5.6 Chapter Summary

This chapter presented the three use cases where our approach was evaluated. Cubix was used to highlight genes co-expression clusters, filter and highlight patterns in genes expression data. Cubix was also used in the design of complex systems, where the number of simulation parameters was reduced to a few concepts. The visualisation of multi-valued lattices helped the analyst to take decisions about which intervals of parameters they want to guarantee for each comfort class. Finally, the distributed approach was used as an attempt to predict possible failures from telemetry data in real-time.

Chapter 6

Conclusions and Future Work

6.1 Recap

The advances in technology for creation, storage and dissemination of data have dramatically increased the need for tools that effectively provide users with means of identifying and understanding relevant information. As more institutions are incorporating real-time solutions in their business practices, many traditional data mining methods are simply not suitable to this end. New data mining methods have to be developed taking into account the strict constraints that stream processing requires: one pass algorithm, limited memory space and incremental changes.

Formal Concept Analysis (FCA) is a relatively young discipline and has drawn the attention of Business Intelligence (BI) analysts because of its simplicity and unique insights it provides. In FCA, the concept lattice displays the generalization and specialization relationships among objects and their attributes. This hierarchical structure can provide reasoning for classification and clustering, implication discovery and rule learning. Concept mining is a computationally intensive task and the vast majority of existing algorithms do not take advantage of parallel processing techniques.

The present thesis addressed the problem of mining and analysing formal concepts over a data stream. The proposed approach is comprised of several distributed components that carry the computation of concepts from a basic transaction, filter and transforms data, stores and provides analytic features to visually explore data. The distributed architecture is built on top of a real-time platform called *Storm*.

In addition to the concept mining architecture, we proposed several visual analytics techniques to enhance data understanding and exploration in FCA. These analytics features allow filtering, selecting, searching and clustering concepts in the concept lattice and association rules. We have proposed new visualisations such as the *Matrix of concepts*, the *Sankey lattice* and the *Heat map visualisation* for multi-valued concepts.

Transforming a concept lattice into a tree can be useful for browsing large concept lattices. In our experiments we explored statistically motivated criteria to evaluate single parent concepts in the *tree extraction* process. A case study on the web tourism domain demonstrated the usefulness of the tree extraction method.

Based on our participatory design session, we developed a visual analytics tool for FCA called **Cubix**. Cubix's workflow allows users to carry out an analysis starting from a real data set, converting it into a formal context, simplifying the context to make it manageable, and visualizing the result as a concept lattice. Cubix can also generate formal contexts from SPARQL queries to a triplestore.

Finally, we presented three use cases that demonstrate the application of the approach in real-world settings: aircraft cabin design, where new visualisations for continuous data helped analysts to quickly identify classes of comfort for passengers; Genes co-expression analysis using a combination of both analytics features and semantic integration; and real-time telemetry data analysis for anomaly detection. Experimental evaluations show that our approach is useful and provides better insights than current state of the art in FCA.

6.2 Summary of Contributions

In a nutshell, the contributions of this thesis are:

Distributed concept mining over data stream:

- Distributed concept mining architecture;

Visual Analytics:

- New visualisations for concept lattice and association rules;

- Combination of visual analytics features with conceptual metrics;
- Tree extraction algorithm from concept lattices;
- Visual analytics tool for Formal Concept Analysis: Cubix.

Although there is a growing interest on distributed frequent itemset mining over data stream, such systems are still in their infancy, and a lot of exciting work remains to be done in the design, implementation, and visualisation. I hope that this thesis will serve as a reference for the state of the art for both researchers and practitioners interested in building visual analytics and distributed Formal Concept Analysis systems.

6.3 Limitations and Future Work

Improvements in the Frequent Itemset Mining Algorithm. In the future, the algorithm could be improved by incorporating extra canonicity tests to assess if a itemset may become a closed itemset in the future. The PFCbO algorithm (Outrata and Vychodil, 2012b)) and CFI (Jiang and Gruenwald, 2006b) are good examples of how a good canonicity test can drop the computation cost when mining formal concepts. Also, with some changes, the CET Nodes can be represented in the FP-tree, thus reducing the storage space needed.

Performance Benchmarks. We are currently evaluating the performance and scalability of our approach in comparison with the state of the art algorithms. To provide a fair comparison, we implemented a distributed version of the Moment algorithm (Chi et al., 2006). The datasets are provided by the IBM Quest Synthetic Data Generator[1], which is commonly used in the benchmarks of data stream mining algorithms. The results of this evaluation are going to be published along with the benchmark code.

Large Concept Lattice Visualisation. We plan to explore other visual metaphors and more sophisticated navigation and interaction techniques for dealing with very large lattices (up to 10k concepts), and ways to navigate and zoom into different levels of concepts clusters. We also plan to release the tool under an open source license. A recent paper by Liu et al. (2013) showed how interactive visualisations for big data can

[1] http://sourceforge.net/projects/ibmquestdatagen/

take advantage of binned aggregation and data cubes to restrict the amount of data to be computed and visualised.

Evolution of the Concept Lattice over Time. Interestingly, when incrementally updating the concept lattice in real-time in the sliding window we noticed that some patterns seem to be appear systematically. For example, concepts located in the bottom of the lattice (most specific) are likely to change more frequently than those located in the top. Some concepts appear to last longer and always together with particular concepts. Concepts with long "durability" can be seen as more stable over time. There may be a way of characterizing the evens in a dynamic concept lattice. Statistical indexes may be proposed to reveal insights about those events.

Appendix A

Timeline of FCA and Frequent Pattern Mining Algorithms

Page intentionally left in blank. For reader's convenience, the image is displayed in landscape orientation in the next page.

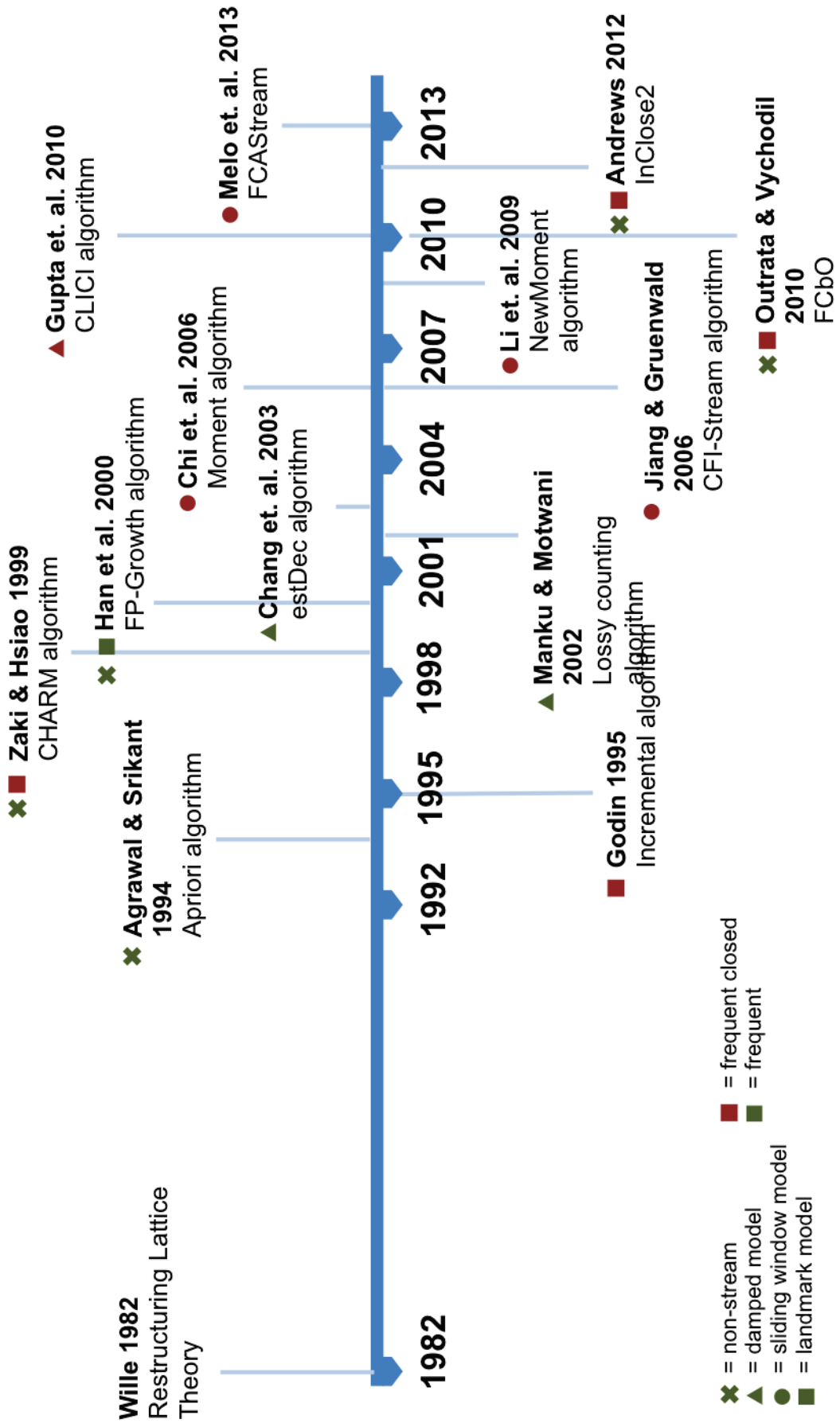


FIGURE A.1: Timeline of the mainstream algorithms for FCA and Frequent Pattern Mining.

Appendix B

Matrix Visualisation Drawing Algorithm

Algorithm 11: DrawConceptMatrix

Data: A context $\mathbb{K} = (G, M, I)$ and a list of formal concepts L.

```
1  begin
2      /* map all concepts for each pair object-attribute                    */
3      matrixConcepts ⟵ map{key : g ∈ G, value : (map{key : m ∈ M, value :
       listofconcepts⟨A, B⟩ ∈ L|gIa ∈ ⟨A, B⟩})}
4      for i ← 0 to |G| do
5          currentObject ⟵ G[i]
6          posX ⟵ i*cellHeight
7          posY ⟵ 0
8          rect ⟵ drawRectangle(posX, posY, width, cellHeight)
9          /* Iterate over pairs object-attribute                            */
10         for j ← 0 to |M| do
11             currentAttribute ⟵ M[j]
12             shift ⟵ 0
13             foreach concept ⟨A, B⟩ in matrixConcepts[currentObject][currentAttribute]
               do
14                 /* draw overlapping rectangles for each concept in the
                      relation object × attribute                             */
15                 innerWidth ⟵ cellWidth - shift*margin
16                 innerHeight ⟵ cellHeight - shift*margin
17                 drawRectangle(posX, posY, innerWidth, innerHeight)
18                 shift++
19             end
20         end
21     end
22 end
```

Appendix C

Interview Questions

1. Please shortly describe the tasks you conducted with CUBIST
2. What do you expect from a system to fulfill these tasks?
3. Did the system offer you the right information to fulfill your analytical tasks?
   (a) If yes, what kind of information and system functionality provided did you find especially helpful?
   (b) If no, what was missing from your point of view (regarding data provided/ visualization possibilities/ interaction possibilities) to derive the desired information?
4. Did you discover new facts during your analysis tasks that you had not expected to discover at all before?
   (a) If yes, what kind of new facts did you discover that were most surprising for you?
   (b) No.
5. If the tasks fulfilled are typical for your daily work, do you think the tool can enrich your daily work by offering new ways to analyze your data?
   (a) Yes, because...
   (b) No, because...
   (c) Partly, because...

6. Which analytical systems do you currently use in your daily work?

   (a) The tools are for example:

   (b) None.

7. From your point of view:

   (a) Please shortly describe what is missing in current systems to use them effectively for your daily tasks:

   (b) Do you think CUBIST fills an analytical gap or provides functionalities that better fit your analytical tasks? Why do you think so?

8. Next to the data/use case currently implemented in the system, do you see any content from your daily life (private and professional) to be integrated in the system in future? Why do you think this would be benefit?

9. For which kind of tasks from you daily work do you believe the system can be especially useful? Please describe the tasks and the possible benefit shortly:

10. For which kind of tasks from your daily work do you believe the system is annoying / ineffective? Please describe the tasks and the possible drawbacks shortly:

Appendix D

Post-Evaluation Surveys

The purpose and function of the component is clear.	strongly agree ☐ ☐ ☐ neutral ☐ ☐ ☐ strongly disagree	n/a ☐
The component is easy to understand.	strongly agree ☐ ☐ ☐ neutral ☐ ☐ ☐ strongly disagree	n/a ☐
The component is easy to use.	strongly agree ☐ ☐ ☐ neutral ☐ ☐ ☐ strongly disagree	n/a ☐
The interface is appealing and attractive.	strongly agree ☐ ☐ ☐ neutral ☐ ☐ ☐ strongly disagree	n/a ☐
The component is useful.	strongly agree ☐ ☐ ☐ neutral ☐ ☐ ☐ strongly disagree	n/a ☐
For its purpose (as I understand it), the component provides sufficiently enough functionality.	strongly agree ☐ ☐ ☐ neutral ☐ ☐ ☐ strongly disagree	n/a ☐
The time needed to navigate and explore the data was appropriate in comparison to the insights I gained from the analysis.	strongly agree ☐ ☐ ☐ neutral ☐ ☐ ☐ strongly disagree	n/a ☐
For some kinds of information needs or queries, particularly this component (or similar components based on the same approach) is useful.	strongly agree ☐ ☐ ☐ neutral ☐ ☐ ☐ strongly disagree	n/a ☐
I have similar functionalities in the tools I usually use.	strongly agree ☐ ☐ ☐ neutral ☐ ☐ ☐ strongly disagree	n/a ☐

FIGURE D.1: UI Component Questions.

The visualizations were easy to understand.	strongly agree			neutral			strongly disagree	n/a
	☐	☐	☐	☐	☐	☐	☐	☐
There are visualizations available that did fit my tasks very well.	strongly agree			neutral			strongly disagree	n/a
	☐	☐	☐	☐	☐	☐	☐	☐
The integration of different visualizations was helpful for fulfilling my task.	strongly agree			neutral			strongly disagree	n/a
	☐	☐	☐	☐	☐	☐	☐	☐
The integration of different visualization was confusing.	strongly agree			neutral			strongly disagree	n/a
	☐	☐	☐	☐	☐	☐	☐	☐
It is clear how the different visualization interact.	strongly agree			neutral			strongly disagree	n/a
	☐	☐	☐	☐	☐	☐	☐	☐
The navigation/interaction functionalities were easy to understand and apply.	strongly agree			neutral			strongly disagree	n/a
	☐	☐	☐	☐	☐	☐	☐	☐
It was easy to follow the steps performed by the system when using the interaction functionalities.	strongly agree			neutral			strongly disagree	n/a
	☐	☐	☐	☐	☐	☐	☐	☐

FIGURE D.2: Visualisations Questions.

Bibliography

FAO. *Agribusiness Handbooks*, volume 7. European Bank for Reconstruction and Environment, 1999.

Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis*. Springer, mathematical foundations edition, 1999.

Ducrou, P. J., Eklund, and T. Wilson. An intelligent user interface for browsing and searching mpeg-7 images using concept lattices. In *CLA 2006*, volume 4923, pages 1–21. Springer-Verlag Berlin Heidelberg, 2008.

Peter Becker and Joachim Correia. The toscanaj suite for implementing conceptual information systems. *Formal Concept Analysis*, pages 324–348, 2005. doi: 10.1007/11528784_17. URL http://dx.doi.org/10.1007/11528784_17.

Lotfi Lakhal and Gerd Stumme. *Efficient Mining of Association Rules Based on Formal Concept Analysis*, volume 3626 of *LNAI*, pages 180–195. Springer, Heidelberg, 2005. URL http://www.kde.cs.uni-kassel.de/stumme/papers/2005/lakhal2005efficient.pdf.

Petr Krajca, Jan Outrata, and Vilem Vychodil. V.: Parallel recursive algorithm for fca. In *Palacky University, Olomouc*, pages 71–82, 2008.

Claudio Carpineto and Giovanni Romano. Exploiting the potential of concept lattices for information retrieval with credo. *JOURNAL OF UNIVERSAL COMPUTER SCIENCE*, 10:985–1013, 2004a.

Eklund, Peter, Villerd, and Jean. A survey of hybrid representations of concept lattices. *Conceptual Knowledge Processing Formal Concept Analysis*, pages 296– 311, 2010.

Benjamin Bach. Facettice. Diplom, Technische Universitt Dresden, 2010.

Sergei Kuznetsov, Sergei Obiedkov, and Camille Roth. Reducing the representation complexity of lattice-based taxonomies. In *Proceedings of the 15th international conference on Conceptual Structures: Knowledge Architectures for Smart Applications*, ICCS '07, pages 241–254, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 978-3-540-73680-6. doi: 10.1007/978-3-540-73681-3_18. URL http://dx.doi.org/10.1007/978-3-540-73681-3_18.

Simon Andrews. In-close2, a high performance formal concept miner. In *Proceedings of the 19th international conference on Conceptual structures for discovering knowledge*, ICCS'11, pages 50–62, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-22687-8. URL http://dl.acm.org/citation.cfm?id=2032828.2032834.

Jan Outrata and Vilem Vychodil. Fast algorithm for computing fixpoints of galois connections induced by object-attribute relational data. *Inf. Sci.*, 185(1):114–127, February 2012a. ISSN 0020-0255. doi: 10.1016/j.ins.2011.09.023. URL http://dx.doi.org/10.1016/j.ins.2011.09.023.

E. M. Norris. An algorithm for computing the maximal rectangles in a binary relation. *Revue Roumaine de Mathématiques Pures et Appliquées*, 23(2):243–250, 1978.

J. P. Bordat. Calcul pratique du treillis de galois d'une correspondance. *Informatiques et Sciences Humaines*, 96:31–47, 1986.

Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman, editors. *Readings in information visualization: using vision to think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999. ISBN 1-55860-533-9.

W. Andrews, M. Beyer, A. Bitterer, N. Chandler, B. Gassman, G. Herschel, Hostmann, D. Newman, T. Payne, J. Radcliffe, N. Rayner, S. Schlegel, and A. White. Hype Cycle for Business Intelligence and Performance Management. *Gartner, Inc.*, 2008.

Rudolf Wille. Restructuring lattice theory: an approach based on hierarchies of concepts. In Ivan Rival, editor, *Ordered sets*, pages 445–470, Dordrecht–Boston, 1982. Reidel.

Thanh Tho Quan, Siu Cheung Hui, and Tru Hoang Cao. A fuzzy fca-based approach to conceptual clustering for automatic generation of concept hierarchy on uncertainty data. In Vclav Snsel and Radim Belohlvek, editors, *Proceedings of the CLA 2004 International Workshop on Concept Lattices and their Applications, Ostrava,*

Czech Republic, September 23-24, 2004, volume 110 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004. doi: http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS//Vol-110/paper3.pdf.

Claudio Carpineto and Giovanni Romano. *Concept Data Analysis: Theory and Applications*. John Wiley & Sons, 2004b. ISBN 0470850558.

Ruoming Jin, Yuri Breitbart, and Chibuike Muoh. Data discretization unification. In *ICDM*, pages 183–192. IEEE Computer Society, 2007.

Susan P. Imberman, Ph. D, and Bernard Domanski. Finding association rules from quantitative data using data booleanization, 1999.

Colin Ware, Helen Purchase, Linda Colpoys, and Matthew Mcgill. Cognitive measurements of graph aesthetics. *Information Visualization*, 1:103–110, 2002.

Stumme Gerd, Rafik Taouil, Yves Bastide, Nicolas Pasquier, and Lotfi Lakhal. Computing iceberg concept lattices with titanic. *Data & Knowledge Engineering*, 42:189–222, August 2002a.

Mohammed J. Zaki and Ching-Jui Hsiao. Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Transactions on Knowledge and Data Engineering*, 17:2005, 2005.

Bénédicte Le Grand, Michel Soto, and Marie-Aude Aufaure. Conceptual and spatial footprints for complex systems analysis: Application to the semantic web. In *Proceedings of the 20th International Conference on Database and Expert Systems Applications*, DEXA '09, pages 114–127, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-03572-2. doi: 10.1007/978-3-642-03573-9_9. URL http://dx.doi.org/10.1007/978-3-642-03573-9_9.

Hannah Blau, Neil Immerman, and David Jensen. A Visual Language for Querying and Updating Graphs. Technical Report 037, University of Massachusetts Amherst Computer Science, 2002.

Isabel F. Cruz, Alberto O. Mendelzon, and Peter T. Wood. A graphical query language supporting recursion. In *Proceedings of the 1987 ACM SIGMOD international conference on Management of data*, SIGMOD '87, pages 323–330, New York,

NY, USA, 1987. ACM. ISBN 0-89791-236-5. doi: 10.1145/38713.38749. URL http://doi.acm.org/10.1145/38713.38749.

Mariano Consens and Alberto Mendelzon. Hy+: a hygraph-based query and visualization system. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, SIGMOD '93, pages 511–516, New York, NY, USA, 1993. ACM. ISBN 0-89791-592-5. doi: 10.1145/170035.171537. URL http://doi.acm.org/10.1145/170035.171537.

Marie-Aude Aufaure Michel Soto, Benedicte Le Grand. Spatial visualisation of conceptual data. *International Conference Information Visualisation*, pages 57–61, 2009.

Peter Eklund and Jean Villerd. A survey of hybrid representations of concept lattices in conceptual knowledge processing. In *Proceedings of the 8th international conference on Formal Concept Analysis*, ICFCA'10, pages 296–311, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-11927-1, 978-3-642-11927-9. doi: 10.1007/978-3-642-11928-6_21. URL http://dx.doi.org/10.1007/978-3-642-11928-6_21.

Sergei O. Kuznetsov and Sergei Obiedkov. Comparing performance of algorithms for generating concept lattices. *Journal of Experimental and Theoretical Artificial Intelligence*, 14:189–216, 2002.

Bernhard Ganter. Two basic algorithms in concept analysis. In *Proceedings of the 8th international conference on Formal Concept Analysis*, ICFCA'10, pages 312–340, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-11927-1, 978-3-642-11927-9. doi: 10.1007/978-3-642-11928-6_22. URL http://dx.doi.org/10.1007/978-3-642-11928-6_22.

Petr Krajca, Jan Outrata, and Vilm Vychodil. Advances in algorithms based on cbo. In Marzena Kryszkiewicz and Sergei A. Obiedkov, editors, *CLA*, volume 672 of *CEUR Workshop Proceedings*, pages 325–337. CEUR-WS.org, 2010. URL http://dblp.uni-trier.de/db/conf/cla/cla2010.html#KrajcaOV10.

Jan Outrata and Vilem Vychodil. Fast algorithm for computing fixpoints of galois connections induced by object-attribute relational data. *Inf. Sci.*, 185(1):114–127, February 2012b. ISSN 0020-0255. doi: 10.1016/j.ins.2011.09.023. URL http://dx.doi.org/10.1016/j.ins.2011.09.023.

Simon Andrews. In-close, a fast algorithm for computing formal concepts. In *the Seventeenth International Conference on Conceptual Structures*, 2009.

Robert Godin, Rokia Missaoui, and Hassan Alaoui. Incremental concept formation algorithms based on galois (concept) lattices, 1995.

Dean Van Der Merwe, Sergei Obiedkov, and Derrick Kourie. Addintent: A new incremental algorithm for constructing concept lattices. In *Lecture Notes in Computer Science Vol 2961*, pages 372–385, 2004.

Petko Valtchev and Rokia Missaoui. Building concept (galois) lattices from parts: Generalizing the incremental methods. In *Proceedings of the 9th International Conference on Conceptual Structures: Broadening the Base*, ICCS '01, pages 290–303, London, UK, UK, 2001. Springer-Verlag. ISBN 3-540-42344-3. URL http://dl.acm.org/citation.cfm?id=645494.657555.

Uta Priss. Lattice-based information retrieval. *Knowledge Organization*, 27:132–142, 2000.

Stumme Gerd, Rafik Taouil, Yves Bastide, Nicolas Pasquier, and Lotfi Lakhal. Computing iceberg concept lattices with titanic. *Data & Knowledge Engineering*, 42:189–222, August 2002b.

Claudio Carpineto and Giovanni Romano. Using concept lattices for text retrieval and mining. In *Formal Concept Analysis*, pages 161–179, 2005.

Simon Andrews and Constantinos Orphanides. Analysis of large data sets using formal concept lattices. In M. Kryszkiewicz and S Obiedkov, editors, *7th International Conference on Concept Lattices and Their Applications*, pages 104–115. University of Seville, Seville, 2010. URL http://shura.shu.ac.uk/3706/.

Robert Godin, Rokia Missaoui, and Alain April. Experimental comparison of navigation in a galois lattice with conventional information retrieval methods. *International Journal of Man-machine Studies*, 38:747–767, 1998.

Richard Cole and Peter Eklund. Scalability in formal concept analysis. *Computational Intelligence*, 15:11–27, 1999.

Ch. Aswani Kumar and S. Srinivas. Short communication: Concept lattice reduction using fuzzy k-means clustering. *Expert Syst. Appl.*, 37(3):2696–2704, March 2010.

ISSN 0957-4174. doi: 10.1016/j.eswa.2009.09.026. URL http://dx.doi.org/10.1016/j.eswa.2009.09.026.

Vclav Snsel, Martin Polovincak, Hussam M. Dahwa Abdulla, and Zdenek Horak. On concept lattices and implication bases from reduced contexts. In *ICCS Supplement'08*, pages 83–90, 2008.

Karen S. Cheung and Douglas Vogel. Complexity reduction in lattice-based information retrieval. *Inf. Retr.*, 8(2):285–299, apr 2005. ISSN 1386-4564.

Sébastien Ferré and Olivier Ridoux. A logical generalization of formal concept analysis. In *8th International Conference on Conceptual Structures, ICCS 2000*, volume 1867 of *LNCS*, pages 371–384. Springer, 2000.

Bernhard Ganter and Sergei O. Kuznetsov. Pattern structures and their projections. In *9th International Conference on Conceptual Structures, ICCS 2001*, volume 2120 of *LNCS*, pages 129–142. Springer, 2001.

Nizar Messai, Marie-Dominique Devignes, Amedeo Napoli, and Malika Smaïl-Tabbone. Many-valued concept lattices for conceptual clustering and information retrieval. In *18th European Conference on Artificial Intelligence ECAI 2008*, volume 178, pages 127–131. IOS Press, 2008.

Nizar Messai, Marie-Dominique Devignes, Amedeo Napoli, and Malika Tabbone. Using domain knowledge to guide lattice-based complex data exploration. In *19th European Conference on Artificial Intelligence ECAI 2010*, volume 215, pages 847–852. IOS Press, 2010.

Bernhard Ganter and Christian Meschke. A formal concept analysis approach to rough data tables. In *Proceedings of the 12th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, RSFDGrC '09, pages 117–126, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-10645-3. doi: 10.1007/978-3-642-10646-0_14. URL http://dx.doi.org/10.1007/978-3-642-10646-0_14.

Ruggero Pensa and Jean-Francois Boulicaut. Towards fault-tolerant formal concept analysis. In *Proc. 9th Congress of the Italian Association for Artificial Intelligence AI*IA'05*, LNAI, pages 212–223. Springer, September 2005. URL http://liris.cnrs.fr/publis/?id=1823.

Jean-Francois Boulicaut, Artur Bykowski, and Christophe Rigotti. Free-Sets: a Condensed Representation of Boolean Data for the Approximation of Frequency Queries. *Data Mining and Knowledge Discovery*, 7(1):5–22, 2003. URL http://liris.cnrs.fr/publis/?id=741.

Petr Krajca and Vilem Vychodil. Distributed algorithm for computing formal concepts using map-reduce framework. In *Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VIII*, IDA '09, pages 333–344, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-03914-0. doi: 10.1007/978-3-642-03915-7_29. URL http://dx.doi.org/10.1007/978-3-642-03915-7_29.

M.J. Zaki. Parallel and distributed association mining: a survey. *Concurrency, IEEE*, 7(4):14–25, 1999. ISSN 1092-3063. doi: 10.1109/4434.806975.

T.Sathish Kumar, V.Kavitha, and Dr.T.Ravichandran. Article:efficient tree based distributed data mining algorithms for mining frequent patterns. *International Journal of Computer Applications*, 10(1):11–16, November 2010. Published By Foundation of Computer Science.

Anamika Gupta, Vasudha Bhatnagar, and Naveen Kumar. Mining closed itemsets in data stream using formal concept analysis. In *Proceedings of the 12th international conference on Data warehousing and knowledge discovery*, DaWaK'10, pages 285–296, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15104-3, 978-3-642-15104-0. URL http://dl.acm.org/citation.cfm?id=1881923.1881953.

C. Melo, B. Le-Grand, M. Aufaure, and A. Bezerianos. Extracting and visualising tree-like structures from concept lattices. In *15th International Conference on Information Visualisation*, pages 261 –266, july 2011a. doi: 10.1109/IV.2011.46.

Sergei O. Kuznetsov and Sergei A. Obiedkov. Algorithms for the construction of concept lattices and their diagram graphs. In *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*, PKDD '01, pages 289–300, London, UK, UK, 2001. Springer-Verlag. ISBN 3-540-42534-9. URL http://dl.acm.org/citation.cfm?id=645805.669994.

Daniel A. Keim, Florian Mansmann, Jrn Schneidewind, Hartmut Ziegler, and Jim Thomas. Visual analytics: Scope and challenges. December 2008. Visual Data Mining: Theory, Techniques and Tools for Visual Analytics, Springer, Lecture Notes In Computer Science (lncs).

Elma Akand, Michael Bain, and Mark Temple. A Visual Analytics Approach to Augmenting Formal Concepts with Relational Background Knowledge in a Biological Domain. December 2010. URL http://krr.meraka.org.za/~{}aow2010/Akand-etal.pdf.

Jon Ducrou, Björn Vormbrock, and Peter W. Eklund. Fca-based browsing and searching of a collection of images. In Henrik Schärfe, Pascal Hitzler, and Peter Øhrstrøm, editors, *ICCS*, volume 4068 of *Lecture Notes in Computer Science*, pages 203–214. Springer, 2006. ISBN 3-540-35893-5.

Jean Villerd, Sylvie Ranwez, Michel Crampes, David Carteret, and Jean Michel Penalva. Using concept lattices for visual navigation assistance in large databases: Application to a pantent database. In Peter W. Eklund, Jean Diatta, and Michel Liquiere, editors, *CLA*, volume 331 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

Yun Chi, Haixun Wang, Philip S. Yu, and Richard R. Muntz. Catch the moment: maintaining closed frequent itemsets over a data stream sliding window. *Knowl. Inf. Syst.*, 10(3):265–294, October 2006. ISSN 0219-1377. doi: 10.1007/s10115-006-0003-0. URL http://dx.doi.org/10.1007/s10115-006-0003-0.

Erik D. Demaine, Alejandro López-Ortiz, and J. Ian Munro. Frequency estimation of internet packet streams with limited space. In *Proceedings of the 10th Annual European Symposium on Algorithms*, ESA '02, pages 348–360, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-44180-8. URL http://dl.acm.org/citation.cfm?id=647912.740658.

Y. Dora Cai, David Clutter, Greg Pape, Jiawei Han, Michael Welge, and Loretta Auvil. Maids: mining alarming incidents from data streams. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, pages 919–920, New York, NY, USA, 2004. ACM. ISBN 1-58113-859-8. doi: 10.1145/1007568.1007695. URL http://doi.acm.org/10.1145/1007568.1007695.

Mihail Halatchev and Le Gruenwald. Estimating missing values in related sensor data streams. In Jayant R. Haritsa and T. M. Vijayaraman, editors, *Advances in Data Management 2005, Proceedings of the Eleventh International Conference on Management of Data, January 6, 7, and 8, 2005, Goa, India*, pages 83–94. Computer Society of India, 2005. doi: http://comad2005.persistent.co.in/COMAD2005Proc/pages083-094.pdf.

Nan Jiang and Le Gruenwald. Research issues in data stream association rule mining. *SIGMOD Rec.*, 35(1):14–19, March 2006a. ISSN 0163-5808. doi: 10.1145/1121995.1121998. URL http://doi.acm.org/10.1145/1121995.1121998.

Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2):1–12, May 2000a. ISSN 0163-5808. doi: 10.1145/335191.335372. URL http://doi.acm.org/10.1145/335191.335372.

Mohammed J. Zaki and Ching-Jui Hsiao. Charm: An efficient algorithm for closed association rule mining. Technical report, Computer Science, Rensselaer Polytechnic Institute, 1999.

Joong Hyuk Chang and Won Suk Lee. Finding recent frequent itemsets adaptively over online data streams. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 487–492, New York, NY, USA, 2003. ACM. ISBN 1-58113-737-0. doi: 10.1145/956750.956807. URL http://doi.acm.org/10.1145/956750.956807.

Hua-Fu Li, Chin-Chuan Ho, and Suh-Yin Lee. Incremental updates of closed frequent itemsets over continuous data streams. *Expert Syst. Appl.*, 36(2):2451–2458, March 2009. ISSN 0957-4174. doi: 10.1016/j.eswa.2007.12.054. URL http://dx.doi.org/10.1016/j.eswa.2007.12.054.

Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. In *VLDB*, pages 346–357, 2002a.

Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499. Morgan Kaufmann, 1994. ISBN 1-55860-153-8.

Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. In *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB '02, pages 346–357. VLDB Endowment, 2002b. URL http://dl.acm.org/citation.cfm?id=1287369.1287400.

Jian Pei, Jiawei Han, and Runying Mao. Closet: An efficient algorithm for mining frequent closed itemsets. pages 21–30, 2000.

Jianyong Wang, Jiawei Han, and Jian Pei. Closet+: Searching for the best strategies for mining frequent closed itemsets. pages 236–245, 2003.

Gsta Grahne and Jianfei Zhu. Efficiently using prefix-trees in mining frequent itemsets, 2003.

Claudio Lucchese. Dci closed: A fast and memory efficient algorithm to mine frequent closed itemsets. In *In Proc. of the IEEE ICDM 2004 Workshop on Frequent Itemset Mining Implementations (FIMI04*, 2004.

Endu Duneja and A.k. Sachan. Article: A survey on frequent itemset mining with association rules. *International Journal of Computer Applications*, 46(23):18–24, May 2012. Published by Foundation of Computer Science, New York, USA.

Nan Jiang and Le Gruenwald. Cfi-stream: mining closed frequent itemsets in data streams. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 592–597, New York, NY, USA, 2006b. ACM. ISBN 1-59593-339-5. doi: 10.1145/1150402.1150473. URL http://doi.acm.org/10.1145/1150402.1150473.

Junbo Chen and ShanPing Li. Gc-tree: a fast online algorithm for mining frequent closed itemsets. In *Proceedings of the 2007 international conference on Emerging technologies in knowledge discovery and data mining*, PAKDD'07, pages 457–468, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3-540-77016-X, 978-3-540-77016-9. URL http://dl.acm.org/citation.cfm?id=1780582.1780630.

J. Han, J.Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM-SIGMID International Conference on Management of Data*, pages 1–12, 2000b.

Rudolf Wille. Lattices in data analysis: how to draw them with a computer. In Ivan Rival, editor, *Algorithms and order*, pages 33–58, Dordrecht–Boston, 1989. Kluwer.

Camille Roth, Sergei Obiedkov, and Derrick Kourie. Towards concise representation for taxonomies of epistemic communities. In *Proceedings of the 4th international conference on Concept lattices and their applications*, CLA'06, pages 240–255, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 3-540-78920-0, 978-3-540-78920-8. URL http://dl.acm.org/citation.cfm?id=1793623.1793644.

Peter Eades and Roberto Tamassia. Algorithms for drawing graphs: An annotated bibliography. Technical report, Providence, RI, USA, 1988.

Anastasia Bezerianos, Pierre Dragicevic, Jean-Daniel Fekete, Juhee Bae, and Ben Watson. GeneaQuilts: A System for Exploring Large Genealogies. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1073–1081, October 2010. doi: 10.1109/TVCG.2010.159. URL http://hal.inria.fr/inria-00532939.

Mario Schmidt. *Der Einsatz von Sankey-Diagrammen im Stoffstrommanagement*. Number 124 in Beitrge der Hochschule Pforzheim. Hochsch., Pforzheim, 2006. URL http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+525615741&sourceid=fbw_bibsonomy.

James J. Thomas and Kristin A. Cook. A visual analytics agenda. *IEEE Comput. Graph. Appl.*, 26(1):10–13, January 2006. ISSN 0272-1716. doi: 10.1109/MCG.2006.5. URL http://dx.doi.org/10.1109/MCG.2006.5.

C. Carpineto and G. Romano. Ulysses: a lattice-based multiple interaction strategy retrieval interface. In *Human-Computer Interaction*, pages 91–104. LNCS 1015-Springer, 1995.

Gary Hall. *Pro WPF and Silverlight MVVM: Effective Application Development with Model-View-ViewModel*. Apress, Berkely, CA, USA, 1st edition, 2010. ISBN 1430231629, 9781430231622.

Matthew O. Ward. Linking and brushing. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*, pages 1623–1626. Springer US, 2009. ISBN 978-0-387-35544-3, 978-0-387-39940-9.

J. A. Hartigan and M. A. Wong. A K-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

George G. Robertson, Jock D. Mackinlay, and Stuart K. Card. Cone trees: animated 3d visualizations of hierarchical information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '91, pages 189–194, New York, NY, USA, 1991. ACM. ISBN 0-89791-383-3. doi: 10.1145/108844.108883. URL http://doi.acm.org/10.1145/108844.108883.

Christian Bachmaier, Ulrik Brandes, and Barbara Schlieper. Drawing phylogenetic trees. In *Proceedings of the 16th international conference on Algorithms and Computation*, ISAAC'05, pages 1110–1121, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3-540-30935-7, 978-3-540-30935-2. doi: 10.1007/11602613_110. URL http://dx.doi.org/10.1007/11602613_110.

Quang Vinh Nguyen and Mao Lin Huang. A space-optimized tree visualization. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, INFOVIS '02, pages 85–, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7695-1751-X. URL http://dl.acm.org/citation.cfm?id=857191.857748.

Brian Johnson and Ben Shneiderman. Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the 2nd conference on Visualization '91*, VIS '91, pages 284–291, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press. ISBN 0-8186-2245-8. URL http://dl.acm.org/citation.cfm?id=949607.949654.

Benedicte Le Grand, Michel Soto, and David Dodds. XML Topic Maps and Semantic Web Mining. In *XML Conference & Exposition 2001*, December 2001. URL http://www.idealliance.org/papers/xml2001/papers/html/04-04-05.html.

Trad Mohamed Riadh, Bénédicte Le Grand, Marie-Aude Aufaure, and Michel Soto. Powerconcept: Conceptual metrics distributed computation. In *Proceedings of the 8th international conference on Formal Concept Analysis*, ICFCA'10, Berlin, Heidelberg, 2010. Springer-Verlag.

Tim Hannan and Alex Pogel. Spring-based lattice drawing highlighting conceptual similarity. In *Proceedings of the 4th international conference on Formal Concept Analysis*,

ICFCA'06, pages 264–279, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-32203-5, 978-3-540-32203-0. doi: 10.1007/11671404_18. URL http://dx.doi.org/10.1007/11671404_18.

John Stasko. An evaluation of space-filling information visualizations for depicting hierarchical structures. *Int. J. Hum.-Comput. Stud.*, 53(5):663–694, November 2000. ISSN 1071-5819. doi: 10.1006/ijhc.2000.0420. URL http://dx.doi.org/10.1006/ijhc.2000.0420.

Cássio A. Melo, Alexander Mikheev, Bénédicte Le Grand, and Marie-Aude Aufaure. Cubix: A visual analytics tool for conceptual and semantic data. In Jilles Vreeken, Charles Ling, Mohammed Javeed Zaki, Arno Siebes, Jeffrey Xu Yu, Bart Goethals, Geoffrey I. Webb, and Xindong Wu, editors, *ICDM Workshops*, pages 894–897. IEEE Computer Society, 2012. ISBN 978-1-4673-5164-5.

Cássio A. Melo, Bénédicte Le Grand, Marie-Aude Aufaure, and Anastasia Bezerianos. Extracting and visualising tree-like structures from concept lattices. In Ebad Banissi, Stefan Bertschi, Remo Aslak Burkhard, Urska Cvek, Martin J. Eppler, Camilla Forsell, Georges G. Grinstein, Jimmy Johansson, Sarah Kenderdine, Francis T. Marchese, Carsten Maple, Marjan Trutschl, Muhammad Sarfraz, Liz J. Stuart, Anna Ursyn, and Theodor G. Wyeld, editors, *IV*, pages 261–266. IEEE Computer Society, 2011b. ISBN 978-1-4577-0868-8.

Jonathan Lazar, Jinjuan Feng, and Harry Hochheiser. *Research Methods in Human-Computer Interaction*. Wiley, Indianapolis, IN, 2010. ISBN 978-0-470-72337-1.

Daniel Fitton, Keith Cheverst, Christian Kray, Alan J. Dix, Mark Rouncefield, and George Saslis-Lagoudakis. Rapid prototyping and user-centered design of interactive display-based systems. *IEEE Pervasive Computing*, 4(4):58–66, 2005. URL http://dblp.uni-trier.de/db/journals/pervasive/pervasive4.html#FittonCKDRS05.

JoAnn T. Hackos and Janice C. Redish. *User and task analysis for interface design*. John Wiley & Sons, Inc., New York, NY, USA, 1998. ISBN 0-471-17831-4.

Robert Amar, James Eagan, and John Stasko. Low-level components of analytic activity in information visualization. In *Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*, INFOVIS '05, pages 15–, Washington, DC,

USA, 2005. IEEE Computer Society. ISBN 0-7803-9464-x. doi: 10.1109/INFOVIS.2005.24. URL http://dx.doi.org/10.1109/INFOVIS.2005.24.

Bongshin Lee, Catherine Plaisant, Cynthia Sims Parr, Jean-Daniel Fekete, and Nathalie Henry. Task taxonomy for graph visualization. In *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*, BELIV '06, pages 1–5, New York, NY, USA, 2006. ACM. ISBN 1-59593-562-2. doi: 10.1145/1168149.1168168. URL http://doi.acm.org/10.1145/1168149.1168168.

Marsha E. Fonteyn, Benjamin Kuipers, and Susan J. Grobe. A Description of Think Aloud Method and Protocol Analysis. *Qual Health Res*, 3(4):430–441, November 1993. doi: 10.1177/104973239300300403. URL http://dx.doi.org/10.1177/104973239300300403.

Amela Prelić, Stefan Bleuler, Philip Zimmermann, Anja Wille, Peter Bühlmann, Wilhelm Gruissem, Lars Hennig, Lothar Thiele, and Eckart Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, May 2006. ISSN 1367-4803. doi: 10.1093/bioinformatics/btl060. URL http://dx.doi.org/10.1093/bioinformatics/btl060.

Mehdi Kaytoue-Uberall, Sbastien Duplessis, and Amedeo Napoli. Using formal concept analysis for the extraction of groups of co-expressed genes. In *MCO'08*, pages 439–449, 2008.

Sylvain Blachon, Ruggero Pensa, Jrmy Besson, Cline Robardet, and Jean-Francois Boulicaut. Clustering formal concepts to discover biologically relevant knowledge from gene expression data. *In Silico Biology*, 7(0033), July 2007. URL http://liris.cnrs.fr/publis/?id=2970.

S. Andrews and K. M^{c}Leod. Gene co-expression in mouse embryo tissues. In *Proceedings of the 1st Combining and Uniting Business Intelligence with Semantic Technologies workshop*, volume 753, 2011. doi: urn:nbn:de:0074-753-1.

Cassio Melo, Marie-Aude Aufaure, Constantinos Orphanides, Simon Andrews, Kenneth McLeod, and Albert Burger. A conceptual approach to gene expression analysis enhanced by visual analytics. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 1314–1319, New York, NY, USA, 2013. ACM.

ISBN 978-1-4503-1656-9. doi: 10.1145/2480362.2480610. URL http://doi.acm.org/10.1145/2480362.2480610.

Dung Bui, Mohamed Hamdaoui, and Florian de Vuyst. Reduced-order modeling of parametrized finite element solutions by POD-ISAT technique. application to aircraft air control system. In *International Conference on Computational Methods for Coupled Problems in Science and Engineering COUPLED PROBLEMS 2011*.

ASHRAE. ASHRAE Standard, ANSI/ASHRAE Standard 55-2004: thermal environmental conditions for human occupancy. Technical report, 2004.

Nizar Messai, Cássio A. Melo, Mohamed Hamdaoui, Dung Bui, and Marie-Aude Aufaure. Conceptual analysis of complex system simulation data for decision support: Application to aircraft cabin design. In Laszlo Szathmary and Uta Priss, editors, *CLA*, volume 972 of *CEUR Workshop Proceedings*, pages 199–210. CEUR-WS.org, 2012.

Zheng Zhang, Jun Li, C. N. Manikopoulos, Jay Jorgenson, and Jose Ucles. A hierarchical anomaly network intrusion detection system using neural network classification. In *CD-ROM Proceedings of 2001 WSES International Conference on: Neural Networks and Applications (NNA 01*, 2001.

Ar Lazarevic, Aysel Ozgur, Levent Ertoz, Jaideep Srivastava, and Vipin Kumar. A comparative study of anomaly detection schemes in network intrusion detection. In *In Proceedings of the Third SIAM International Conference on Data Mining*, 2003.

Ningning Wu and Jing Zhang. Factor-analysis based anomaly detection and clustering. *Decision Support Systems*, 42(1):375 – 389, 2006. ISSN 0167-9236. doi: 10.1016/j.dss.2005.01.005. URL http://www.sciencedirect.com/science/article/pii/S0167923605000096.

Tarem Ahmed, Boris Oreshkin, and Mark Coates. Machine learning approaches to network anomaly detection. In *Proceedings of the 2nd USENIX workshop on Tackling computer systems problems with machine learning techniques*, SYSML'07, pages 7:1–7:6, Berkeley, CA, USA, 2007. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1361442.1361449.

Zhicheng Liu, Biye Jiang, and Jeffrey Heer. immens: Real-time visual querying of big data. *Computer Graphics Forum (Proc. EuroVis)*, 32, 2013. URL http://vis.stanford.edu/papers/immens.