BABEŞ-BOLYAI UNIVERSITY, CLUJ-NAPOCA, ROMANIA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Conceptual Visualization and Navigation Methods for Polyadic Formal Concept Analysis

PhD Thesis

PhD Supervisors:

Prof. Dr. Florian Mircea Boian, Babeş-Bolyai Cluj-Napoca

Prof. Dr. rer. nat. Sebastian Rudolph, TU Dresden

PhD Student: Diana Troancă

2016

Abstract

Formal concept analysis (FCA) is the core of Conceptual Knowledge Processing, being closely related to a deeper understanding of existing facts and relationships, while at the same time trying to find explanations for their existence. Polyadic formal concept analysis is an extension of classical FCA that instead of binary relations uses an n-ary incidence relation to define formal concepts, i.e. data clusters in which all elements are interrelated. In this thesis, we introduce new methods of visualization, navigation and exploration based on polyadic formal concept analysis for n-adic datasets with $n \geq 3$. In Chapter 3.1, we introduce a triadic approach to study the Web usage behavior of an e-learning platform. We analyze temporal aspects of the users' behavior and visualize the results in a circular layout using the `Circos` tool. In the subsequent chapter, we define methods to reduce the size of a triadic dataset without altering its underlying structure. For this purpose, we extend the notions of clarification and reduction from the dyadic to the triadic setting and show that these processes have an influence solely on the efficiency and not on the results of any further analysis. In Chapter 3.3, we introduce a navigation paradigm based on a reachability relation among formal concepts. This relation gives rise to so-called reachability clusters containing mutually reachable concepts. We discuss theoretical aspects about the properties of the formal concepts arising from the defined reachability relation and describe the framework of the proposed navigation paradigm. In Chapter 3.4, we consider the problem of satisfiability of membership constraints in order to determine if a formal concept exists whose components include and exclude certain elements. We analyze the computational complexity of this problem for particular cases as well as for the general n-adic problem and present an answer set programming encoding for the membership constraint satisfaction problem. Next, we propose a navigation paradigm based on membership constraints and implement it for the dyadic, triadic and tetradic case using different strategies, one based on the proposed ASP encoding and one using an exhaustive search in the whole concept set, precomputed with an external FCA tool. We evaluate and compare the implementations and discuss the limitations and the possibility of generalizations of each approach. In the final part of the thesis, we describe the achievements of our research as well as possible directions for future work.

Acknowledgements

I would like to express my sincere gratitude to both of my advisors, Prof. Florian-Mircea Boian and Prof. Sebastian Rudolph, for the continuous support of my PhD study and related research, for their patience, motivation, and immense knowledge. In addition, I would like to thank Prof. Sebastian Rudolph for the financial support that allowed me to join his research group and spend 3 months at the host university TU Dresden, and also to attend various conferences, workshops and doctoral consortiums. Without their guidance and persistent help this dissertation would not have been possible.

Furthermore, I would like to thank the German Academic Exchange Service DAAD for their financial support granted through a one year research scholarship. Besides the opportunity to spend another 10 months at the host university TU Dresden working within a larger group of researchers this enabled me to focus solely on my thesis and achieve the final goals of my research project.

Besides my advisors I thank my fellow researchers Christian Săcărea, Sanda Dragoş and Diana Haliţă for the stimulating discussions and for the sleepless nights we were working together before deadlines.

I would also like to thank the rest of my thesis committee: Prof. Leon Ţâmbulea, Lect. Darius Bufnea, Lect. Adrian Sterca, Conf. Rareş Boian, as well as other fellow researchers for their insightful comments, feedback and also for the questions which incented me to widen my research from various perspectives.

Last but not the least, I would like to thank my family for supporting me throughout writing this thesis and in my life in general.

Contents

1. Introduction

1.1 Motivation

Given that nowadays collecting and storing data is not so problematic as it used to be, there are large collections of data available for further analysis. The challenge, however, is to process this data and to extract useful information. There are a lot of real life situations where a better understanding of the available data can help us improve certain services or methodologies. For example, in the medical field, a lot of researchers try to study data available from patients in order to have a better insight about diseases, treatments and outcomes. Other systems which store large datasets are online platforms. One relatively new, but important, field in this category is online education. In the last years, the use of e-learning platforms has rapidly increased, not just in academia, but also for personal use (for example, the platform Coursera[1] had an immense growth in the number of courses offered since 2012 when it was founded). Analyzing the log data of such online learning platforms, one can try to understand the dynamics and the behavior of the users in order to enhance the platform structure and the learning experience of the students.

The World Wide Web nowadays can be seen as a huge information service center and browsing Web pages has become an essential aspect of our everyday lives. In order to improve the online experience by adapting the content and the design of Web pages, behavioral aspects of these processes have become more and more important, hence researchers developed tools for Web usage mining. However, when trying to apply these tools to an educational portal, we observe that special attention has to be given to these systems. Considering that most of the websites have a commercial purpose, the goals of the users visiting those sites are rather different from the goals of the users of an e-learning platform, whose behavior is focused solely on information acquisition. Driven by these practical requirements, we propose formal concept analysis as Web usage mining technique.

Formal concept analysis (FCA) provides a powerful mathematical tool that addresses

[1] `https://www.coursera.org/`

knowledge processing and knowledge representation ([Ganter and Wille, 1999]). The methods offered by FCA focus on representing, acquiring and retrieving knowledge and therefore serve the purpose of understanding and investigating datasets. The field of formal concept analysis has constantly developed in the last 30 years, one important point in its evolution being the extension to higher-dimensional datasets. FCA offers reasoning support for understanding large collections of information. Furthermore, it has the advantage that one can apply FCA techniques on a dataset without having extensive knowledge about the underlying reasoning support. This can be a big advantage in employing FCA as a method of knowledge processing used on a large scale. However, in order to achieve this, user-friendly tools are required which automate the FCA procedures and let users focus on idetified correlations and patterns in the processed data. Given this problem setting, we propose different visualization, navigation and exploration tools and analyze their usability in experimental settings.

We test these tools on real datasets, such as:

- the usage data of an e-learning portal called PULSE, used at Babeş-Bolyai University Cluj-Napoca,
- a medical database comprising information about several thousand cancer patients,
- the dblp database[2] which contains information about conference and journal publications such as: authors, journal or conference name, volume, year.

In our experiments, we find that the applicability of the proposed tools is not limited to educational portals, which were the starting point of our research, and the tools can be used for analyzing any dataset, even for data of dimensions higher than three (we analyze a tetradic dataset obtained from the dblp database). This makes the proposed tools valuable for organizing knowledge in a way which supports human thinking and decision making.

1.2 Problem Statement

When trying to formalize situations occurring in everyday life, it is easy to imagine them using binary relations, which is exactly the formalization exploited by *dyadic formal concept analysis*: there are *objects*, *attributes* and a binary *relation* between them that states whether an objects has or has not a certain attribute. There are a lot of problems that can

[2] `http://dblp.uni-trier.de/`

be modeled by this paradigm considering preprocessing techniques like attribute scaling in case of multivalued attributes.

However, sometimes it makes sense to consider a problem as "three-dimensional", using an additional set of *conditions*, or even study a dataset as an n-adic context with $n > 3$. For the *triadic* case, the (ternary) relation expresses if an object has an attribute under a certain condition. Typical examples of triadic contexts, often analyzed by researchers, are based on folksonomies, where we have users, resources and tags [Jäschke *et al.*, 2008; Trabelsi *et al.*, 2012].

There are a number of papers studying applications for the triadic case and also a few studying theoretical aspects of triadic or higher-adic formal concept analysis. However, we believe that there are certain theoretical aspects that are not yet fully defined and understood for the triadic case. One of the problems, that this thesis addresses, is that triadic formal concept analysis does not scale well when handling large datasets. In addition, the tools available for n-adic contexts, with $n > 2$, are limited to the functionality of computing the formal concepts and moreover, they usually require some knowledge of the theoretical aspects of formal contexts in order to use them. Most of these tools have a very large runtime already for medium datasets and are almost unusable for very large datasets.

We consider that besides computing the concept set, an even more important aspect is exploring this concept set using different graphical visualizations which offer navigation functionalities among the concepts. For the usability of such tools it is important to offer visualizations of the data in a human-readable format, which is easy to understand without extensive FCA knowledge. With this purpose, we propose in this thesis different methods of navigation and exploration for higher-adic datasets, all based on graphical representations of the data which offer a better understanding of the underlying structure and the existing correlations within the data. We discuss the scalability of the proposed paradigms and show that some of these approaches have the advantage of being applicable to any n-adic dataset.

In order to study the applicability of these tools, we analyze the computational complexity of the underlying problems as well as their time efficiency. Furthermore, we run experiments on multiple real datasets in order to test the different exploration strategies proposed by us. From the interpretation of the experimental results, we deduce that these exploration methods give more insight into the structure of the data by detecting patterns and correlations which are not visible in the raw data.

Obviously, the proposed navigation tools have certain limitations. These limitations together with possible solutions, which lead to new ideas for future work, are discussed in

the thesis. However, we believe that despite some limitations, the proposed exploration approaches can be successfully used as an interactive data mining technique and more importantly, they make data analysis techniques based on formal concept analysis more accessible to the users, regardless of their previous knowledge about FCA.

1.3 Thesis Focus and Key Contributions

This thesis offers an overview of polyadic formal concept analysis with the main focus on the triadic case. The dyadic case was extensively studied by other researchers and aspects such as understanding the underlying structure of the context, visualizing the context, navigating among concepts, reducing and clarifying the context, all have a good theoretical and practical basis for dyadic datasets. In our work, we try to extend these theoretical notions from the dyadic case to the triadic one and we study practical aspects and implementations for higher-adic cases. The only tools available so far for contexts of arity higher than three focus on computing formal concepts. However, there are no tools regarding reduction and visualization of a context, or navigation within the context. For that reason, we implement tools regarding all of this aspects and moreover, we analyze different techniques of computing formal concepts and compare them to available algorithms. For the tools we develop, we analyze besides their efficiency also their scalability and extend them, where possible, to the general case of n-adic contexts.

The main contributions of the thesis as well as where the results were published are detailed in the following:

Chapter 3.1: In this chapter, we analyze alternative visualization methods for a formal context in a circular layout using a tool called Circos (originally developed to visualize genomic data in bioinformatics). We then use formal concept analysis as an educational Web usage mining technique on the data of an e-learning platform called PULSE. In our analysis, we investigate temporal patterns of Web usage behavior within the system and identify three types of student behaviors: relaxed, normal and intense. We find correlations between these types of behavior and the timeline of the attended courses taking into consideration what different activities occured at each point in the timeline (task assignment, task deadlines, partial exam, final exam, etc.). Finally, the results obtained in the analysis are visualized with the previously mentioned circular layout. The techniques used for this analysis as well as experimental results were published in 2014 in two papers, at the International Conference on Knowledge Science, Engineering and Management [**Dragoş *et al.*, 2014b**] and in the journal Studia Universitatis Babeş-Bolyai [**Dragoş *et al.*, 2014a**].

Chapter 3.2: This chapter presents extensions of the theoretical notions of clarification and reduction of a formal context from the dyadic to the triadic case. Most of the time, these are processes that should occur in the preprocessing phase of an analysis of a dataset, depending on the final purpose of the analysis. Clarification and reduction serve the purpose of reducing the size of the dataset without altering its underlying structure. Intuitively, elements that can be clarified have identical behavior to some other elements, and elements that can be reduced have the common behavior of a group of elements (i.e., their behavior could be expressed as the "intersection" of the behaviors of all elements in the group). We deduce that the clarification and reduction processes eliminate redundant information and increase the efficiency of the algorithms applied in the next phases of the analysis, such as computing the structure of the context and visualize it or navigate among the formal concepts. The theoretical aspects of clarification and reduction as well as an application on a cancer registry database were published in 2015 at the International Workshop "What can FCA do for Artificial Intelligence" co-located with the International Joint Conference on Artificial Intelligence [**Rudolph *et al.*, 2015b**]. In the experiments described in our paper, we observe that in real datasets the number of elements that can be eliminated through the clarification and reduction process is often high, hence these preprocessing steps play an important role for speeding up the data analysis.

Chapter 3.3: In this chapter, we describe the first navigation paradigm proposed for triadic datasets based on reachability notions. Motivated by the fact that dyadic contexts have a lattice representation which enables navigation among the concepts, we propose a navigation method for triadic datasets that builds upon the lattice representations of appropriately defined dyadic projections. Intuitively, after choosing a concept and a so-called *perspective*, i.e. one of the three dimensions of the context, the user can visualize a subspace of concepts reachable through that perspective. The navigation is such that the user can choose a different perspective for each navigation step, hence being able to explore the structure of the dataset. We analyze the properties of the defined notion of reachability and the underlying structure of the context with respect to the reachability relation. We observe that there are so-called *clusters* of reachable concepts and moreover, from a starting point it is not always the case that every other concept is reachable. Understanding the properties of the clusters is a non-trivial task which gives rise to interesting theoretical questions which are discussed in our work. The theoretical and practical aspects of this navigation paradigm were published in 2015 at the International Conference on Formal Concept Analysis [**Rudolph *et al.*, 2015c**].

Chapter 3.4: The second navigation paradigm described in this chapter has a completely different approach and focuses on narrowing down the space of concepts according

to the user's interests by applying different constraints on the data. With this purpose, we define the notions of membership constraints and consider the corresponding satisfiability problem in a context, in order to determine if there is a concept that satisfies the given constraints. We analyze the computational complexity of the satisfiability problems for dyadic, triadic and finally n-adic datasets. Given that, in general, the membership constraint satisfiability problems turn out to be NP-complete it is not trivial to find efficient algorithms for computing them. For that purpose, we present a generic answer set programming (ASP) encoding of membership constraints. This encoding enables us to use ASP tools in the implementation, which are highly optimized to solve satisfiability problems, even when they are NP-complete. Next we describe an interactive navigation paradigm based on membership constraints which allows the user to gradually add constraints and navigate towards a single concept representing the potential goal of the navigation. These results were published in 2015 at the International Joint Conference on Artificial Intelligence [**Rudolph *et al.*, 2015a**]. As a next step of our work, we implement this navigation paradigm using different techniques. The first technique is based on the ASP encoding and has the advantage of scalability, while the second technique, which is based on an exhaustive search in the concept space, depends on an external tool for computing the formal concepts and therefore it is not always scalable. In order to ensure that the user always gets to a concept in the final state of the navigation, we introduce a *propagation* phase that adds other necessary constraints depending on the constraint added by the user. We analyze optimization methods for the propagation phase and, finally, run experiments in order to compare the two approaches. The detailed description of the navigation paradigm and the different techniques used in the implementation as well as the experimental results were described in a paper submitted in 2016 at the Workshop on Artificial Intelligence for Knowledge Management co-located with the International Joint Conference on Artificial Intelligence [**Rudolph *et al.*, 2016**].

1.4 Thesis Outline

The thesis has four main parts. The first introductory part contains the motivation of our research and a description of the problem that we are trying to solve. Moreover, the key contributions of the thesis are highlighted here with the corresponding papers in which the results were published.

The second part includes preliminaries and the state of the art. It starts by introducing formal concept analysis for the dyadic, triadic and n-adic case. Besides the theoretical aspects, for each of the cases, existing tools and algorithms are presented. In this chapter,

one can see that some theoretical results are available only for dyadic FCA (hence the need for extension). The next two chapters discuss some notions of complexity theory and answer set programming, which are necessary in order to understand the following chapters.

The third part, called "Visualization, Navigation and Exploration in Triadic Datasets" is the main part of the thesis comprising all the theoretical and practical results obtained in our research. It starts with Chapter 3.1 which has a more practical approach while trying to solve the first problem that we address, namely visualization of triadic datasets. Here, we introduce Web usage mining and Web analytics metrics and explain why most Web usage mining techniques do not work as expected on e-learning systems. Then we use formal concept analysis as a Web usage mining technique and analyze the log data of an e-learning portal called PULSE. We offer a detailed description of the analysis which includes three phases: preprocessing, pattern discovery and pattern analysis. Finally, we visualize the experimental results in a circular layout using a tool called `Circos` and interpret these results in order to identify temporal patterns of Web usage behavior.

In Chapter 3.2, the notions of clarification and reduction are extended from the dyadic to the triadic setting. After formally defining the two processes for triadic contexts, we run experiments on a cancer registry database. Herefrom we conclude that clarification and reduction are an important part of the preprocessing phase of any data analysis, since they can drastically reduce the size of a dataset without altering its underlying structure, i.e. without changing the results of the analysis.

Chapter 3.3 describes the first navigation paradigm that we propose for triadic datasets. We motivate the chosen method of navigation and before introducing the theoretical aspects, we present a proof of concept by navigating within a small triadic dataset. Next we introduce the notions of reachability relation among triconcepts and reachability cluster, and study properties arising from these notions. Finally, we describe an exploration strategy based on the reachability relation, which uses conveniently chosen dyadic projections in order to take advantage of the navigation strategies of a dyadic context.

Chapter 3.4 contains a second navigation paradigm based on a different approach. This approach focuses on narrowing down the space of concepts based on constraints specified by the user. For this purpose we introduce the theoretical aspects of membership constraints for the dyadic, triadic and n-adic case. Moreover, we discuss the computational complexity of the satisfiability problems of membership constraints. Given that the satisfiability problem proves to be NP-complete in general, we present an encoding for membership constraints in answer set programming, which will enable us to use highly optimized ASP tools. In the next chapter we discuss possible implementations and opti-

mizations for an exploration and navigation tool based on membership constraints. The two different tools described here are `ASP Navigation Tool`, based on the ASP encoding, and `Brute Force Navigation Tool`, based on an exhaustive search in the concept space, which was precomputed with an external FCA tool. In the last chapter we evaluate and compare the two navigation paradigms as well as the different implementation strategies.

The final part of the thesis highlights the achievements of our research and discusses open issues and plans for future work.

2. Preliminaries and State of the Art

2.1 Formal Concept Analysis

2.1.1 History of Formal Concept Analysis

Formal concept analysis (FCA) was introduced by Bernhard Ganter, Rudolf Wille and Peter Burmeister in the early 1980s. The theory has its mathematical basis in general lattice theory created by Garrett Birkhoff in the 1930s and published a few years later [Birkhoff, 1940].

The main advantage of formal concept analysis is the fact that the FCA tools do not require extensive knowledge on lattice theory. Furthermore, FCA is perfectly suitable for information retrieval, since elements satisfying some properties can be expressed in the FCA formalization as a set of objects having some attributes in common.

In 1995, Fritz Lehman and Rudolf Wille extended formal concept analysis to the triadic case. They added a third dimension, changing the way of modelling a formal concept analysis problem as follows: *objects* have *attributes* under certain *conditions*. However, because of its higher complexity, there was little focus on the triadic case.

Considering the limited attention the triadic case received, it is not surprising that there are even less results for further generalizations of formal concept analysis. For the n-adic case, there are some theoretical aspects that have been studied by Voutsadakis [2002], but there are few applications for dimensions higher than 3 [Jelassi *et al.*, 2012; Cerf *et al.*, 2009; Cerf *et al.*, 2013]. The reason behind this is not just the high complexity, but also the lack of appropriate datasets. The datasets available have usually only been interpreted as bidimensional or tridimensional contexts. Nevertheless, it is interesting to understand the aspects of n-adic formal concept analysis even on a theoretical level.

2.1.2 Dyadic Formal Concept Analysis

2.1.2.1 Theoretical Foundations

This chapter aims to present an overview of dyadic formal concept analysis, in order to better understand the extensions to the triadic as well as to the general n-adic case. For a deeper understanding of the theoretical foundations please refer to the references [Ganter and Wille, 1999].

The fundamental structures used by formal concept analysis are those of a *formal context* and a *formal concept*, notions often simply referred to as context and concept.

Definition 2.1.1 *A (dyadic)* **formal context** $\mathbb{K} = (G, M, I)$ *is defined as a triple consisting of two sets and a binary relation* $I \subseteq G \times M$ *between the two sets.* G *represents the set of* **objects**, M *the set of* **attributes** *and* I *is called the* **incidence relation**. *The notation for an element of the incidence relation is* gIm *or* $(g, m) \in I$ *and it is read* **object g has attribute m**.

Formal contexts can be represented as cross tables, the rows of which are objects, the columns attributes and the incidence relation is represented by crosses in the table. Therefore, if object g has attribute m, there will be a cross in the cell corresponding to row g and column m. Although this is a simple and easy to understand representation, it is only useful for small contexts. In the case of large contexts it becomes impossible to read the incidence relation or to extract useful information only by looking at the table.

Example 2.1.1 *The following is an example of a cross table representation for the planets context. The objects are Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune and the attributes are properties related to the size, distance from the sun and whether they have satellites or not.*

planets	*dist near*	*dist far*	*size small*	*size medium*	*size large*	*satellite yes*	*satellite no*
Mercury	×		×				×
Venus	×		×				×
Earth	×		×			×	
Mars	×		×			×	
Jupiter		×			×	×	
Saturn		×			×	×	
Uranus		×		×		×	
Neptun		×		×		×	

Figure 2.1: Planets context

In order to define the notion of a formal concept, the derivation operators have to be introduced first.

Definition 2.1.2 *We define the derivation operator for the object set G and the attribute set M by:*

$$A' = \{m \in M \mid gIm, \forall g \in A\} \text{ for } A \subseteq G, \text{ and}$$
$$B' = \{g \in G \mid gIm, \forall m \in B\} \text{ for } B \subseteq M.$$

For an element $g \in G$ or $m \in M$, instead of writing $\{g\}'$ and $\{m\}'$, often the notations g' and m' are used.

Based on these derivation operators, the notion of formal concept is introduced.

Definition 2.1.3 *If $\mathbb{K} = (G, M, I)$ is a formal context, then a (dyadic)* **formal concept** *is defined as a pair (A, B), with $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$. A is called* **extent** *and B* **intent** *of the concept. The set of all concepts of the context (G, M, I) is denoted by $\mathfrak{B}(G, M, I)$ or $\mathfrak{B}(\mathbb{K})$.*

Definition 2.1.4 (Object concept and attribute concept) *Let $\mathbb{K} = (G, M, I)$ be a formal context, $g \in G$ an object and $m \in M$ an attribute. Then, the formal concept (g'', g') is called an* **object concept** *and it is denoted by γg, while the formal concept (m', m'') is called an* **attribute concept** *and it is denoted by μm.*

The context in Figure 2.1 has the following concepts:

- $(\emptyset, \{near, far, small, medium, large, yes, no\})$
- $(\{Jupiter, Saturn\}, \{large, far, yes\})$
- $(\{Uranus, Neptun\}, \{medium, far, yes\})$
- $(\{Earth, Mars\}, \{yes, near, small\})$
- $(\{Mercury, Venus\}, \{no, near, small\})$
- $(\{Jupiter, Saturn, Uranus, Neptun\}, \{far, yes\})$
- $(\{Earth, Mars, Mercury, Venus\}, \{near, small\})$
- $(\{Jupiter, Saturn, Uranus, Neptun, Earth, Mars\}, \{yes\})$
- $(\{Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune\}, \emptyset)$

The following proposition offers a better understanding of the rules of derivation.

Proposition 2.1.1 *Let (G, M, I) be a context and $A_1, A_2, A_3 \subseteq G$. Then the following hold:*

- $A_1 \subseteq A_2 \Rightarrow A_2' \subseteq A_1'$
- $A_3 \subseteq A_3''$
- $A_3' = A_3'''$

Furthermore, the same correspondences apply for subsets of attributes.

Since formal concept analysis is based on order theory, the notion of closure system, closure operator and complete lattice have to be introduced next.

Definition 2.1.5 (Closure system) *A closure system on a set G is a set of subsets that contains G and is closed under intersection.*

Definition 2.1.6 (Closure operator) *A closure opeator φ on G is a map that has the following properties for $\forall X, Y \subseteq G$:*

- *assigns a closure $\varphi X \subseteq G$ to X*
- *monotone: $X \subseteq Y \Rightarrow \varphi X \subseteq \varphi Y$*
- *extensive: $X \subseteq \varphi X$*
- *idempotent: $\varphi\varphi X = \varphi X$*

The set $\{\varphi(X) \mid X \subseteq G\}$ is called the set of closures of φ.

Before introducing the notion of a complete lattice, we define a few notions from order theory.

Definition 2.1.7 (Infimum and supremum) *Let $(B, \leq)$ be an ordered set and $A \subseteq B$. We call an element $l \in A$ a* **lower bound** *of A if for $\forall a \in A$ we have that $l \leq a$. Similarly, an element $u \in A$ is an* **upper bound** *of A if for $\forall a \in A$ we have that $l \geq a$. Furthermore, the* **infimum** *of A is defined as the largest element of the set of all lower bounds of A and is denoted by $\inf A$ or $\bigwedge A$. If we talk about the infimum of two elements x and y, i.e. $A = \{x, y\}$, the notations $\inf(x, y)$ and $x \wedge y$ are also used. Analogously, the* **supremum** *of A is defined as the smallest element of the set of all upper bounds of A and is denoted by $\sup A$ or $\bigvee A$. For the supremum of two elements x and y we use the notations $\sup(x, y)$ or $x \vee y$. It is notable that the infimum and supremum do not always exist.*

Definition 2.1.8 (Complete lattice) *An ordered set* $\mathbf{V} = (V, \leq)$ *is a lattice, if for any two elements* $x, y \in V$ *the supremum* $\sup(x, y)$ *and the infimum* $\inf(x, y)$ *always exist.* V *is called a complete lattice, if for any subset* $X \subseteq V$*, the supremum* $\bigvee X$ *and the infimum* $\bigwedge X$ *exist.*

Observation 2.1.1 *For any closure operator, the set of all closures is a closure system. Any closure system is an ordered set by set inclusion* $\subseteq$ *and it is actually a complete lattice.*

Next we can define the notions of $\bigwedge$*-irreducible* and $\bigvee$*-irreducible* in a complete lattice as follows [Ganter and Wille, 1999].

Definition 2.1.9 ($\bigwedge$-irreducible and $\bigvee$-irreducible) *Let* $\mathbf{V}$ *be a complete lattive and* $v \in V$ *an element of the lattice for which we define the following:*

$$v_* = \bigvee \{x \in V \mid x < v\}$$

$$v^* = \bigwedge \{x \in V \mid v < x\}$$

We say that the element v *is* **$\bigvee$-irreducible** *(read* **supremum-irreducible***) if* $v \neq v_*$*, i.e. if* v *cannot be represented as the supremum of strictly smaller elements. Analogously,* v *is* **$\bigwedge$-irreducible** *(read* **infimum-irreducible***) if* $v \neq v^*$*, i.e. if* v *cannot be represented as the infimum of strictly larger elements.*

The set of concepts of a formal context can be ordered by a subconcept-superconcept relation as follows:

Definition 2.1.10 *Let* (A_1, B_1) *and* (A_2, B_2) *be two concepts of a context* $\mathbb{K}$*.* (A_1, B_1) *is a* **subconcept** *of* (A_2, B_2) *if* $A_1 \subseteq A_2$*. In this case,* (A_2, B_2) *is called a* **superconcept** *of* (A_1, B_1)*. The notation is* $(A_1, B_1) \leq (A_2, B_2)$*. The set of all concepts with this order relation,* $(\mathfrak{B}(\mathbb{K}), \leq)$*, is a complete lattice, called the* **concept lattice** *of* $\mathbb{K}$*.*

In the subconcept-superconcept relation there is a "smallest" and a "biggest" element. These are, usually, the trivial concepts that have either the extent or the intent equal to the empty set. Hence, for the context (G, M, I), the concept $(\emptyset, M)$ would correspond to the node at the bottom of the concept lattice and the concept $(G, \emptyset)$ to the node at the top of the lattice. However, it can be the case that the minimal concept does not have an empty object set if there is at least one object that is in relation to all the attributes. The same applies to the maximal concept, that can have a non-empty attribute set. Sometimes only non-trivial concepts, i.e. concepts having non-empty components, are

taken into consideration when analyzing a formal context, excluding the trivial ones which are artificial and don't reveal any information about the data.

The fact that the concept lattice is a complete lattice is actually proven by the basic theorem on concept lattices [Ganter and Wille, 1999]. Another consequence of the basic theorem is that the set of extents, respectively the set of intents are closure systems on the object set, respectively on the attribute set. The corresponding closure operator for each of the closure systems is the double derivation $X \mapsto X''$.

The concept lattice of the previously given example (Figure 2.1), is represented in Figure 2.2.

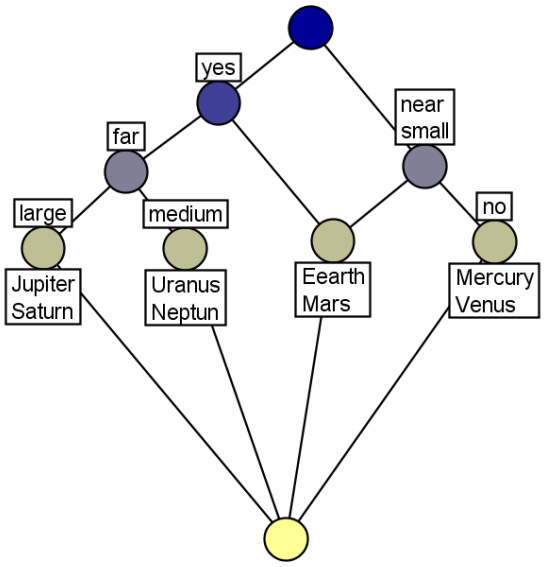


Figure 2.2: Lattice for the planets context

Intuitively, a concept can be understood as a set of objects with a common set of attributes, that satisfies the condition of maximality (i.e. no other object or attribute can be added to the extent, respectively to the intent of the concept without violating the first property). This interpretation of the formal concepts makes formal concept analysis a useful tool for information retrieval, since in a dataset, we usually look for a set of objects that have certain properties, i.e. attributes.

All the concepts can be read from the concept lattice in the following way. Each concept corresponds to exactly one node in the lattice. When looking at a node, the extent of the corresponding concept contains all the objects from the lattice reachable when going (only) downward. Analogously, the intent contains all the attributes reachable when going (only) upward.

Because the lattice is based on an order relation, navigating in a lattice structure

is easy. Consider for example the concept $(\{Jupiter, Saturn\}, \{large, far, yes\})$ as a starting point. When navigating upwards, we get to the following concept: $(\{Jupiter, Saturn, Uranus, Neptun\}, \{far, yes\})$. Hence, in this navigation step two new objects were added to the extent, but one attribute was removed from the intent, which can be easily deduced from the concept lattice. Similarly, when navigating downwards the extent gets smaller and the intent larger.

However, for larger contexts, the concept lattice becomes hard to read because of the numerous labels that need to be attached to the nodes and impossible to use as a visualization tool. Therefore, the possibility of reducing the dimensions of the context, i.e. the number of objects and attributes, was studied. There are two operations that can reduce the size of the context without altering its underlying structure: clarification and reduction. Clarification is based on the idea that objects with the same intent can be merged into one element of the same type, since they have the same behavior. The same applies for attributes with the same extent.

Definition 2.1.11 *A dyadic context (G, M, I) is* **clarified** *if for any objects $g, h \in G$, from $g' = h'$ follows $g = h$, and for all attributes $m, n \in M$, $m' = n'$ implies $m = n$.*

Observation 2.1.2 *Intuitively, when examining the cross table representation of the context, elements that can be clarified are objects corresponding to equal lines and attributes corresponding to equal columns. The same properties can be observed in the concept lattice of the context: if two elements of the same type (objects or attributes) correspond to the same node in the concept lattice than they can be merged in the clarification process without altering the structure of the concept lattice.*

The second operation, reduction, takes advantage of the fact that objects and attributes that can be written as combinations of other objects, respectively attributes, can be eliminated without having an influence on the conceptual structure.

Definition 2.1.12 *A clarified context (G, M, I) is called* **row reduced** *if every object concept is $\vee$-irreducible and* **column reduced** *if every attribute concept is $\wedge$-irreducible. The context is called* **reduced** *if it is both row reduced and column reduced.*

Proposition 2.1.2 *Consider a clarified context (G, M, I) and two elements $g \in G$, $m \in M$. Then we have the following characterizations of reducible object, respectively attribute:*

- *object g is reducible if and only if there exists a subset $X \subseteq G$ with $g \notin X$, s.t. $g' = X'$*

- *attribute m is reducible if and only if there exists a subset $Y \subseteq M$ with $m \notin Y$, s.t. $m' = Y'$*

Proof. The proof follows from the fact that, in case that $g' = X'$, then the object concept γg becomes the supremum of all object concepts $\gamma x, x \in X$. Similarly, the attribute concept μm becomes the infimum of all attribute concepts $\mu y, y \in Y$, when $m' = Y'$ [Ganter and Wille, 1999]. □

Observation 2.1.3 *Intuitively, an object that can be reduced corresponds to a line that can be expressed as a combination of other lines as their intersection. Analogously, an attribute that can be reduced corresponds to a column that can be expressed as a combination of other columns. However, the elements that can be eliminated in the reduction process are not immediately observable in the concept lattice.*

The planets context from Example 2.1.1 is already clarified and reduced. The following example, although being an artificially generated context, will illustrate the procedure of clarification and reduction.

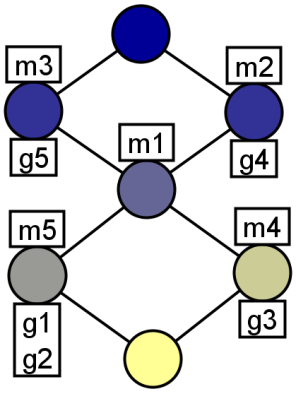
Example 2.1.2 $\mathbb{K} = (G, M, I), G = \{g_1, g_2, g_3, g_4, g_5\}, M = \{m_1, m_2, m_3, m_4, m_5\}$

	m_1	m_2	m_3	m_4	m_5
g_1	×	×	×		×
g_2	×	×	×		×
g_3	×	×	×	×	
g_4		×			
g_5			×		

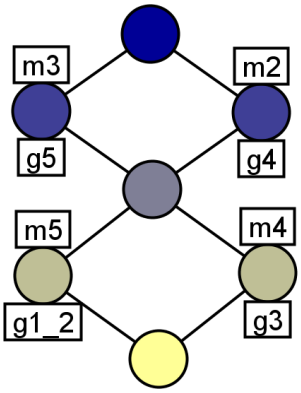
The concept lattice of the context in Example 2.1.2 in represented in Figure 2.3a. According to Observation 2.1.2 objects g_1 and g_2 can be merged into one object in the clarification process, named g_{1_2} in this example. Also, attribute m_1 can be expressed as the combination of attributes m_2 and m_3, so it will be eliminated in the reduction process.

Ganter and Wille define another method for the reduction of a clarified context, by means of arrow relations. These are formally described by the following defition [Ganter and Wille, 1999].

Figure 2.3: Concept lattices before and after reduction



(a) Concept lattice of the original context



(b) Concept lattice of the reduced context

Definition 2.1.13 *Let (G, M, I) be a context, $g, h \in G$ objects, and $m, n \in M$ attributes. Then, we define the following relations:*

$$g \swarrow m \Leftrightarrow \begin{cases} (g,m) \notin I \\ if\ g' \subset h',\ then\ (h,m) \in I \end{cases}$$

$$g \nearrow m \Leftrightarrow \begin{cases} (g,m) \notin I \\ if\ m' \subset n',\ then\ (g,n) \in I \end{cases}$$

$$g \swarrow\!\!\!\!\nearrow m \Leftrightarrow g \swarrow m \text{ and } g \nearrow m$$

Intuitively, the relation $g \swarrow m$ means that the derivation g' is maximal among all object intents which do not contain attribute m. The correlation between the arrow relations and reduction in a context is shown in the following proposition.

Proposition 2.1.3 *Let (G, M, I) be a context, $g \in G$ an object, and $m \in M$ an attribute. Then, we have that:*

- *the object concept γg is $\vee$-irreducible if and only if there is an attribute $n \in M$ with $g \swarrow n$. Furthermore, in a finite context, this is also equivalent to the fact that there is an attribute $n_1 \in M$ with $g \swarrow\!\!\!\!\nearrow n_1$.*
- *the attribute concept μm is $\wedge$-irreducible if and only if there is an object $h \in G$ with $h \nearrow m$. Furthermore, in a finite context, this is also equivalent to the fact that there is an object $h_1 \in G$ with $h_1 \swarrow\!\!\!\!\nearrow m$.*

Observation 2.1.4 *From the definition, we can deduce that the arrow relations only hold between pairs of objects and attributes which do not belong to the incidence relation I. Hence, in the cross table representation of the context, we can add the arrows where the corresponding relation holds, since it will not overlap with a cell where we already have a cross (meaning the pair belongs to the incidence relation). After adding the corresponding arrows in the table, in order to reduce the context we delete all rows and columns that do not contain a double arrow ↙↗.*

Example 2.1.3 *For example, the clarified context from the previous Example 2.1.2 and the corresponding arrow relations can be seen in Figure 2.4. We can observe that the only row or column that do not contain a double arrow is the column corresponding to attribute m_1, which as we have also seen in Example 2.1.2, can be reduced.*

	m_1	m_2	m_3	m_4	m_5
g_{1_2}	×	×	×	↙↗	×
g_3	×	×	×	×	↙↗
g_4	↙	×	↙↗		
g_5	↙	↙↗	×		

Figure 2.4: Clarified context with arrow relations

Despite the clarification and reduction processes, concept lattices can still grow exponentially in the size of the context. One method of addressing this problem is reducing the complexity of the diagram using conceptual scaling. In order to reduce the complexity, one can consider only a subset of attributes at a time. Furthermore, conceptual scaling also gives a solution for dealing with many-valued attributes.

When modeling a problem in the form of objects having attributes, there is often the case that attributes are not Boolean, but may have different values. In that case, the context is called a multi-valued context. Formally, a multi-valued context, also called many-valued context, is defined as follows:

Definition 2.1.14 *A multi-valued context (G, M, W, I) contains besides the set of objects G and the set of attributes M, a set W of attribute values. For the ternary relation I holds that $(g, m, w) \in I$ and $(g, m, v) \in I \Rightarrow w = v$. An element from the relation $(g, m, w) \in I$ is read as "attribute m has value w for object g". This justifies to write $m(g) = w$ for $(g, m, w) \in I$.*

In order to transform a multi-valued context into a one-valued context it has to be transformed by a process called conceptual scaling [Ganter and Wille, 1999]. Conceptual scaling involves a human expert that knows how to interpret the data, since each attribute has to be interpreted using a conceptual scale.

Definition 2.1.15 *A conceptual scale for a multivalued attribute is a one-valued context* (G_m, M_m, I_m) *with* $m(G) \subseteq G_m$.

Intuitively, a conceptual scale is just a formal context that interprets a multi-valued attribute. An example of a conceptual scale for an attribute that can have the following values $\{verypoor, poor, good, excellent\}$ is:

	excellent	good	poor	very poor
excellent	×	×		
good		×		
poor			×	
very poor			×	×

Using this scale, an attribute from a multi-valued context can be transformed into the derived one-valued context as seen in Figure 2.5.

Figure 2.5: Conceptual scaling

(a) multi-valued context

attribute1
good
excellent
good
poor
very poor
excellent

(b) one-valued context

attribute1			
excellent	good	poor	very poor
	×		
×	×		
	×		
		×	
		×	×
×	×		

Intuitively, in the derived context every multi-valued attribute is replaced by the scale attributes, while the objects remain unchanged. Formally, any context can be a scale, but the most frequently used scales are the elementary scales introduced in what follows.

For the scales definition the following abbreviation is used: $\mathbf{n} = \{1, ..., n\}$

- **Nominal scales** $N_n = (\mathbf{n}, \mathbf{n}, =)$

  Nominal scales are used for scaling attributes, whose values are mutually exclusive, like for example gender. Having attributes that exclude each other, the concept extents build a partition of the object set. The nominal scale for such an attribute with 4 values is represented in Figure 2.6

	1	2	3	4
1	×			
2		×		
3			×	
4				×

Figure 2.6: Nominal scale N_4

- **Ordinal scale** $O_n = (\mathbf{n}, \mathbf{n}, \leq)$

  Ordinal scales are used for scaling attributes with ordered values, having the property that each value implies the smaller value, for example *expensive, very expensive, extremely expensive*. The concept extents form a ranking.

	1	2	3	4
1	×	×	×	×
2		×	×	×
3			×	×
4				×

Figure 2.7: Ordinal scale O_4

- **Interordinal scale** $I_n = (\mathbf{n}.\mathbf{n}, \leq) \mid (\mathbf{n}.\mathbf{n}, \geq)$

  In the case of the interordinal scale, the concept intents are the intervals of scale values.

	≤ 1	≤ 2	≤ 3	≤ 4	≥ 1	≥ 2	≥ 3	≥ 4
1	×	×	×	×	×			
2		×	×	×	×	×		
3			×	×	×	×	×	
4				×	×	×	×	×

Figure 2.8: Interordinal scale I_4

- **Biordinal scale** $M_{n,m} = (\mathbf{n}, \mathbf{n}, \leq) \cup (\mathbf{m}, \mathbf{m}, \geq)$

  Sometimes attributes with ordered values do not imply all the other smaller value. In that case, usually a biordinal scale can be used, so that attributes are assigned to one of two poles. For example, an attribute could have as values *very cheap, cheap, expensive, very expensive, extremely expensive*. A biordinal scale is represented in Figure 2.9.

	≤ 1	≤ 2	≤ 3	≤ 4	≥ 5	≥ 6
1	×	×	×	×		
2		×	×	×		
3			×	×		
4				×		
5					×	
6					×	×

Figure 2.9: Biordinal scale $M_{4,2}$

- **Dichotomic scale** $D = (\{0, 1\}, \{0, 1\}, =)$

  The dichotomic scale is a special case that is used for attributes that have values similar to *yes/no*

Conceptual scaling is usually a necessity in the preprocessing part, where the context is built, because it is often the case that attributes have multiple values. Therefore, a lot of the examples and experiments there will be presented in the thesis will be derived contexts that have undergone the process of conceptual scaling. Obviously, the obtained derived context is not unique, since it is a matter of interpretation and the transformation depends on the chosen scale.

2.1.2.2 Existing Tools and Algorithms

There are several algorithms and tools for computing formal concepts and concept lattices for a given dyadic context. The difficulty usually encountered in computing the concepts is that different sets can have the same closure. Therefore, when naively computing all the closures, one would have to check multiple times if a computed closure already exists in the output or not. This would result in an exponential number of lookups and would increase the complexity of the algorithm. Therefore, the main difficulty remains to generate all concepts and, if possible, avoid repetitive generation. Some algorithms solve this problem by generating concepts in a specific order or according to some rules that

ensure the uniqueness of a concept and others find efficient methods to check whether the concept had already been generated or not.

One of the first and most popular algorithms that compute the concepts of a dyadic context (but not the concept lattice) is `Next-Closure` [Ganter, 1984], an algorithm that computes concepts sequentially. Its general, but simple form allows one to compute all the closed sets for a chosen closure operator on a finite set. Furthermore, it allows many useful modifications, which widen its field of application [Borchmann, 2012]. The `Next-Closure` algorithm computes the formal concepts' intents in lectic order. Intuitively, for a set $M = \{1, ..., n\}$ we say that $A \subseteq M$ is lectically smaller than $B \subseteq M$ if the two sets are not equal and the smallest element with respect to some predefined order in which they differ is included in B. In addition to the small time complexity of the `Next-Closure` algorithm, another advantage is the linear space complexity.

However, there are cases where the concept lattice is too large to be represented in a readable format. Particularly, in the worst case the size of the lattice grows exponentially with the size of the context. One solution to this problem is to graphically represent only frequent formal concepts. Frequent attribute sets are measured by comparing the cardinality of the corresponding object set to a given minimum support value. Under these conditions, a concept is considered frequent if its intent is frequent. The notion of iceberg concept lattice is defined as the set of all frequent formal concepts. Intuitively, the iceberg concept lattice only shows the top part of the original concept lattice. Such iceberg concept lattices are particularly useful for strongly correlated data. These type of lattices can be computed by a modified version of the `Next-Closure` algorithm or by the algorithm `Titanic`.

`Titanic` [Stumme *et al.*, 2002] is another efficient algorithm for computing concept lattices that has a completely different approach than other existing algorithms. In its general form, the algorithm computes closure systems for a closure operator with an associated weight function. Although this algorithm can be used to compute the general concept lattice, it has a different, improved version for computing the iceberg concept lattice. `Titanic` tries to optimize the computation by addressing three issues:

- computing the closure of attribute sets using only support values
- obtaining the closure system by computing as few closures as possible
- deriving support values from already known support values, where possible

Compared to the `Next-Closure` algorithm, `Titanic` is more performant, especially when applied on large datasets.

Two algorithms that are similar to the `Next-Closure` algorithm but more efficient, are `Close-By-One`, also known as `CbO` [Kuznetsov, 1999] and `FCbO` [Krajca *et al.*, 2008; Krajca *et al.*, 2010; Outrata and Vychodil, 2012], an extension of `CbO`. `CbO` computes all concepts, as well as the concept lattice in linear time with respect to the number of concepts and it is based on a depth-first search strategy. Krajca, Outrata and Vychodil proposed in 2010 the algorithm `FCbO` as an extension to Kuznetsov's algorithm `CbO`. They tried to improve the efficiency of the algorithm first by reducing the number of concepts computed multiple times and second, by suggesting preprocessing methods that would increase the speed of the concepts' computation. In this extension, they combined two search strategies: depth-first and breadth-first search. Furthermore, it turns out that there are algorithms, like `Next-Closure`, `CbO`, `FCbO` that benefit from a well-chosen ordering of the attributes in the preprocessing phase. This reduces the number of formal concepts that are computed multiple times and hence improves the performance of the algorithms.

`In-Close` proposed by Andrews [2009] is another algorithm for enumerating concepts that uses incremental closure. Conceptually, `In-Close` is also based on the `CbO` algorithm. An advantage of `In-Close` is that it requires no preprocessing. Andrews compares `In-Close` to the predecessor of `FCbO` [Krajca *et al.*, 2008], which is already more efficient than the `Next-Closure` algorithm. He proves in some experiments that `In-Close` outperforms Krajca's algorithms. However, the authors improve the efficiency of their alogrithm in `FCbO` [Krajca *et al.*, 2010] by suggesting a new canonicity test, so that it outperforms `In-Close`, as well as other algorithms proposed by Lindig [2000] or Berry, Bordat and Sigayret [2007]. Despite a relatively high time efficiency, `In-Close` had the disadvantage of using exponential memory for generating concepts. An improved version `In-Close2` was proposed in 2011 [Andrews], which adds new optimizations and data preprocessing techniques to the previous version. Although the purpose of `In-Close2` was to outperform `FCbO`, some argue that the latter still remained a more efficient algorithm [Pisková and Horváth, 2013]. Still, Andrews argues that for some of the experiments, `In-Close2` outperformed `FCbO`, but for other `FCbO` was more efficient [Andrews, 2011].

Another algorithm whose efficiency is comparable to `FCbO` and `In-Close` is `AddIntent`, which appeared earlier under the name of `AddAtom`. `AddIntent` was first implemented in 1996 and was improved in a later version by Merwe, Obiedkov and Kourie [2004]. It is an incremental algorithm that computes not only the formal concepts, but also the lattice structure. `AddIntent` uses the ordering of subconcept-superconcept unlike other algorithms, hence being able to construct the relations between the concepts. Normally one cannot have a fair comparing scale between an algorithm that computes the concept set and the concept lattice and one that computes solely the concept set. Still, `AddIntent`

outperforms some of the existent algorithms that compute only the concept set. The authors implemented also an attribute-incremental version of `AddIntent`, which is called `AddExtent`. This version was compared to `In-Close2` and `FCbO` and each of the three algorithms outperforms the other two on some datasets [Andrews, 2011]. Obviously, if one needs the concept lattice, the use of `AddIntent` is preferred. Otherwise, the other two algorithms might prove to have better results.

There are a lot of other algorithms proposed for FCA that try to improve at least one aspect of another algorithm. For example, Nourine and Raynaud proposed an algorithm that has a better worst-case complexity compared to the `Next-Closure` algorithm, but it needs exponential space, since it stores the whole lattice [Nourine and Raynaud, 1999]. Krajca, Outrata and Vychodil proposed also parallel versions of `CbO` and `FCbO`, called `PCbO` [Krajca *et al.*, 2008] and `PFCbO` [Krajca *et al.*, 2010] that have different performances depending on the method chosen for splitting the data for parallelization. Furthermore, there are also a lot of algorithms that generate closed itemsets that can be adapted to the FCA needs, namely to compute formal concepts. Different methods and approaches of algorithms are the topic of several papers that try to compare algorithms, both theoretically (in the worst case scenario), as well as experimentally [Godin *et al.*, 1995; Kuznetsov and Obiedkov, 2002; Pisková and Horváth, 2013]. According to a more recent comparison done by Pisková and Horváth [2013], `FCbO` is to date one of the most efficient algorithms for computing the formal concept set. Still, there are datasets on which other algorithms perform better. In general, the choice of the algorithm depends on the dataset and particularly on its properties such as density and size. Another aspect that should be taken into consideration is the comparison of the output of several algorithms, since it was shown that some algorithms fail to compute the concepts correctly for specific datasets [Pisková and Horváth, 2013].

There are several tools that implement the previously mentioned or new algorithms for formal concept computation and/or lattice computation. A few of the most popular tools are described in the next paragraphs.

The `ToscanaJ Suite` [Becker *et al.*, 2002; Becker and Correia, 2005] is probably one of the most comprehensive and popular project that relies on Ganter's algorithm, `NextClosure`. It includes tools to create a formal context, perform conceptual scaling on a context, compute the formal concepts and the concept lattice. In addition to automating the process of concept lattice construction and formal concept computation, `ToscanaJ`'s goal was to give users the possibility to use formal concept analysis methods without having a mathematical background. `ToscanaJ` has several components: `Elba`, `Siena` and `Toscana`. The first two components are responsible for creating and modifying

the conceptual schema, and the third component for reading it.

The tools for creating the conceptual scheme differ in the input type. Therefore, `Elba` uses a relational database for the creation of the conceptual scheme, while `Siena` supports input and management of the data without any connection to a database. However, `ToscanaJ` can offer more functionalities when used with a relational database system. After connecting to a database, `Elba` retrieves the information about available tables in the database and the user can choose which tables to use for the data analysis, while in `Siena`, one can define a context by introducing the object and attribute set and defining the relation as a cross table.

Before creating the conceptual schema, `ToscanaJ` also offers the possibility of using conceptual scaling. Therefore, when analyzing multi-valued datasets, the user can select a scale, filter a certain subset of elements or even use nested scales. After building the conceptual scale, the concept lattice can be visualized. There are also a few different options available for the lattice labeling. Therefore, nodes representing formal concepts, can be labeled by one of the following:

- the number of objects in the set
- a list of the items in the set
- the percentual distribution with respect to the current object set in the diagram

Furthermore, in the graph visualization, `ToscanaJ` represents the size of the extent through a color gradient of the nodes, which offers more insight in the elements' distribution. In addition to the basic functionalities described in the previous paragraphs, `ToscanaJ` offers other advanced functionalities and integrates with several data formats. Hence, it is a comprehensive and largely used tool suite for formal concept analysis.

Another tool that computes the formal concepts as well as the concept lattice is `ConExp Explorer`, also called `ConExp`, which is based on a new algorithm, `Grail`, proposed by Yevtushenko [2000]. `ConExp` supports context processing, including clarification and reduction of the context and the typical FCA operations: computing the concept set and the conceptual diagram. Furthermore, `ConExp` offers more advanced functionalities, such as calculating the base of implications and association rules of the formal context and performing attribute exploration. Similar to `ToscanaJ`, this tool also offers different layouts for the lattice visualization.

The two tools described previously, are mainly used for performing FCA operations on an already existent formal context. The only integrated method to build a context from existing data, being the use of `Elba` which can connect to a database. However,

often available data is in different data formats and it is not trivial to transform it into a formal context. The `FcaBedrock` tool is a context creator that can process large data [Andrews and Orphanides, 2010]. It can process `csv` files and before creating the context, it can convert many-valued attributes to formal attributes.

Another tool supporting the use of formal concept analysis and its interoperability with other tools is `FcaStone` proposed by Priss [2008]. The purpose of this tool is to convert files between different formats used either by FCA tools, such as `ToscanaJ`, `ConExp` or by other graph visualization tools. Furthermore, `FcaStone` has also integrated a simple, though not efficient, implementation of Ganter's algorithm, so that it can convert a context file to a lattice diagram.

All the mentioned algorithms and tools are supporting one or a more of the aspects of the lattice life-cycle. However in order to get from a raw dataset to the concept lattice, one must use more tools. That was the motivation of the `Galicia` project that aims to support the entire process from data preprocessing and lattice construction to the visualization and navigation in the concept lattice [Valtchev *et al.*, 2003]. For this purpose, the project, which is an open-source platform includes a series of existing algorithms and offers the possibility to integrate other algorithms as well. Hence, `Galicia` aims to be a complete tool supporting the theoretical, as well as practical aspects of FCA and offering the possibility to integrate all proposed algorithms in one tool.

2.1.3 Triadic Formal Concept Analysis

2.1.3.1 Theoretical Foundations

Formal concept analysis has constantly developed in the last 30 years. One important direction of evolution is the extension to triadic formal concept analysis (3FCA) proposed by Lehmann and Wille [Lehmann and Wille, 1995; Wille, 1995].

Similar to the dyadic case, the fundamental structures used by 3FCA are those of a triadic formal context and a triadic formal concept, also referred to as triadic concept.

Definition 2.1.16 *A* **triadic formal context** $\mathbb{K} = (K_1, K_2, K_3, Y)$ *is defined as a quadruple consisting of three sets and a ternary relation* $Y \subseteq K_1 \times K_2 \times K_3$. K_1 *represents the set of* **objects**, K_2 *the set of* **attributes** *and* K_3 *the set of* **conditions**. *The notation for an element of the incidence relation is* $(g, m, b) \in Y$ *or* $b(g, m)$ *and it is read* **object g has attribute m under condition b**.

Triadic formal contexts can also be represented as cross tables on layers. Each layer corresponds to one condition and is represented as a dyadic context, the rows of which

are objects and the the columns attributes. However, the cross-table representation is sometimes difficult to visualize, already for contexts of medium size. The following is an example of a triadic context represented as cross tables [Glodeanu, 2013]. The objects of the triadic dataset are hostels, the attributes services provided by the hostels, while the conditions are Web portals where the hostels can be rated.

Example 2.1.4 *Hostels context*

b_0	m_0: *character*	m_1: *safety*	m_2: *location*	m_3: *staff*	m_4: *fun*	m_5: *cleanliness*
$g_0 : NuevoS.$			×			
$g_1 : Samay$		×	×	×		×
$g_2 : OasisB.$	×	×	×	×		×
$g_3 : One$	×	×		×		×
$g_4 : OleB.$	×	×		×		×
$g_5 : GardenB.$			×	×		×

b_1	m_0: *character*	m_1: *safety*	m_2: *location*	m_3: *staff*	m_4: *fun*	m_5: *cleanliness*
$g_0 : NuevoS.$			×	×		
$g_1 : Samay$		×	×	×		×
$g_2 : OasisB.$	×	×	×	×	×	×
$g_3 : One$	×	×	×	×	×	×
$g_4 : OleB.$	×	×	×	×	×	×
$g_5 : GardenB.$	×	×	×	×	×	×

b_2	m_0: *character*	m_1: *safety*	m_2: *location*	m_3: *staff*	m_4: *fun*	m_5: *cleanliness*
$g_0 : NuevoS.$			×	×		
$g_1 : Samay$	×	×	×	×		×
$g_2 : OasisB.$		×	×	×	×	×
$g_3 : One$	×	×	×	×	×	×
$g_4 : OleB.$	×	×	×	×	×	×
$g_5 : GardenB.$	×	×	×	×		×

Definition 2.1.17 (Derived contexts) *Every triadic context* (K_1, K_2, K_3, Y) *gives rise to the following dyadic contexts:*

$\mathbb{K}^{(1)} = (K_1, K_2 \times K_3, Y^{(1)})$ *with* $gY^{(1)}(m, b) :\Leftrightarrow (g, m, b) \in Y$,

$\mathbb{K}^{(2)} = (K_2, K_1 \times K_3, Y^{(2)})$ *with* $mY^{(2)}(g, b) :\Leftrightarrow (g, m, b) \in Y$, *and*

$\mathbb{K}^{(3)} = (K_3, K_1 \times K_2, Y^{(3)})$ *with* $bY^{(3)}(g, m) :\Leftrightarrow (g, m, b) \in Y$.

$\mathbb{K}^{(ij)}_{A_k} = (K_i, K_j, Y^{(ij)}_{A_k})$, *with* $\{i, j, k\} = \{1, 2, 3\}$ *and* $A_k \subseteq K_k$, *where* $(a_i, a_j) \in Y^{(ij)}_{A_k}$ *if and only if* $(a_i, a_j, a_k) \in Y$ *for all* $a_k \in A_k$.

Intuitively, the contexts $\mathbb{K}^{(i)}$ represent "flattened" versions of the triadic context, obtained by putting the "slices" of (K_1, K_2, K_3, Y) side by side. Moreover, $\mathbb{K}^{(ij)}_{A_k}$ corresponds to the intersection of all those slices that correspond to elements of A_k.

In triadic FCA, there are two extensions for the dyadic derivation operators.

Definition 2.1.18 ((i)-derivation operators) *For $\{i, j, k\} = \{1, 2, 3\}$ with $j < k$ and for $X \subseteq K_i$ and $Z \subseteq K_j \times K_k$ the (i)-derivation operators are defined by:*

$X \mapsto X^{(i)} = \{(a_j, a_k) \in K_j \times K_k \mid (a_i, a_j, a_k) \in Y \text{ for all } a_i \in X\}$.

$Z \mapsto Z^{(i)} = \{a_i \in K_i \mid (a_i, a_j, a_k) \in Y \text{ for all } (a_j, a_k) \in Z\}$.

Obviously, these derivation operators correspond to the derivation operators of the dyadic contexts $\mathbb{K}^{(i)}, i \in \{1, 2, 3\}$.

Definition 2.1.19 ((i, j, X_k)-derivation operators) *For $\{i, j, k\} = \{1, 2, 3\}$ and $X_i \subseteq K_i, X_j \subseteq K_j, X_k \subseteq K_k$, the (i, j, X_k)-derivation operators are defined by*

$X_i \mapsto X_i^{(i,j,X_k)} = \{a_j \in K_j \mid (a_i, a_j, a_k) \in Y \text{ for all } (a_i, a_k) \in X_i \times X_k\}$

$X_j \mapsto X_j^{(i,j,X_k)} = \{a_i \in K_i \mid (a_i, a_j, a_k) \in Y \text{ for all } (a_j, a_k) \in X_j \times X_k\}$.

The (i, j, X_k)-derivation operators correspond to those of the dyadic contexts $(K_i, K_j, Y^{(ij)}_{X_k})$.

Similar to the notion of formal concepts in dyadic FCA, triadic concepts can be defined [Wille, 1995]. A triadic concept is a maximal box of incidences and can be generated using derivation operators. This is formally described in the following definition and propositions [Wille, 1995].

Definition 2.1.20 *A* triadic concept *(short:* triconcept*) of $\mathbb{K} = (K_1, K_2, K_3, Y)$ is a triple (A_1, A_2, A_3) with $A_i \subseteq K_i$ for $i \in \{1, 2, 3\}$ and $A_i = (A_j \times A_k)^{(i)}$ for every $\{i, j, k\} = \{1, 2, 3\}$ with $j < k$. The sets A_1, A_2, and A_3 are called* **extent, intent** *and* **modus** *of the triadic concept, respectively. We let $\mathfrak{T}(\mathbb{K})$ denote the set of all triadic concepts of $\mathbb{K}$.*

Proposition 2.1.4 *The triconcepts of a triadic context (K_1, K_2, K_3, Y) are exactly the maximal triples $(A_1, A_2, A_3) \in \mathfrak{P}(K_1) \times \mathfrak{P}(K_2) \times \mathfrak{P}(K_3)$ with $A_1 \times A_2 \times A_3 \subseteq Y$, with respect to the component-wise set inclusion.*

Proposition 2.1.5 *For $X_i \subseteq K_i$ and $X_k \subseteq K_k$ with $\{i, j, k\} = \{1, 2, 3\}$, let $A_j = X_i^{(i,j,X_k)}, A_i = A_j^{(i,j,X_k)}$ and $A_k = (A_i \times A_j)^{(k)}$ (if $i < j$) or $A_k = (A_j \times A_i)^{(k)}$ (if $j < i$). Then (A_1, A_2, A_3) is the triadic concept $\mathfrak{b}_{ik}(X_i, X_k)$ with the property that it has the smallest k-th component among all triadic concepts (B_1, B_2, B_3) with the largest j-th component satisfying $X_i \subseteq B_i$ and $X_k \subseteq B_k$. In particular, $\mathfrak{b}_{ik}(A_i, A_k) = (A_1, A_2, A_3)$ for each triadic concept (A_1, A_2, A_3) of $\mathbb{K}$.*

In analogy to the dyadic case, triadic concepts have an ordinal structured that can be graphically represented. Particularly, the set $\mathfrak{T}(\mathbb{K})$ of all triadic concepts is structured by three quasiorders given by the inclusion order within each of the three components [Lehmann and Wille, 1995].

Definition 2.1.21 (Quasiorder) *A quasiorder or preorder is a binary relation that is reflexive and transitive.*

Definition 2.1.22 (Partial order) *A partial order is a binary relation that is reflexive, transitive and antisymmetric.*

Observation 2.1.5 *The quasiorder is neither necessarily antisymmetric, nor symmetric, therefore it is a weaker version of the partial order. A quasiorder that is antisymmetric becomes a partial order. A quasiorder that is symmetric is an equivalence relation.*

Proposition 2.1.6 *For each of the three dimensions of the triadic context $i \in \{1, 2, 3\}$ there is a corresponding quasiorder $\lesssim_i$ and its corresponding equivalence relation $\sim_i$:*

$(A_1, A_2, A_3) \lesssim_i (B_1, B_2, B_3) \Leftrightarrow A_i \subseteq B_i$

$(A_1, A_2, A_3) \sim_i (B_1, B_2, B_3) \Leftrightarrow A_i = B_i$

Furthermore, each quasiorder has corresponding equivalence classes. Let $[(A_1, A_2, A_3)]_i$ denote the equivalence class of $\lesssim_i$ represented by the triadic concept (A_1, A_2, A_3) .

Proposition 2.1.7 *The quasiorder induces an order relation $\leq_i$ on the factor set of all equivalence classes $\mathfrak{T}(\mathbb{K})/\sim_i$: $[(A_1, A_2, A_3)]_i \leq_i [(B_1, B_2, B_3)]_i \Leftrightarrow A_i \subseteq B_i$*

Unfortunately, in the triadic case, the set of extents, intents and modi do not form a closure system as in the dyadic case [Lehmann and Wille, 1995]. However, the triadic structure can still be represented graphically. According to Proposition 2.1.7, the extent, intent and modi sets are ordered sets and can be represented as Hasse diagrams (line diagrams). The three equivalence classes will be represented as parallel lines and together with the three Hasse diagrams on the side a triadic diagram is obtained. In Figure 2.10 we can see the tradic diagram of the tricontext in Example 2.1.4 as represented by C.V. Glodeanu [2013].

As mentioned before, the parallel lines represent the equivalence classes and the circles at the intersection of the lines represent the triconcepts. On the right side of the triadic diagram one can read the extents set, on the left side the intents set and on the upper side the modi set. In order to deduce the extent of a triconcept one has to follow the interrupted line and read all the objects attached to that level or to a level that is reachable going downwards, similar to reading the components in the dyadic concept lattice. Intuitively,

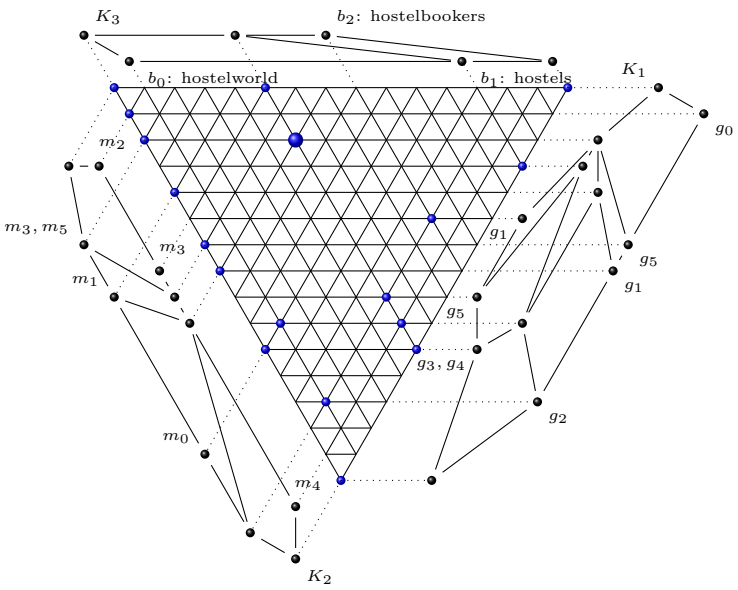


Figure 2.10: Trilattice of the tricontext "Hostels"

extents get larger when reading the Hasse diagram from the lower to the upper part. Similarly, the intents get larger from the upper to the lower part and modi get larger from right to left.

For example, let's consider the highlighted triconcept on the third horizontal line. This triconcept has the following components:

$$(\{g_1, g_2, g_3, g_4, g_5\}, \{m_1, m_2, m_3, m_5\}, \{b_1, b_2\})$$

Observation 2.1.6 *In the Hasse diagram of the extents, object g_1 occurs twice. The reason is that there is no smallest extent that contains the object g_1. This might be the case, because, as mentioned before, extents, respectively intents and modi, do not necessarily form a closure system in the triadic case.*

One can easily deduce that for medium sized tricontexts, the trilattice becomes very difficult to draw. Furthermore, it is not trivial to navigate in a trilattice of considerable sizes. For that reason one of the main directions of our research is to find new methods for navigation in triadic as well as higher-dimensional datasets.

The theoretical aspects of clarification and reduction for tricontexts have not been addressed yet. Therefore, we propose later on a method to extend clarification and reduction of dyadic contexts to triadic ones.

2.1.3.2 Existing Tools and Algorithms

Most of the dyadic algorithms for computing dyadic formal concepts, presented in section 2.1.2.2, cannot be extended efficiently for the triadic case. Hence, new algorithms have to be proposed for mining closed cubes in triadic contexts.

One of the most popular algorithms is `Trias` implemented by Robert Jäschke[1]. Jäschke et al. proposed `Trias` as a solution to the problem of frequent closed itemset mining for folksonomies [Jäschke *et al.*, 2006]. Folksonomies are the core data structure of social resource sharing systems and the authors hoped for an increase of interest in triadic formal concept analysis together with the increase of social network usage. `Trias` projects the ternary relation and applies dyadic formal concept mining on the two obtained dyadic contexts. In the dyadic contexts it uses the `NextClosure` algorithm (mentioned in section 2.1.2.2) enhanced with frequency pruning and computes three-dimensional closed itemsets, hence triconcepts. `Trias` offers the possibility to set the minimum support of the components in the input configuration file. Intuitively, the minimum support relates to the number of elements in the extent, intent and modus. Although any value can be chosen for minimum support, multiple experiments that we ran on triadic data with `Trias` proved that, when trying to include trivial triconcepts in the computation, i.e. choosing the minimum support equal to zero, the results from the `Trias` output are not correct. Despite this drawback, the most common scenario is to choose a higher minimum support in order to compute larger conceptual components and ignore the marginal cases which might just be exceptions. The strategy used for concept mining implies that `Trias` is efficient when one of the dimensions of the context is small.

Two other algorithms proposed by Ji, Tan and Tung are `Representative Slice Mining (RSM)` and `CubeMiner` [Ji *et al.*, 2006]. `RSM` is one of the algorithms that tries to build upon the dyadic case. In order to do that, the `RSM` algorithm first transforms a triadic context into a set of dyadic datasets. Than it uses existing algorithms to mine closed patterns in the dyadic datasets and prunes away the ones that are not closed in the triadic context. Because of the strategy to split the triadic context in dyadic datasets, `RSM` is efficient when one dimension of the triadic context is small. `CubeMiner`, on the other hand, has a completely different approach. It computes the closed patterns directly in the triadic datasets. The algorithm uses a recursive procedure based on so-called "cutters", which are partitions of the dataset containing triples that do not belong to the triadic relation. However, the results obtained in the first phase of the algorithm contain also unclosed sets. In order to obtain the triconcepts, namely the closed patterns, pruning strategies have to be used. In addition to the two algorithms, the authors proposed

[1] `https://github.com/rjoberon/trias-algorithm`

parallel versions of both algorithms, which could reduce the time complexity.

Some algorithms try to address the generalized problem of closed pattern in n-ary relations. One of these algorithms is `Data-Peeler` [Cerf *et al.*, 2008; Cerf *et al.*, 2009] that proposes a method of computing closed patterns in sets of dimensions two or higher. The enumeration strategy of possible patterns has a major impact on the efficiency of the algorithm. The strategy used by `Data-Peeler` is to split the dataset into smaller parts that can be studied independently, meaning the set of closed patterns will be composed from the union of closed patterns of the different parts. Furthermore, in order to be more efficient, the pruning of unclosed patterns is not performed at the end of the enumeration phase, but as soon as possible. One disadvantage of `Data-Peeler` is the space complexity, since the whole dataset must be stored in the main memory. The authors of `Data-Peeler` compare the performances of `Data-Peeler`, `RSM`, `CubeMiner` and `Trias` and conclude that `Data-Peeler` outperforms the other three algorithms.

Another algorithm that was recently proposed by Trabelsi, Jelassi and Yahia claims to outperform all the previously mentioned algorithms for triadic concept computation [Trabelsi *et al.*, 2012]. The authors start by discussing the disadvantages of the other algorithms. Obviously, `Trias`' disadvantage is the use of projections into dyadic contexts, which evolves in a computationally expensive algorithm. The disadvantage of `CubeMiner` is the use of the so-called cutters, since the number of cutters can get very high and therefore lower the efficiency of the algorithm. Another disadvantage of both `CubeMiner` and `Data-Peeler` is the depth-first strategy, which becomes ineffective for a large number of elements. The authors propose a new scalable algorithm, `Tricons`, and for that purpose they define a new closure operator. The closure operator splits the dataset into equivalence classes and hence enables the computation of tri-generators, defined by the authors to enable computing the triconcepts. After extracting the tri-generators, the modus and intent of the triconcepts can be computed. The experiments run showed an efficiency increase of 33.57% compared to `Trias` and it followed that `Tricons` outperforms `CubeMiner` and `Data-Peeler` as well. Furthermore, they contradict the result of Cerf et al. [Cerf *et al.*, 2009] and state that `Trias` outperforms `Data-Peeler`. From the space complexity point of view, `Tricons` has an advantage, since it is the only algorithm that does not store the dataset in memory before computing the triconcepts.

Besides the algorithms for computing triadic formal concepts, there are other approaches that try to relax the condition of formal contexts, by admitting some "zeros" in the clusters, i.e. some elements that are not in relation with all the pairs from the other two dimensions of the cluster. Such algorithms [Gnatyshak *et al.*, 2013] might be more efficient than the one above, but usually they cannot be used for computing formal

concepts. Moreover, Cerf et al. extend `Data-Peeler` to a new algorithm called `Fenster`, which deals with noisy datasets and at the same time add constraints such as closedness and completeness to the outputted patterns [Cerf *et al.*, 2013]. `Fenster` uses the enumeration strategy of `Data-Peeler`, which is the main efficiency strategy used by the initial algorithm. However, in order to remain scalable and efficient for noisy data, it uses other strategies as well. Moreover, the authors state that one can add a large variaty of constraints when running the algorithm in order to search for patterns satisfying certain properties. Cerf et al test `Fenster` on synthetic as well as real datasets and compare the results to similar algorithms that compute noise tolerant patterns. The main observed advantage of `Fenster` is the fact that it can enforce the closedness constraint, i.e. it can compute formal concepts, and hence show fewer and more relevant patterns.

When using triadic formal concept analysis on certain datasets which do not have a triadic structure initially, a preprocessing phase is necessary in order to obtain the triadic subset to be analyzed. With this purpose we used a preprocessing tool called `Toscana2Trias`, which was developed at Babeş-Bolyai university as a previous student project and represents an extension of the dyadic tool `ToscanaJ` [Becker and Correia, 2005]. `Toscana2Trias` allows the selection of triadic data starting from a given set of scales, if the data has been preprocessed with `ToscanaJ`. As the name of the tool already suggests, the output can be further processed with `Trias` [Jäschke *et al.*, 2006] in order to compute the triconcepts.

2.1.4 Polyadic Formal Concept Analysis

Polyadic formal concept analysis introduced in the general form by Voutsadakis [2002], describes conceptual hierarchies arising from n-ary relations. An n-adic formal context can be defined as follows:

Definition 2.1.23 *An* n-context *is an* $(n+1)$*-tuple* $\mathbb{K} = (K_1, \ldots, K_n, R)$ *with* $K_1, \ldots, K_n$ *being sets, and* $R \subseteq K_1 \times \ldots \times K_n$ *the* n*-ary* incidence relation*.*

In the n-adic case, the derivation operators are quite complex and more difficult to understand. For that reason and since we don't use these operators in our future work, we give an alternative definition to the n-adic formal concepts.

Definition 2.1.24 *An* n-concept *of an* n*-context* $\mathbb{K}$ *is an* n*-tuple* $(A_1, \ldots, A_n)$ *satisfying* $A_1 \times \ldots \times A_n \subseteq R$ *and for every* n*-tuple* $(C_1, \ldots, C_n)$ *with* $A_i \supseteq C_i$ *for all* $i \in \{1, \ldots, n\}$*, satisfying* $C_1 \times \ldots \times C_n \subseteq R$ *holds* $C_i = A_i$ *for all* $i \in \{1, \ldots, n\}$*.*

As mentioned previously, the n-adic case has not been studied intensively and there are few algorithms for computing formal concepts in n-ary datasets, such as `Data-Peeler` [Cerf *et al.*, 2008; Cerf *et al.*, 2009] and `Fenster` [Cerf *et al.*, 2013]. Moreover, to the best of our knowledge, there are no navigation or visualization methods for the general case.

2.2 Complexity Theory

In this chapter, we introduce some notions of Boolean logic, first-order logic and complexity classes [Papadimitriou, 1994; Arora and Barak, 2009]. For modeling complexity and efficiency in complexity theory, we use Turing machines. In what follows we assume the reader to be familiar with Turing machines and for a deeper understanding we refer to the references [Papadimitriou, 1994; Arora and Barak, 2009].

First, we introduce a general definition of complexity classes, followed, in the next paragraphs, by the definitions of several complexity classes such as P, NP and AC^0. Other related complexity classes, such as coNP, EXP or NEXP, will not be introduced here, since they are not relevant for understanding the next chapters.

Definition 2.2.1 *A* **complexity class** *is a set of functions that can be computed within a given resource.*

One important complexity class is the class P of decision problems that are solvable by Turing machines in polynomial time.

Definition 2.2.2 (Boolean function) *An n-ary Boolean function is a function $f\{true, false\}^n \mapsto \{true, false\}$. Often the numerical values 1 and 0 are used instead of the Boolean values true, respectively false.*

Definition 2.2.3 (Class P)

$$\mathrm{P} = \bigcup_{p \geq 1} \mathrm{DTIME}(n^p)$$

where $\mathrm{DTIME}(T(n))$ *is the set of Boolean functions computable in $c \cdot T(n)$-time with $c > 0$ constant.*

The class NP can be similarly defined using non-deterministic Turing machines, i.e. machines that have two different transition functions. Hence, NP is the class of problems solvable by a non-deterministic Turing machine in polynomial time. Proving that P≠NP is still an open problem in complexity theory, however, until proven otherwise, we consider the conjecture P≠NP. Formally, NP-hardness and NP-completeness are defined as follows:

Definition 2.2.4 *A decision problem A is* **NP-hard** *if for every problem $B \in$NP, there is a polynomial-time reduction from B to A.*
A problem A is **NP-complete** *if $A \in$NP and A is* NP*-hard.*

Intuitively, NP-hard problems are at least as hard as the other NP problems. The method usually used for proving the complexity of a problem is reducing it to another problem that has a known complexity. Since our research was based on the complexity of some problems from Boolean logic, the following represents a brief introduction into the topic.

Boolean logic is based on Boolean expressions that contain variables combined using Boolean connectives such as logical or $\vee$, logical and $\wedge$ and negation $\neg$. A Boolean variable x_1 or its negation $\neg x_1$ are called literals. Furthermore, literals combined only with the $\vee$ operator, respectively the $\wedge$ operator, are called *disjunctions*, respectively *conjunctions*. Each variable can take one of the truth values *true* or *false* and, depending on the values of the variables, the whole expression is evaluated to *true* or *false*.

It is a known fact that expressions containing other Boolean connectives such as $\Rightarrow$ or $\Leftrightarrow$ can be equivalently converted to expressions that contain only the previously mentioned Boolean operators $\vee$, $\wedge$ and $\neg$ using the De Morgan's laws and other logical equivalences. Formulas of propositional logic are built up using variables and the basic Boolean operators $\vee$, $\wedge$ and $\neg$. Furthermore, we can define standardized forms for formulas as follows.

Definition 2.2.5 *A Boolean expression ϕ is in* **conjunctive normal form (CNF)** *if $\phi = \wedge_{i=1}^{n} C_i$, where each C_i is the disjunction of one or more literals and $n \geq 1$.*
Furthermore, ϕ is in **k-CNF** *if it has at most k literals per clause.*

Definition 2.2.6 *A Boolean expression ϕ is in* **disjunctive normal form (DNF)** *if $\phi = \vee_{i=1}^{n} D_i$, where each D_i is the conjunction of one or more literals and $n \geq 1$.*

It was proven that every Boolean expression has an equivalent expression in conjunctive normal form, as well as one in disjunctive normal form.

An essential notion in complexity theory is the property of satisfiability for a Boolean expression.

Definition 2.2.7 *A Boolean expression is called* **satisfiable** *if there is a truth assignment for its variables such that the whole expression is evaluated to true. Furthermore, if the Boolean expression is true for all the possible truth assignments, then it is called* **valid** *or a* **tautology**.

The fact that a Boolean expression is *unsatisfiable*, i.e. there is no truth assignment for which it is *true*, is equivalent to the fact that its negation is valid. As mentioned before,

when studying certain properties of an expression, that expression is usually rewritten in one of the canonical forms. Hence, the satisfiability problem SAT is defined as deciding whether a Boolean expression in conjunctive normal form is satisfiable or not.

problem: SAT
input: family $\mathcal{L} = \{L_1, \ldots, L_n\}$ of sets L_i of literals of the form p or $\neg p$.
output: YES in case the Boolean formula $\varphi_{\mathcal{L}} = \bigwedge_{L \in \mathcal{L}} (\bigvee_{\ell \in L} \ell)$ is satisfiable, NO otherwise.

In order to be able to study particular cases of the satisfiability problem, the notation kSAT with $k \geq 1$ denotes a SAT problem for a formula in k-CNF. Obviously, different particular cases of SAT problems have been studied, however the relevant one for our research is the case $n = 3$.

problem: 3SAT
input: family $\mathcal{L} = \{L_1, \ldots, L_n\}$ of 3-element sets L_i of literals of the form p or $\neg p$.
output: YES in case the Boolean formula $\varphi_{\mathcal{L}} = \bigwedge_{\{\ell_1, \ell_2, \ell_3\} \in \mathcal{L}} (\ell_1 \vee \ell_2 \vee \ell_3)$ is satisfiable, NO otherwise.

The following represents an example of a satisfiable 3SAT problem.

Example 2.2.1 *Consider $\mathcal{L} = \{L_1, L_2, L_3\}$ with $L_1 = \{r, s, \neg q\}$, $L_2 = \{s, \neg q, \neg r\}$, and $L_3 = \{\neg q, \neg r, \neg s\}$. The corresponding 3SAT problem amounts to checking if $\varphi_{\mathcal{L}} = (\neg q \vee r \vee s) \wedge (\neg q \vee \neg r \vee s) \wedge (\neg q \vee \neg r \vee \neg s)$ is satisfiable. We deduce that the 3SAT problem is satisfiable (the output is* YES*), since, for the valuation $v = \{q \mapsto true, r \mapsto false, s \mapsto true\}$, the formula $\varphi_{\mathcal{L}}$ evaluates to true.*

The following theorem has been proven for SAT [Papadimitriou, 1994; Arora and Barak, 2009].

Theorem 2.2.1 (Cook-Levin Theorem) *SAT is* NP*-complete.*

Using polynomial reduction from SAT to 3SAT it can be shown that 3SAT is NP-hard and furthermore, 3SAT proves to be also NP-complete [Papadimitriou, 1994; Arora and Barak, 2009].

Theorem 2.2.2 *3SAT is* NP*-complete.*

It can be shown that every NP problem can be reduced to a 3SAT problem. We will use 3SAT later to show NP-hardness of certain problems.

In what follows, we will introduce Boolean circuits, which is a generalization of Boolean formulae, as an alternative model of computation. Boolean circuits represent a model

for nonuniform computation, i.e. a different algorithm is used for each input size, in constrast to the uniform computation of Turing machines where the same algorithm is used regardless of the input size. Intuitively, a Boolean circuit is just a graph showing the different combinations that can be used to obtain the output using the basic Boolean operations $\vee$, $\wedge$ and $\neg$. Formally, a Boolean circuit is defined as follows:

Definition 2.2.8 *A* **Boolean circuit** *is a directed acyclic graph that contains input nodes, i.e. with no incoming edges, labeled with the input variables, output nodes, i.e. nodes with no outgoing edges, and several other nodes called* **gates***, which are labeled with one of the logical operations $\vee$, $\wedge$ and $\neg$. The length of the longest directed path from an input node to an output node is called* **depth** *of the circuit. The special case of Boolean circuits, where each node has at most one outgoing edge is called* **Boolean formula** *or* **propositional formula**.

Intuitively, a Boolean circuit with n input nodes and m output nodes represents a function $\{0,1\}^n \to \{0,1\}^m$. The next complexity class we introduce is AC^0, the class of problems solvable by Boolean circuits of polynomial size and constant depth.

Definition 2.2.9 (class $\mathbf{AC^0}$)
AC^0 *is the class of problems solved by a family of circuits $\{C_n\}$ where C_n has the depth $O(1)$, the size $O(n^c)$, where c is a constant, and gates $\vee$ and $\wedge$ of unbounded fan-in.*

We assume the reader to be familiar with first order logic.

Remark 2.2.1 *It is known that the complexity class* AC^0 *coincides with expressibility by first-order formulae [Immerman, 1999]. Therefore, we can prove that a problem is in* AC^0*, by finding an equivalent first-order formula.*

2.3 Answer Set Programming

Answer Set Programming (ASP) [Gebser *et al.*, 2012] is a logic programming language and hence uses a declarative approach to solve NP-hard problems. This approach differs from the imperative approach in the sense that the programmer does not tell the computer what steps to follow in order to solve the problem, but rather describes the problem and lets the computer decide how to solve it. Mainly, in ASP one has to express the problem in a logic programming format consisting of facts and rules, so that the solutions of the problem correspond to models of the logic program.

In what follows, we briefly introduce the syntax and semantics of propositional normal logic programs under the stable model semantics [Gebser *et al.*, 2012; Gelfond and Lifschitz, 1988]. We will present directly the syntax used in the source code in order to avoid a translation phase from one syntax to the other.

Let $\mathcal{D}$ denote the *domain*, i.e. a countable set of elements, also called *constants*. Next, we define an *atom* as an expression of the type $p(t_1, \ldots, t_n)$, where p is a *predicate* of arity $n \geq 0$ and every t_i is an element from the domain or a variable, denoted by an upper case letter. An atom is called *ground* if it is variable-free. The set of all ground atoms over $\mathcal{D}$ is denoted by $\mathcal{G}_\mathcal{D}$. A *(normal) rule* ρ is defined as:

$$a_1 :- b_1, \ldots, b_k, not\ b_{k+1}, \ldots, not\ b_m$$

, where $a_1, b_1, \ldots, b_m$ are atoms or a *count* expression, $m \geq k \geq 0$ with the observation that the left or the right part of the rule might be missing, but not both at the same time. Count expressions have the form $\#count\{c : c_1, \ldots, c_i\} \circ u$, where c is an atom, u is a non-negative integer, $c_j = d_j$ or $c_j = not\ d_j$, for all $1 \leq j \leq i$ and for an atom d_j, and $\circ$ denotes one of the operations $\leq, <, =, >, \geq$. The left part of the rule, i.e. the part before ": $-$" is called *head*, denoted $H(\rho) = \{a_1\}$, while the right part is the *body* of the rule, denoted $B(\rho) = \{b_1, \ldots\ b_k, not\ b_{k+1}, \ldots, not\ b_m\}$. As mentioned previously a rule does not necessarily contain a non-empty head and body, namely, when the head of the rule is empty, we call the rule a *constraint* and, when the body of the rule is missing and a_1 is ground, it is called a *fact*. In case the rule is a fact, we usually omit the sign ": $-$" and write just the atom in the head of the rule. In the definition of the rule, *not* denotes default negation, which refers to the absence of information as opposed to classical negation ($\neg a$) which implies that the negated information is present. Intuitively, "*not a*" means that $a \notin I$, while $\neg a$ implies $\neg a \in I$, for an interpretation I, where $I \subseteq \mathcal{G}_\mathcal{D}$ can be understood as the set of ground atoms which are true. Furthermore, we denote $B^+(\rho) = \{b_1, \ldots, b_k\}$ and $B^-(\rho) = \{b_{k+1}, \ldots, b_m\}$. A rule ρ is called *safe* if each variable in ρ occurs in $B^+(\rho)$. Finally, we define a propositional normal logic program as a finite set of normal rules.

In order to define when a program Π is satisfied by an interpretation I, let $\mathcal{U}_\Pi$ denote the subset of constants from the domain that appear in the program Π and $Gr(\Pi)$ the grounded program, i.e. the set of grounded rules obtained by applying all the possible substitutions from the variables to the constants in $\mathcal{U}_\Pi$, for all the rules $\rho \in \Pi$. We say that an interpretation $I \subseteq \mathcal{G}_\mathcal{D}$ satisfies a normal ground rule $\rho \in \Pi$ (that is not a count-expression) if and only if the following implication holds:

$$B^+(\rho) \subseteq I, B^-(\rho) \cap I = \emptyset \Rightarrow H(\rho) \subseteq I.$$

Then the interpretation I satisfies a non-groud rule if it satisfies all the possible groundings of the rule. In case the rule contains a count-expression $\{c|c_1, \ldots, c_n\} \circ u$, we compute the number n of instantiations for c that satisfy $\{c|c_1, \ldots, c_n\}$ and also belong to the interpretation I, and say that I satisfies the count-expression if $n \circ u$, where u is a non-negative integer and $\circ$ denotes one of the operations $\leq, <, =, >, \geq$. Finally, the interpretation I satisfies a program Π if it satisfies all its rules, i.e. it satisfies the grounded program $Gr(\Pi)$.

An interpretation $I \in \mathcal{G}_\mathcal{D}$ is called an *answer set* or a *stable model* [Gelfond and Lifschitz, 1988] of the program Π if and only if it is the $\subseteq$-minimal model satisfying the reduct Π^I defined by $\Pi^I = \{H(\rho) : -B^+(\rho) \mid I \cap B^-(\rho), \rho \in Gr(\Pi)\}$. Furthermore, we define *cautious* entailement as the intersection of all answer sets.

ASP solving is split in two phases. In the first phase, a grounder has to be used in order to process the logic program into a finite variable-free propositional representation of the problem encoding. In the next phase, a solver uses as input the output of the grounder and computes the solutions, i.e. the answer sets of the problem.

Despite the fact that ASP is classified as logic programming, it was especially developed for solving problems related to knowledge representation and reasoning and it differs from other logic programming languages such as Prolog. The main difference is that, in ASP, the user has almost no control over the algorithm used to compute the solution, whereas in Prolog the user can exercise some control over the solving process.

Researchers from the University of Potsdam developed a project called Potassco[2], which is a collection of answer set solving tools. In our research we used the solving tools from the Potassco collection [Gebser *et al.*, 2011], since it is currently the most prominent solver leading the latest competitions [Calimeri *et al.*, 2016]. In what follows we will describe these tools in more details.

The first tool, `gringo` is a grounder that can be used in the first phase in order to transform the initial encoding into an equivalent variable-free, i.e. ground, program. `Gringo` has a simple, but comprehensive syntax that can express different types of rules (normal, choice, cardinality, etc.), constraints (integrity, cardinality, etc.), but also optimization statements. Intuitively, a constraint expresses a "forbidden" behavior of the models, i.e. if the body of the constraint is true then the model is not a stable model. The output of `gringo` is in the *smodels* format, which is an intermediate format used for the ASP solver input.

The second tool, `clasp`, is a solver that uses the smodel format for the input and computes the answer sets of the program. The output of `clasp` can be configured by

[2] `http://potassco.sourceforge.net/`

the user and shows all or some of the following details: the number of solutions and whether the problem is *satisfiable*, i.e. it has at least one stable model, or *unsatisfiable*, the solutions and the detailed time of the execution. The format of the answer sets is also configurable by adding to the encoding which predicates to print in the output.

Both the tools, `gringo` and `clasp`, were combined into one tool, called `clingo`, in order to avoid processing the ASP program with `gringo` and then further process the output with `clasp`. Avoiding the intermediate step is particularly useful if the grounded program is not of interest and the only results needed are the answer sets of the program. Furthermore, in case one needs to compute the execution time of the whole ASP solving process, the integrated tool shows the cumulative duration of the two phases, grounding and solving.

`Clingo` supports numerous options that can configure the final output, including an option regarding cautious entailment, which iteratively computes the intersection over all answer sets, in the order in which they are computed. However, no matter the order in which they are computed, the last outputted solution when using the *cautious* option is always the intersection of all answer sets of the program. In Chapter 3.4 we describe the tool that we built based on ASP as well as the encoding used for modeling the problem. For the implementation of this tool, the cautious option serves the purpose of optimization and it turns out to have a great impact on the execution time of the tool we developed.

3. Visualization, Navigation and Exploration in Polyadic Datasets

3.1 Formal Concept Analysis for Web Usage Mining

3.1.1 Web Usage Mining and Web Analytics Metrics

In this chapter we apply formal concept analysis to Web usage mining of an educational portal. The analysis and experiments described in Section 3.1.4 and Section 3.1.5 were published as a conference and a journal paper in 2014 [**Dragoş *et al.*, 2014a; Dragoş *et al.*, 2014b**].

Web mining is a research field that uses data mining techniques in order to discover and extract knowledge from Web data. It is often used in order to deal with information overload problems as well as to analyze and extract relevant knowledge from the available Web data such as content of Web pages, link structure or logs [Kosala and Blockeel, 2000]. Web mining can be divided in three different categories [Romero *et al.*, 2009]:

- Web content mining, which deals with extracting knowledge from the documents themselves, by analyzing either their content or their description
- Web structure mining, which deals with inferring knowledge from the structure of the Web and the links
- Web usage mining, which deals with extracting knowledge from Web access logs and analyzing potential patterns

In this chapter we will address the third type, namely Web usage mining. A large amount of secondary data about Web usage is stored in databases or Web server logs. Statistics and data mining techniques are used to discover and extract useful information from these logs. When data mining techniques are being used, the whole process is called Web usage mining. Web usage mining has different goals, such as extracting knowledge in

order to identify potential patterns, predicting the behavior of users interacting with the website, optimizing and personalizing the structure of the website. Changing the structure of the website can benefit the user as well as the owner of the website depending on the type of the changes:

- allowing the user to find the information he is looking for easier and faster
- intentionally adding a page the user usually does not visit on a path that he is likely to follow, i.e. changing the structure to satisfy the goals of the website by forcing the user to visit a page he is not necessarily interested in
- making information more accessible and effective by changing the structure to be more intuitive according to the navigational patterns of the users

All the enumerated aspects of the Web usage mining are topic of numerous research papers [Srivastava *et al.*, 2000; Kosala and Blockeel, 2000; Romero *et al.*, 2013; Eirinaki and Vazirgiannis, 2003; Romero *et al.*, 2009]. Considering the fact that Web usage mining focuses on pattern analysis, usage profiles, system improvement and business intelligence, it is not surprising that there are several tools supporting one or more of these aspects. One of the more complex tools is `WUM` which provides also a mining query language MINT that allows the expert to guide the discovery of navigational patterns [Spiliopoulou and Faulstich, 1998]. We will not go into details about all the existing Web usage mining tools, since they are out of focus in this analysis. We do however need to introduce the analytics metrics used by these tools, which are also the metrics used in our work based on formal concept analysis.

The analytics metrics that analytics tools are based on, are usually defined by organizations such as JICWEBS (The Joint Industry Committee for Web Standards in the UK and Ireland), ABC (Audit Bureau of Circulation) or DAA (Digital Analytics Association, former Web Analytics Association - WAA). The main analytics metrics are defined based on the following notions [Dragoş, 2011]:

- *Website* - a set of Web pages interrelated by links
- *Page view/ Page request* - the rendering of one page, including all its elements and resources such as images, javascripts and others
- *Visitor/ User* - a uniquely identified client that requests Web pages; a user is identified most accurately via a registered account, but more often it is only identified (not always accurately) via it's computer using cookies, the IP or the user agent

- *Visit/Session* - a sequence of pages visited by the same user within a website and during a period of time of maximum 30 minutes, which is the usual limit of a session duration
- *Bounce* - a visit that consists of a single page view
- *Visit Duration* - the average time spent by a user on a site during one visit and computed using the timestamps of the first and the last accessed page; bounces as well as the last pages visited in a longer sequence are not taken into consideration since it cannot be determined how much time they spent on these pages
- *Visit Depth* - the average number of pages accessed during one visit
- *Frequency* - the average number of visits per unique users
- *Recency* - the period of time since the last visit

One of the main problems in Web usage mining is identifying unique users and sessions. As mentioned before, the most common methods for user identification are IP-, user-agent- or cookies-based. For the case of IP-based identification there are several problems. First of all, users can access a website from different IPs, by using different devices. Furthermore, there can also be multiple users using the same device (for example more students using the same laboratory's computer), hence having the same IP. The user-agent is also not an accurate method of identification since a user may use different browsers. Cookies on the other hand, are often blocked and hence cannot offer relevant information for user or session identification. In fact, the only accurate solution for user identification is a registered account where users have to log in at the beginning of a session.

When trying to identify the session, there is usually a time limit taken into consideration by restricting a session to 30 minutes. Another approach uses the structure of the visits within a session and considers that the accessed pages within a session all have to be linked directly by the *referer-access file* relation. Depending on the type of the system and the purpose of the analysis, the right identification method has to be chosen.

However, even if the identification methods are adapted to the available data, Web usage mining fails to offer sufficient support in some use cases. For example, Norguet et al. [2007] mention an example from the industry, where the executive level of an organization often requires summarized and conceptual information in order to take important decisions in an informed and effective manner. For this purpose, the results produced by Web analytics fail to provide the necessary support, since the Web analytics tools usually provide information targeted at Web developers and designers of websites. We believe that

another example where available Web analytics tools cannot offer relevant or sufficient information is the case of educational platforms. In order to understand why and to improve these results, first we need to analyze the different phases of the Web usage mining process.

The process of Web usage mining comprises three phases:

- preprocessing
- pattern discovery
- pattern analysis

The preprocessing phase is essential, since, at this stage, the data is cleaned and prepared for pattern discovery. In the data preparation phase, some entries need to be removed and some need to be enhanced. The most common example of information that should be filtered out before pattern analysis is crawler activity. These entries, obviously, do not give any insight in the user's activity on the website and should not be considered for further analysis. On the other hand, there can be missing information due to caching. In order to deal with this problem the logs can be enhanced by adding the missing page references. This process was called by Cooley et al. *path completion* [Cooley *et al.*, 1999]. After cleaning and completing the data, the users, sessions and page views should be identified. Sometimes, for this purpose the website topology has to be known and if available also a page classification which has to be provided by the domain expert. The expert usually has some predictions about how the site will be used and organizes the structure accordingly, hence he can provide an accurate classification of the websites.

In the second phase, different data mining techniques such as clustering, classification and association rules can be applied in order to identify navigational patterns. For each of these techniques numerous methods have been proposed and applied over the years. The most common techniques that have been used for classifying data are the following: neural networks, genetic algorithms, regression techniques, discriminant function analysis. However, among the previously mentioned data mining techniques, clustering seems to be the most straight-forward and also efficient technique, since it can detect items having similar characteristics. Eirinaki and Vazirgiannis highlight that in Web mining there are two cases of clustering: user clustering and page clustering [Eirinaki and Vazirgiannis, 2003]. In addition to detecting navigational patterns, basic statistics, such as the number of requests for each page or the average time spent on a page can be inferred in this phase.

The patterns stored in the previous phase can be further analyzed in the third phase and the results can be applied to user profiling, website personalization and website improvement. While website improvement regards changing the website for all users, site

personalization means taking the user's characterization into account and dynamically changing the website to better suite each user's needs. An important feedback for the domain expert is to see if the website is being used as predicted. In case there is unpredicted behavior it can mean several things: the design and structure of the website is not intuitive or efficient, the website is not of great interest to the users or it does not serve the needs of the users. However, the obtained information can be used to address all these problems and to further analyze and predict the users' behavior.

3.1.2 Web Usage Mining for E-learning Systems

In the last years research focused on applying data mining techniques to help teachers and administrators of e-learning platforms improve the educational system. Cristóbal Romero et al. have been studying knowledge discovery in e-learning systems for a long time and they call it *Educational Data Mining* (EDM) [Romero and Ventura, 2007; Romero *et al.*, 2009; Romero *et al.*, 2013]. A few of the problems in e-learning that researchers try to address are: students' assessment, adapting the course to the user's needs, offering recommendations of learning paths to students, characterizing students' behavior and classifying students, offering feedback to the instructors so they can improve the courses.

Learning management systems facilitate the communication between teachers and students. On the one side, teachers can distribute new content, assignments, quizzes or news to the students, on the other side, students can, besides benefit from the available information, use forums and chats to collaborate amongst each other or a contact form to communicate with the teacher. During the learning process, a lot of additional information can be gathered from the logs containing the activities of the students. Furthermore, for using an e-learning platform, registration is required, so personal information about each student's account is also available. However, for browsing some public parts of the platform, usually containing general information about courses, it is not required to be logged in. This means the user identification problem is not completely solved, but if one wants to analyze the navigational paths containing pages with course content, then information about user identification is usually available.

Nevertheless, when analyzing the data logs of an e-learning platform, it turns out that some of the usual techniques of Web usage mining cannot be successfully applied. The reason behind this is that most of the Web usage mining techniques were developed for e-commerce websites or catalogs which share some basic functionality principles since the users have similar goals. There are several Web analytics tools based on Web analytics metrics, which have proven to give a good insight into analyzed commercial websites. However, in e-learning the same principles do not apply and the same tools prove to be

rather inefficient. The main reason is that the goals of the users are completely different, since the behavior of users in e-learning environments is driven by the desire of acquiring information. This is a subjective goal which is hard to measure. The learning process takes time and therefore the heuristics used by most analytics instruments do not apply to a visit on an educational site [Dragoş, 2011]. For example, on many websites it can be considered that during a limited period of time requests coming from the same host correspond to the same user. This is not the case for e-learning systems, since they are used for teaching and the portal is often accessed from the university laboratories, meaning that for a fixed period of time several students can access the portal. Hence, new Web usage mining approaches and techniques have to be researched for e-learning systems.

The preprocessing of the data of an e-learning system usually takes place at the end of a course or semester and has a few aspects that differ from other systems. As mentioned previously most of these systems are used by registered users. This does not only simplify the user identification, but also the session identification especially in the cases when the user logs out, although there are users accessing some pages of the system without logging in or users that log in but end their visit by closing the browser instead of logging out. Moreover, many e-learning systems save the usage data not only in log files, but also in databases. This gives the advantage of a better and more reliable analysis of the usage behavior.

In our research ([**Dragoş *et al.*, 2014a; Dragoş *et al.*, 2014b**]), we apply Web usage mining to an e-learning system called PULSE [Dragoş, 2007; Dragoş, 2009; Dragoş, 2010]. PULSE was previously studied by S. Dragoş, R. Dragoş and C. Săcărea using different techniques and tools such as Web analytics [Dragoş and Dragoş, 2009a], Web mining [Dragoş, 2011], visualization using force directed graphs [Dragoş and Beldean, 2013] and formal concept analysis [Dragoş and Săcărea, 2012]. The first tools the authors implemented with the purpose of analyzing the usage data of PULSE were `WATEC` (Web Analytics Tool for Educational Content), a PHP tool that creates a database with the usage data and generates statistics about the logged data, and `GAL` (Google Analytics-Like) [Dragoş and Dragoş, 2009a; Dragoş and Dragoş, 2009b; Dragoş, 2011]. `GAL` is based on Google Analytics, but adapted to handle educational data usage. `WATEC` and `GAL` differ in the Web analytics metrics used, such as identification of unique visitors and visits. When comparing the two tools, `WATEC` proved to be more efficient in analyzing Web usage for educational systems. In 2012, S. Dragoş and C. Săcărea tried to analyze the log data of the e-learning portal by visualizing it with `ToscanaJ` [Becker *et al.*, 2002], a formal concept analysis tool [Dragoş and Săcărea, 2012]. The learning management system PULSE is

described in more details in the following section.

3.1.3 PULSE - a PHP Utility used in Laboratories for Student Evaluation

PULSE is a learning management system developed by S. Dragoş in 2007 and extended in the following years [Dragoş, 2007; Dragoş, 2009; Dragoş, 2011]. It was specifically designed for managing the work of students during laboratory sessions and therefore supports both students and professors in this process with functionalities specific to each role. Hence, a user logged in as a student has a personalized view and can see information strictly linked to his account such as:

- tasks assigned to him
- personal attendance records
- grades for the submitted tasks

The only common information for all student accounts is the general course documentation or news. This solves the problem of confidentiality regarding grading, since it is often required that grades are not made public.

When logged in as a professor, the user can find information regarding each student registered in the courses linked to the professor's account, such as:

- submitted tasks
- attendance
- student's grades

Furthermore, the instructor has the possibility to order the students' list alphabetically or by different criteria regarding the grades (final grade, average grade).

Many learning management systems offer methods of automatic evaluation, however this is not possible for the laboratory tasks because not only the solution itself has to be evaluated, but also the knowledge gained by the student while resolving the task. Hence the student has to explain the implemented solution and answer additional question that verify his level of understanding [Dragoş, 2007]. The only process that can be automated for the support of laboratory activities is the assignment of tasks. The professor needs to provide a list of tasks, which are then randomly and automatically assigned to the students.

PULSE logs all the usage data in a MySQL database, which contains the following data fields [Dragoş, 2011]:

- the time stamp of the request
- the IP address of the originating Web page request
- full request-URI, including the domain, the requested URL, and any applicable query parameters
- full unmodified user-agent string
- referrer URL
- login id
- cookie id

3.1.4 Applying Formal Concept Analysis on PULSE Usage Data

3.1.4.1 Data Preprocessing and Pattern Discovery

In the Web usage mining research field, a lot of different methods and techniques were applied. However, it was observed that the most efficient techniques for educational Web usage mining are classification and clustering. Clustering can actually be considered as an unsupervised method of classification. Researchers that applied classification methods to educational Web usage mining proposed different methods and algorithms, such as `PageGather`, an algorithm based on clustering that determines groups of pages visited together [Perkowitz and Etzioni, 1997] and many others. We decided to use formal concept analysis, which is a clustering technique that has, to the best of our knowledge, not been used before in educational Web usage mining [**Dragoş *et al.*, 2014a; Dragoş *et al.*, 2014b**].

The analysis was performed on the data collected from the second semester of the academic year 2012-2013, i.e. from the beginning of February to the end of July 2013. For the analyzed time interval there were 40768 PULSE accesses. The data fields from the collected information used in the current investigation are:

- full request-URI
- referrer URL
- login id

- the time stamp of the request
- cookie id

The data to be analyzed contains 751 distinct request-URIs, i.e. access files, 471 distinct referrers, 130 distinct login IDs, 25798 distinct timestamps and 3472 distinct cookie IDs. In the preprocessing phase, in order to obtain a more coarse granularity, scales had to be defined for some of the data fields. We performed the scaling process with the tool `ToscanaJ`, which uses SQL statements to gather the required information from the databases. The request-URI represents the address of the accessed Web page along with all query information used for that actual request. Although we value the information contained in this field, the granularity of the accessed Web pages is too fine for our intent, since there are 751 distinct access file entries in the database. Therefore, the accessed Web pages have been divided into 9 classes according to the pages' utility. As mentioned previously, the PULSE portal was intended to be used mainly during laboratory sessions for students to consult theoretical support provided by the teacher, to access appointed assignments and check their grades and attendance records. The pages corresponding to these activities belong to the classes **Lecture** and **Lab**. Given the authentication phase to the educational portal, we define two more classes: **Home** corresponding to the login phase and **Logout** corresponding to the logout phase. Moreover, PULSE offers a set of other utilities which are split into classes as follows: the **TeacherAdm** class containing administrative utilities for the teachers, the **FAQ** class which contains a section for frequently asked questions, the **Feedback** class, the **News** class and the **Change** class which contains data about the course from the previous academic years. The access files have been nominally scaled based on these classes and the nominal scale has been visualized in `ToscanaJ` as represented in Figure 3.1.

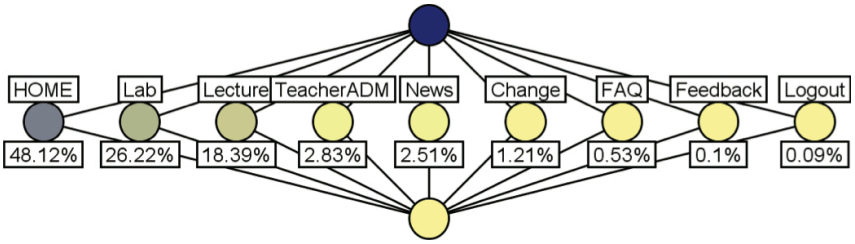
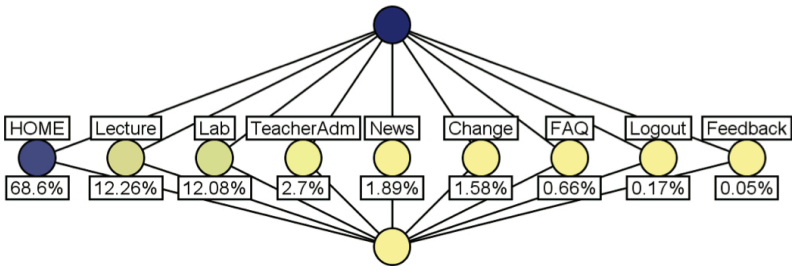


Figure 3.1: Nominal scale for access file

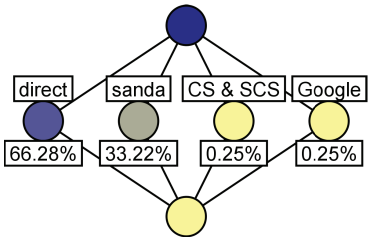
On the other side, the referrer URLs represent the Web page from which the current access file was accessed. These Web pages may be among the access files and they usually are if the user had a continuous navigation path through the websites. The number

of these referrers is also high, 471 distinct referrers, hence classes have been defined in a similar manner. Overall we recognize two different categories of referrers: **inside PULSE** for accesses from within the PULSE portal and **outside PULSE** for accesses that were made from pages not belonging to the educational portal. Each of the two categories of referrers was separately divided into classes. The classes corresponding to the referrers inside of PULSE are the same as the corresponding classes of the access files as depicted in Figure 3.2a. The classes outside of PULSE are described in Table 3.1 and the `ToscanaJ` visualization of their nominal scale is depicted in Figure 3.2b.

Figure 3.2: Nominal scales for the referrers



(a) Nominal scale for the referrers within PULSE



(b) Nominal scale for the referrers outide of PULSE

Table 3.1: Referrer classes outside of PULSE

Class	**Description**
sanda	teacher personal site
cs & scs	faculty site & student site
google	accesses from google search, and/or google mail
direct	direct accesses from bookmarks or by typing the URL of that Web page directly on the browser

The next step in the preprocessing phase, after defining the scales, was to use the `Toscana2Trias` tool. As mentioned in Section 2.1.3.2, after preprocessing the data with `ToscanaJ`, `Toscana2Trias` allows selecting triadic data starting from a given set of scales. In our experiments we extracted and analyzed different triadic structures, which will be presented later in the experiments section.

For the pattern discovery, we used the `Trias` algorithm [Jäschke *et al.*, 2006] to generate all the triconcepts of the previously described triadic context. However, for the pattern analysis phase, we needed a visualization tool that can effectively represent a

relatively high number of triconcepts. Since the problem of visualizing triadic data has not yet been satisfactory solved, we tried to find alternative ways to visualize the results. Trilattices could not be considered in this case, since it would be almost impossible to design a trilattice for a large number of triconcepts. Hence, we considered `Circos`, a visualization tool which is described in more detail in the following section.

3.1.4.2 Pattern Analysis and Visualization using `Circos`

`Circos`[1] is a tool developed by Martin Krzywinski with the purpose of visualizing data in a circular layout. The tool was initially designed to investigate structural patterns arising in bioinformatics, mainly to visualize genomic data. However, it has since been used on data from various fields. As Krzywinski mentions in several of his presentations, graphical displays of the data are very important and while "large data sets hide patterns", a "good visualization reveals patterns". The circular layout of `Circos` emphasizes patterns in the dataset, showing connections between represented data, which makes it suitable also for a formal context.

The input format of `Circos` is a two-dimensional table with numerical data. Therefore, the triadic structure has to undergo a preprocessing phase in order to be transformed in a valid `Circos` input table. We decided to represents the triadic structure in such a way that the two dimensions of the table correspond to two of the three dimensions of the analyzed data, while the third dimension of the data will be involved in computing the numerical values of the table [**Dragoş *et al.*, 2014a; Dragoş *et al.*, 2014b**]. In what follows we will describe this procedure in more detail.

Let the input table for `Circos` be represented by the function $R \times C \mapsto V$, where R is the set of row indicators,C the set of column indicators and V the set of values for the corresponding table cells. The triadic context containing the data to be analyzed is denoted by $\mathbb{K} = (G, M, B, Y)$. The set C of column indicators is represented by the set of attributes described by the following projection $pr_M(Y) = \{m \in M \mid \exists (g, b) \in G \times B, (g, m, b) \in Y\}$, i.e. all attributes that can be found in the incidence relation. Similarly, the set R of row indicators is represented by the set of conditions from the projection $pr_B(Y)$, i.e. all conditions from the ternary relation.

In order to compute the numerical values $v \in V$ of the table, first we consider the dyadic projection of the context $\mathbb{K}^{(1)} = (G, M \times B, Y^{(1)})$ with $gY^{(1)}(m, b) \Leftrightarrow (g, m, b) \in Y$. For each pair $(m, b) \in M \times B$ that can also be found in the table, i.e. $(m, b) \in C \times R$, we compute the corresponding attribute concept $\mu(m, b) = ((m, b)', (m, b)'')$ of the context $\mathbb{K}^{(1)}$. The numerical value of the table cell corresponding to the row m and the column b

[1] `http://circos.ca/`

is the cardinality of the extent $(m, b)'$ of the computed attribute concept $\mu(m, b)$. So we have that $(m, b) \mapsto |(m, b)'|$. Hence, the elements of V can be computed using the table representation of the triadic context, by counting the elements of the object set which are in relation with attribute m and condition b [2].

The final step before using `Circos` to graphically represent the data is to create a configuration file, which controls the creation of the images. The configuration file contains numerous options that can change different aspects of the layout and coloring of the represented data. To conclude the analysis, we visualize our data by running `Circos` and obtaining an output in png or svg format [**Dragoş *et al.*, 2014a; Dragoş *et al.*, 2014b**].

3.1.5 `Circos` Interpretations of Triadic Data

For the first analysis we were interested in investigating temporal patterns of Web usage behavior within PULSE [**Dragoş *et al.*, 2014a; Dragoş *et al.*, 2014b**]. Hence, we restricted our focus to the access files, the referrers and the timestamps of the system. This is a natural triadic structure wherefrom we can extract user dynamics related knowledge structures in form of triadic concepts. As we have seen in the preprocessing phase, access files and referrers were divided into different classes according to the scope of the pages and scales were built on these classes. However, for this analysis we wanted to study the user dynamics within the educational portal, hence the referrer classes outside of PULSE were ignored. We used the scale in Figure 3.1 for the access files and the scale in Figure 3.2a for the referrers and extracted the following triadic structure:

- objects are represented by the students' login id
- attributes are represented by pairs of the form *(referrer class, access file class)*
- conditions are represented by the timestamps

The motivation behind the choice of student login id as object type is the fact that the password-based authentication solves the problem of user identification in the usage data. Moreover, the attributes were chosen to model the navigation from one page to another in order to obtain in the end results about the navigational patterns.

Using the algorithm described in Section 3.1.4.2 the triadic structure was transformed in a data input table for `Circos`. Then we ran `Circos` to obtain a graphical representation

[2] In the articles we published on this topic [**Dragoş *et al.*, 2014a; Dragoş *et al.*, 2014b**] we used a different algorithm for computing $|(m, b)'|$. However, a different, more efficient algorithm is presented here.

of the data. However, for this first attempt one single circular representation did not manage to show the data in a meaningful and readable way or to reveal any patterns. The reason behind is that the volume of the represented data as well as its density were too high. Hence, as a next step we reduced the volume of the analyzed data using different methods:

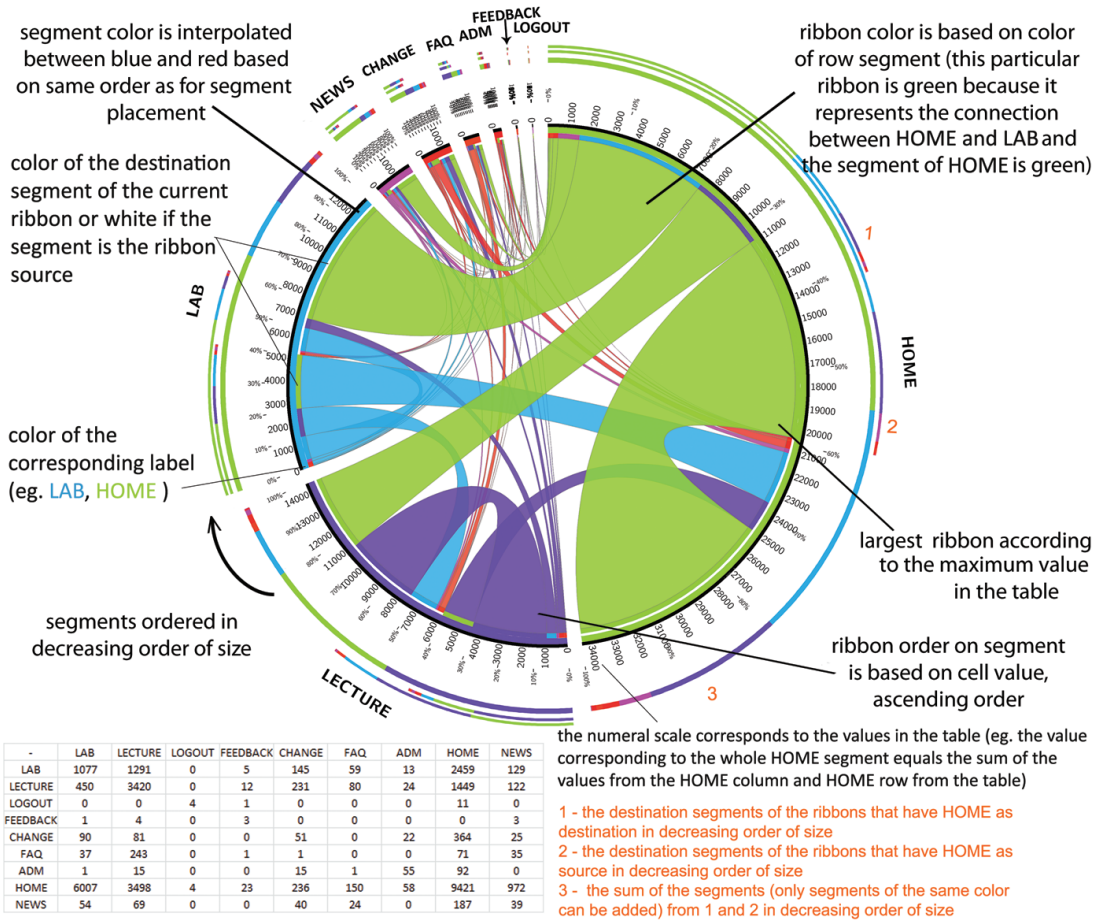
- choosing a different triadic structure with a lower density of the data
- filtering the data using different criterias:
  - the program in which students are enrolled and their year of study
  - different time granulations: one semester, one third of a semester, a week

Next we tried to analyze navigational patterns among the sessions. While the user identification was solved by the authentication method, sessions were identified using cookies. Hence, for this analysis we chose the following triadic structure:

- cookie ids as objects
- referrer classes inside PULSE as attributes
- access file classes as conditions

Using the same algorithm in the preprocessing phase, the data was structured as a table having referrer classes as the column indicators and access file classes as the row indicators. The `Circos` representation of this dataset can be seen in Figure 3.3. We added additional information on the sides of the `Circos` representation in Figure 3.3 in order to explain how the elements of the figure can be interpreted. Elements that are related are joined by links in the form of ribbons. In this particular case, each ribbon corresponds to a pair (referrer class, access file class). However, since the referrer and the access file classes are the same in this case, the classes are only represented once, but can play the role of both referrer and access file class. The quantitative part of the table, i.e. the numerical values corresponding to one of the three dimension of the original data can be deduced from the thickness of the ribbons.

For a better understanding we represented in Figure 3.4 the same relation between referrer and access file classes in the form of a directed graph. The nodes of the graph have the same colors as the corresponding segments in the `Circos` representation. Just as in Figure 3.3, here we can also observe loops, since both sets are divided into the same classes.



-	LAB	LECTURE	LOGOUT	FEEDBACK	CHANGE	FAQ	ADM	HOME	NEWS
LAB	1077	1291	0	5	145	59	13	2459	129
LECTURE	450	3420	0	12	231	80	24	1449	122
LOGOUT	0	0	4	1	0	0	0	11	0
FEEDBACK	1	4	0	3	0	0	0	0	3
CHANGE	90	81	0	0	51	0	22	364	25
FAQ	37	243	0	1	1	0	0	71	35
ADM	1	15	0	0	15	1	55	92	0
HOME	6007	3498	4	23	236	150	58	9421	972
NEWS	54	69	0	0	40	24	0	187	39

the numeral scale corresponds to the values in the table (eg. the value corresponding to the whole HOME segment equals the sum of the values from the HOME column and HOME row from the table)

1 - the destination segments of the ribbons that have HOME as destination in decreasing order of size
2 - the destination segments of the ribbons that have HOME as source in decreasing order of size
3 - the sum of the segments (only segments of the same color can be added) from 1 and 2 in decreasing order of size

Figure 3.3: Navigational patterns for user sessions

Next, we tried to reduce the volume of the data and aggregate the student login ids into student groups according to the program that students were enrolled in and their year of study. We continued our tests treating each group of students separately and analyzing their behavior in time [**Dragoş *et al.*, 2014a; Dragoş *et al.*, 2014b**]. For this purpose we considered the following triadic structure:

- timestamps as objects
- referrer classes as attributes
- access file classes as conditions

A time granularity of one third of a semester did not provide any significant patterns, therefore we fine tuned the time granularity to a week and added an additional filter regarding the activity for specific courses. The analyzed data was gathered for an entire

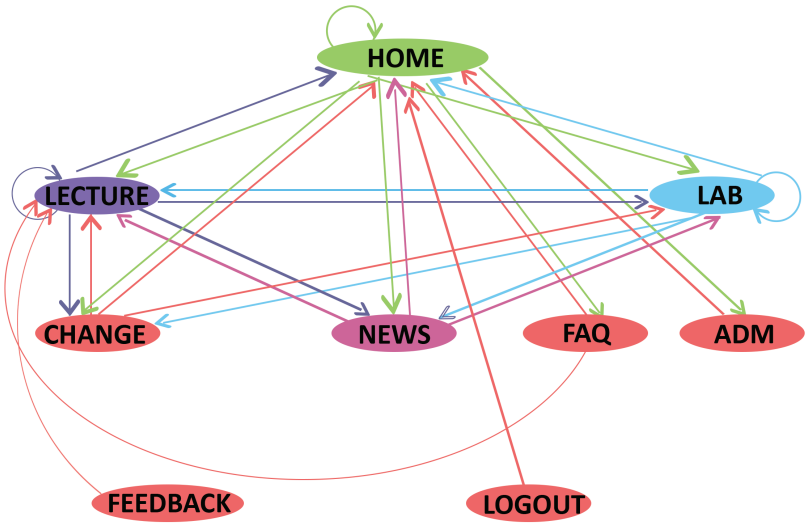


Figure 3.4: Graph visualization of the connections between referrer and access file classes

semester during which students were enrolled in two courses from the PULSE platform: operating systems (SO1), which is a compulsory course, and Web design optimization (WDO), which is an elective course. Two student groups were enrolled in the SO1 course, mathematics-computer science in Romanian language denoted "ar" and software engineering in Romanian language denoted "ri". On the other hand, for WDO there were students from five different groups enrolled, but not all the students from these groups were enrolled since it was an elective course. Some of these groups were poorly represented, so we chose to study the behavior of two student groups which had a higher number of attendees: software engineering in English language denoted "ei" and computer science in English language denoted "ie".

We observed from our analysis and the circular visualizations of data subsets that there are three types of behaviors, which we denote: *relaxed, intense and normal* [**Dragoş *et al.*, 2014a; Dragoş *et al.*, 2014b**]. The *relaxed* behavior occurs mainly during the holiday, for example in the 10th week, but also after final exams or between the final exam and the reexamination: 18th and 20th week for group "ar", 18th and 19th week for group "ri" and after the 14th week for groups "ei" and "ie". The reason why the weeks differ for each student group is that the structure of the courses also differs depending on the group. The pattern for this type of behavior is depicted in Figure 3.6b and Figure 3.7c

and can be distinguished by fewer accesses and a reduced number of access file classes visited (usually only the main classes are visited). For the elective course WDO the results showed a generally more relaxed behavior than for the compulsory course due to the fact that this type of course implies personal research. Therefore, the teaching material provided is less visited that in the case of the compulsory course SO1.

The *intense* behavior occurs during examination periods. The pattern depicted in Figure 3.6c and Figure 3.7b shows an increase in the number of accesses. The pattern can be observed even in the weeks preceding the exam, its peek however occurs during the week of the exam. This behavior could be observed for the compulsory course in the following weeks: 17th week for the "ar" and "ri" groups, 19th week for the "ar" group and 20th week for the "ri" group. The elective course had a different evaluation method, namely students had to develop three projects. This projects were due in the 7th, 9th and 13th week, which were also the weeks with *intense* behavior.

The *normal* behavior occurs during the semester when there is no examination period, but also no holiday. The pattern for this type of behavior, as depicted in Figure 3.6a and Figure 3.7a, shows that almost all access file classes are visited. The three main classes, **HOME**, **LAB** and **LECTURE**, were still the most visited, while the next most visited class was **NEWS**. These results are to be expected as PULSE is mainly intended to provide support for laboratory and lectures.

When comparing the different behavior types, for example for group "ar" as depicted in Figure 3.6, one can notice that during intense behavior the Web pages from the **LECTURE** class are visited the most as the students prepare for the examination, while in the normal period the Web pages from the **LAB** class are visited more as students need to solve their laboratory assignments, that is if we ignore the class **HOME** which is the transition class given the login step.

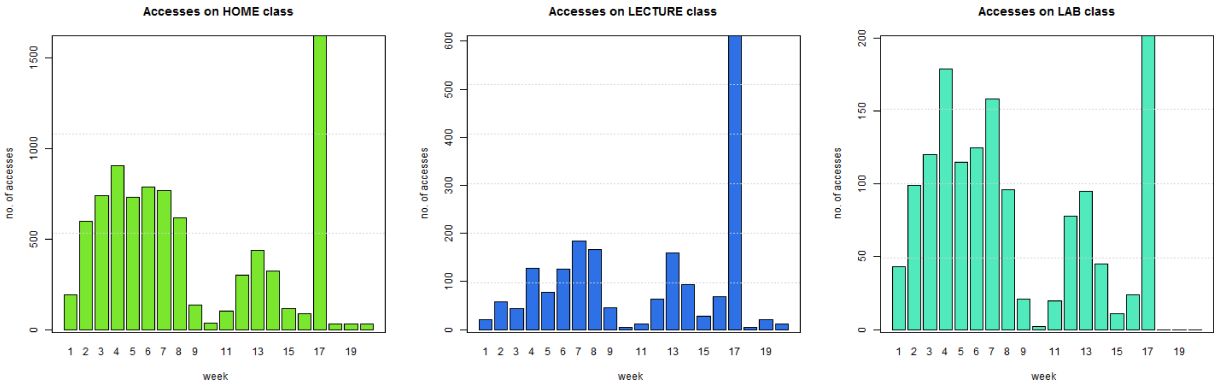


Figure 3.5: Number of accesses for the "ar" student group

The different triadic conceptual landscapes of the analyzed data provide a large amount of information that is suitable for a large variaty of interpretations and visualizations. In order to have a better view of the evolution in terms of number of accesses per week we plotted the results of the "ar" group for different access file classes using histograms. Some of these plots are depicted in Figure 3.5. This representation however, presents only the quantitative aspect of the navigation, namely the number of accesses. The circular visualizations presented so far provide a more qualitative view of the navigational patterns by comprising more details about how the students navigate through the educational platform [**Dragoş *et al.*, 2014a; Dragoş *et al.*, 2014b**].

3.2 Clarification and Reduction of Triadic Contexts

In the next two chapters we will introduce two navigation paradigms for triadic datasets that try to deal with the complexity of the triadic structure of a tricontext. For that purpose, the preprocessing phase of the data plays an important role. While in the dyadic setting there are well-known methods to reduce the size of the data without affecting its underlying structure, these methods are missing in the triadic case. Therefore, driven by practical requirements, we discuss in this chapter triadic extensions of the previously introduced notions from dyadic formal concept analysis: clarification, reduction and object/attribute concepts. Furthermore, we analyze the effects of the clarification and reduction processes on a medical dataset. The results presented in this chapter were published in a workshop paper in 2015 [**Rudolph *et al.*, 2015b**].

As we have seen in Section 2.1.2.1, for dyadic contexts, reducible objects and attributes can be deleted, without affecting the underlying conceptual structure. Clarifying and reducing is thus a preprocessing stage, in order to reduce the dimensions of the context, i.e. the number of elements of each component set of the context, for further analysis. We deduce from Definition 2.1.11 and Observation 2.1.2 that, in the dyadic case, a context is called clarified if there are no identical rows and columns in its cross-table representation. In the triadic case, we can make use of the same idea applied on the "flattened" projection of the tricontext. Since a triconcept (A_1, A_2, A_3) is a maximal triple of triadic incidences, removing identical "slices" in the tricontext does not alter the structure of triconcepts. Clarification in triadic contexts is formally described in the following definition.

Definition 3.2.1 *A triadic context (K_1, K_2, K_3, Y) is* clarified *if for every $i \in \{1, 2, 3\}$ and every $u, v \in K_i$, from $u^{(i)} = v^{(i)}$ follows $u = v$. The derivation $()^{(i)}$ is the corresponding derivation in the triadic context.*

The other important operation performed in the dyadic case is context reduction. This

consists in the removal of reducible objects and attributes, an operation that has no effect on the conceptual structure of the context. Reducible elements in the dyadic case are precisely those elements that can be written as combination of other elements from the corresponding set.

Remark 3.2.1 *As we have seen in Definition 2.1.12, we call a clarified context* (G, M, I) *row-reduced if every object concept is* $\vee$*-irreducible and column reduced if every attribute concept is* $\wedge$*-irreducible. Due to the symmetry of the context, if we switch the role of the objects with that of the attributes and look at the context* (M, G, I^{-1})*, then the new context is row reduced if every object concept (attribute concept in the former context) is* $\vee$*-irreducible. So we can consider only* $\vee$*-irreducible concepts, first in the dyadic context* (G, M, I) *and then, by switching the perspective, in* (M, G, I^{-1})*.*

Similar to the dyadic case, objects, attributes, and conditions which can be written as combinations of others have no influence on the structure of the trilattice of $\mathbb{K}$, so they can be reduced. Considering that for dyadic contexts, reducible elements are defined using object and attribute concepts, we might ask if there are similar notions in the triadic case. Due to the structure of triconcepts, it turns out that a triadic object concept should be defined as a set of triconcepts. Taking this into consideration, we deduce that it would be difficult to find a reduction characterization for tricontexts using triadic object concepts, since we cannot define $\vee$-irreducibility for a triadic object concept. However, using the idea described in Remark 3.2.1, we can define the reduction of a tricontext using object concepts in appropriately defined dyadic contexts [**Rudolph *et al.*, 2015b**].

Definition 3.2.2 *A clarified tricontext* (K_1, K_2, K_3, Y) *is called* object reduced *if every object concept of the dyadic context* $(K_1, K_2 \times K_3, Y^{(1)})$ *is* $\vee$*-irreducible,* attribute reduced *if every object concept of the dyadic context* $(K_2, K_3 \times K_1, Y^{(2)})$ *is* $\vee$*-irreducible, and* condition reduced *if every object concept of the dyadic context* $(K_3, K_1 \times K_2, Y^{(3)})$ *is* $\vee$*-irreducible. An element* $a \in K_i$ *is called irreducible, respectively reducible, if* a *is irreducible, respectively reducible, in the corresponding dyadic projection* $K^{(i)}$*.*

Example 3.2.1 *In the following tricontext we can see an example of an element that can be reduced.*

b_1	m_1	m_2	m_3
g_1	×		
g_2			×
g_3			

b_2	m_1	m_2	m_3
g_1	×		×
g_2	×		
g_3	×		

b_3	m_1	m_2	m_3
g_1	×		
g_2	×	×	
g_3	×		

b_4	m_1	m_2	m_3
g_1	×		
g_2	×	×	
g_3	×		

The non-trivial triconcepts of the described tricontext are:

$(\{g_1\}, \{m_1\}, \{b_1, b_2, b_3, b_4\})$
$(\{g_2\}, \{m_3\}, \{b_1\})$
$(\{g_1, g_2, g_3\}, \{m_1\}, \{b_2, b_3, b_4\})$
$(\{g_1\}, \{m_1, m_3\}, \{b_2\})$
$(\{g_2\}, \{m_1, m_2\}, \{b_3, b_4\})$.

First, we observe that the slices corresponding to the two condition b_3 and b_4 are identical, so in the clarification process we can eliminate condition b_4. Moreover, we can see that by reducing g_3, the number of triconcepts remains unchanged and the trilattice will not change, since the behavior of object g_3 is the same as the common behavior (the "intersection") of the group of objects g_1 and g_2.

Similarly to the dyadic case (see Proposition 2.1.2), we obtain the following characterization for reducible elements.

Proposition 3.2.1 *Let $\mathbb{K} = (K_1, K_2, K_3, Y)$ be a tricontext and $a_i \in K_i$, $i \in \{1, 2, 3\}$. Then the element a_i is reducible if and only if there exists a subset $X \subseteq K_i$ with $a_i \notin K_i$ and $Y_X^{(jk)} = Y_{a_i}^{(jk)}$, where $Y_X^{(jk)} = \{(b_j, b_k) \in K_j \times K_k \mid \forall b_i \in X.\ (b_i, b_j, b_k) \in Y\}$, for $\{i, j, k\} = \{1, 2, 3\}$.*

Proof. The element $a_i \in K_i$ is by definition reducible in $\mathbb{K}$ if it is reducible in $\mathbb{K}^{(i)}$ and from Proposition 2.1.2 we have that a_i is reducible if and only if there exists a subset $X \subseteq K_i$ with $a_i \notin K_i$, s.t. they have the same derivative in $\mathbb{K}^{(i)}$, i.e. $a_i^{(i)} = X^{(i)}$. Now we have the following equivalence chain:
$(a_j, a_k) \in Y_{a_i}^{(jk)} \Leftrightarrow (a_i, a_j, a_k) \in Y \Leftrightarrow (a_j, a_k) \in a_i^{(i)} = X^{(i)} \Leftrightarrow (x, a_j, a_k) \in Y, \forall x \in X \Leftrightarrow$
$(a_j, a_k) \in Y_X^{(jk)}$. □

Remark 3.2.2 *Finite tricontexts can be represented as slices consisting of dyadic contexts. Moreover, this representation has a sixfold symmetry. In order to represent the triadic context in a plane, we just put these slices one next to each other (see previous example). This proposition states that a_i is reducible if and only if the slice of a_i is the intersection of some slices corresponding to the elements of a certain subset $X \subseteq K_i$. This has a striking similarity to the dyadic case, where, for example, an object is reducible, if*

its row is the intersection of the rows from a certain subset X of objects. This also gives us an algorithmic approach to the problem of finding all reducible elements in a tricontext.

Finally, we implement the described clarification and reduction processes for tricontexts and apply them on a cancer registry database comprising information about several thousand patients [**Rudolph *et al.*, 2015b**]. The cancer registry database, in its original form, contains 25 characteristics for each patient, including an *identification number* and other characteristics such as *tumor sequence, topography, morphology, behavior, basis of diagnosis, differentiation degree, surgery, radiotherapy, hormonal therapy, curative surgery, curative chemotherapy*, etc. The described database has been previously analyzed and interpreted by Săcărea using formal concept analysis [Săcărea, 2014]. Therefore, all the characteristics available for a patient are interpreted as conceptual scales and represented as conceptual landscapes for an enhanced knowledge retrieval. This enables us to use the tool `Toscana2Trias`, described in Section 2.1.3.2, in order to obtain triadic structures from the original dataset and analyze it as a tricontext. However, we will not go into details about interpreting the data, since the purpose of this application is solely to analyze the effect of the clarification and reduction processes on the size of the context.

In our tests, we selected two different tricontexts that have patients as objects. For the first example, the selection was restricted to specific types of tumors (as attributes) and their stages (as conditions). After the clarification process, we obtained a small tricontext with 13 objects, 5 attributes and 8 conditions, and 23 triconcepts. In the reduction process, three more objects, one attribute and one condition could be further reduced.

For the next example, we selected a number of 4686 objects, 11 attributes (all 8 degrees of *certainty* in the oncological decision process, *in-situs carcinoma* and *tumor sequence* = 1, i.e. just one tumor) and three conditions (*gender* = *male*, *age* $<$ 59, and *survival* $>$ 30 *months*). This selection generated a relation with 44545 tuples (crosses in the tricontext) and 63 triconcepts, and a clarified tricontext with 61 objects. Herefrom, 38 objects could be reduced as well as 7 attributes (all of them being *certainty*-related, due to the specific selection we have made), resulting in a relation with 77 tuples.

These applications show how the clarification and reduction processes can eliminate redundant information, hence increasing the efficiency in determining the underlying conceptual structure [**Rudolph *et al.*, 2015b**]. We observe that in real datasets, often a large number of elements can be reduced. Clarification and reduction play an important role, not only for optimizing further analysis of the data, but also to have an initial insight into the structure of data, considering that elements that can be eliminated in these processes are grouped into subsets of elements with identical or simi-

lar behavior. In conclusion, clarification and reduction are important processes for the preprocessing phase of a dataset and should be performed, for optimization reasons, before starting to navigate through the conceptual landscape of the context. We will continue, in the next chapters, by introducing two different navigation paradigms that we propose for triadic datasets [**Rudolph *et al.*, 2015c; Rudolph *et al.*, 2015a; Rudolph *et al.*, 2016**]. Moreover, as we will see, one of the proposed navigation paradigms can be easily extended to higher-adic datasets.

3.3 A Triadic Navigation Paradigm based on Reachability Relations

3.3.1 Motivation

Formal concept analysis becomes increasingly popular for its capabilities addressing knowledge processing and knowledge representation as well as offering reasoning support for understanding the structure of large datasets, especially in the dyadic case. As we have shown in Section 2.1.2.1, dyadic datasets can be graphically represented as concept lattices, which offer an intuitive visualization and hence understanding of the dataset's structure. Moreover, in a dyadic lattice, navigation among concepts is straight forward due to the order relations that are defined on the two dimensions of the context.

For cases where the concept lattice gets too big to be represented in a readable way, "local" navigational paradigms have been proposed, where only one concept and its direct neighbor concepts are visualized and the user can explore the concept lattice by successively moving to neighboring concepts [Ferré and Ridoux, 2004; Godin *et al.*, 1993].

In the triadic case, however, trilattices which are the current graphical representation of triadic contexts are hard to obtain, i.e. to graphically represent, even for small datasets and can also be hard to read. Moreover, navigation in a trilattice is not trivial, considering that on each of the three dimensions of the context only quasiorders can be defined and the sets of extents, intents and modi do not form closure systems as in the dyadic case (see Observation 2.1.6). Considering that there are a lot of data collections that map perfectly to a triadic representation, especially in the field of collaborative tagging and folksonomies [Jäschke *et al.*, 2008], we consider that it is essential to have a navigation paradigm suitable for triadic contexts.

Driven by the previously mentioned practical requirements, we propose in this chapter a new navigation paradigm for triadic conceptual landscapes based on a neighborhood notion arising from appropriately defined dyadic concept lattices. For this purpose, we

intend to locally display a smaller part of the space of triconcepts, instead of displaying all of them at once, and then find an intuitive navigation strategy that allows for moving from one such local view to other adjacent ones. Furthermore, we will formally analyze the properties of this strategy and ultimately suggest algorithms for producing the structures necessary for browsing the space of triconcepts using theoretically well-understood methods. The results presented in this section were published in a conference paper in 2015 [**Rudolph *et al.*, 2015c**].

The navigation strategy we propose makes use of the elegance and the expressive power of dyadic concept lattices. Navigation starts locally, with a triconcept. Herefrom, we fix what we call a *perspective*, i.e., one of the three dimensions (extent, intent or modus) and then collect all so-called *directly reachable triconcepts*. For each perspective, the triconcepts directly reachable via this perspective can be arranged in a dyadic concept lattice, hence navigating among them benefits from all advantages concept lattices are offering. After selecting a directly reachable triconcept, one may change the perspective and move towards another set of reachable triconcepts, exploring again another concept lattice. Given the local character of the navigation, this approach allows to cope with large sets of triconcepts. Moreover, the local navigation strategy discussed in this chapter gives rise to a list of theoretical questions: reachability of all triconcepts, the existence and the number of sets of mutually reachable concepts, their structure and a method to navigate from one to another. Understanding these clusters proves to be not trivial and gives interesting insights about the inherent conceptual structure of triadic data.

3.3.2 Proof of Concept

Before introducing the theoretical aspects of the navigation paradigm we will present a small example aiming at explaining how the local navigation paradigm works in a set of triconcepts. Therefore, we will describe a few navigation steps using different graphical representations, since it is easier to understand the navigation while visualizing the concepts and the relation between them.

For this purpose, we consider the hostels tricontext from Example 2.1.4 and its trilattice diagram as represented by Glodeanu [2013]. The objects of the triadic dataset are hostels, the attributes are services provided by the hostels, while the conditions are Web portals where the hostels can be rated.

In the triadic graphical representation from Figure 3.8 one can observe, as mentioned earlier, that the complexity of the trilattice structure and that of the order diagrams of the extents, intents and modi sets make a global navigation approach quite difficult. On the other hand, the local navigation paradigm avoids such complexity issues by taking

advantage of dyadic context lattices, representing local views of the triadic context.

The local navigation paradigm starts from a triconcept (A_1, A_2, A_3) and the first step is to select one of its dimensions, i.e. the dimension of the extent, intent or modus, as a *perspective*. This perspective (k) is the one chosen for the dyadic projection $\mathbb{K}^{(ij)}_{A_k}$. Moreover, the dyadic projection is the local view for this navigation step. Each dyadic concept in the dyadic projection corresponds to a triadic concept that has either A_k as the third component or a larger set, depending on the maximality constraint. These triconcepts are called *directly reachable* and navigation among them is performed in the underlying dyadic concept lattice.

Let the triconcept $T_1 = (\{g_3, g_4, g_5\}, \{m_0, m_1, m_2, m_3, m_5\}, \{b_1, b_2\})$ be a starting point for the local navigation and consider perspective (3), i.e. modus dimension, for the first step. By projecting along $\{b_1, b_2\}$, we obtain the concept lattice displayed in Figure 3.9, where triconcept T_1 corresponds to the rightmost dyadic concept. The extent and intent of the triconcepts can be read from the concept lattice, while the modus is computed using the corresponding derivation operator $(\cdot)^3$ in the tricontext. It follows, as mentioned previously, that all dyadic concepts correspond to triconcepts, having either the same modus or a larger one. The navigation can be continued herefrom by choosing one of the triconcepts directly reachable from T_1, i.e., one of the concepts of $\mathbb{K}^{(12)}_{\{b_1,b_2\}}$, and a perspective in order to navigate within the new concept lattice. For example, the leftmost concept of this lattice diagram corresponds to the triconcept $T_2 = (\{g_2, g_3, g_4\}, \{m_2, m_3, m_4\}, \{b_1, b_2\})$. By choosing T_2 and perspective (1), i.e. extent dimension, the triconcepts reachable herefrom are represented in Figure 3.10. Here, the extent of the triconcepts is computed using the corresponding derivation operator $(\cdot)^1$ in the tricontext, while intent and modus can be read from the dyadic lattice.

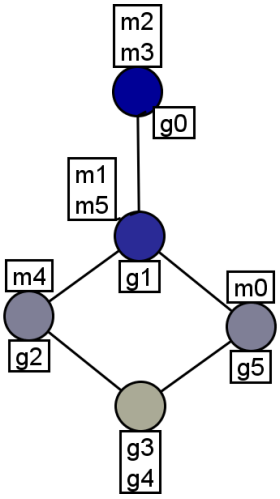


Figure 3.9: Directly reachable triconcepts from T_1 using perspective (3)

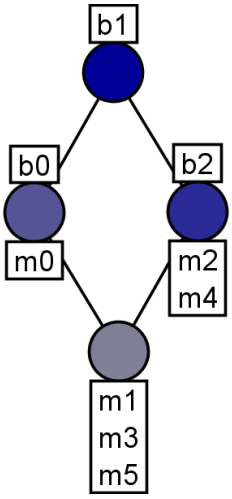


Figure 3.10: Directly reachable triconcepts from T_2 using perspective (1)

This example shows how we can navigate from one triconcept to another and how triconcepts can be clustered according to their reachability. Motivated by this example, we introduce, in the following sections, the theoretical aspects and considerations of the proposed navigation paradigm [**Rudolph *et al.*, 2015c**]. Moreover, we discuss some theoretical questions arising from the described navigation method, like for example whether all concepts are reachable from any starting point.

3.3.3 Reachability Relations among Triconcepts

This section aims to define the exploration paradigm exemplified in the previous section. In order to better understand the connection between the triconcepts and the dyadic contexts used for the local navigation, we present some properties of the triconcepts, which are a direct consequence of Proposition 2.1.5 [**Rudolph *et al.*, 2015c**].

Proposition 3.3.1 *Let* $(A_1, A_2, A_3) \in \mathfrak{T}(\mathbb{K})$ *be a triadic concept.*
Then $(A_1, A_2) \in \mathfrak{B}(\mathbb{K}^{(12)}_{A_3})$.

Proof. Let (A_1, A_2, A_3) be a triconcept. By definition, $\mathbb{K}^{(12)}_{A_3} = (K_1, K_2, Y^{(12)}_{A_3})$, where, for $g \in K_1$ and $m \in K_2$, $(g, m) \in Y^{(12)}_{A_3}$ if and only if $(g, m, b) \in Y, \forall b \in A_3$. We want to prove that $(A_1, A_2) \in \mathfrak{B}(\mathbb{K}^{(12)}_{A_3}) \Leftrightarrow A_1 = A_2^{(1,2,A_3)}$ and $A_2 = A_1^{(1,2,A_3)}$. On the one hand, $A_1^{(1,2,A_3)} = \{m \in K_2 \mid \forall g \in A_1, \forall b \in A_3. (g, m, b) \in Y\}$. On the other, since (A_1, A_2, A_3) is a triconcept, it follows from the triconcept definition that

$A_2 = (A_1 \times A_3)^{(2)} = \{m \in K_2 \mid \forall g \in A_1,\ \forall b \in A_3.\ (g, m, b) \in Y\}$. We conclude that $A_2 = A_1^{(1,2,A_3)}$ and similarly $A_1 = A_2^{(1,2,A_3)}$. □

Proposition 3.3.2 *Let* $(A_1, A_2, A_3) \in \mathfrak{T}(\mathbb{K})$ *be a triadic concept. Let* $(B_1, B_2) \in \mathfrak{B}(\mathbb{K}_{A_3}^{(12)})$. *Then* $(B_1, B_2, (B_1 \times B_2)^{(3)}) \in \mathfrak{T}(\mathbb{K})$.

Proof. From $(B_1, B_2) \in \mathfrak{B}(\mathbb{K}_{A_3}^{(12)})$ it follows that $B_1 = B_2^{(1,2,A_3)}$ and $B_2 = B_1^{(1,2,A_3)}$. Now we can apply Proposition 2.1.5 for $i = 1, j = 2, k = 3, X_1 = B_1, X_3 = A_3$ and hence the computed triconcept is exactly (B_1, B_2, B_3) with $B_3 = (B_1 \times B_2)^{(3)}$. □

Proposition 3.3.1 shows that, by projecting along one of the dimensions, we obtain a formal dyadic context, where the projection of the triconcept is a dyadic concept of the corresponding concept lattice. Moreover, Proposition 3.3.2 proves that every dyadic concept of the projected context generates a triconcept in the tricontext. Hence, given a triconcept (A_1, A_2, A_3), fixing either its extent, intent or modus, gives rise to a dyadic concept lattice, every concept of which can be deterministically turned into a triconcept by computing the missing component using the appropriate derivation operator. Taking this into consideration, we define the reachability relation as follows.

Definition 3.3.1 (Direct reachability) *For* (A_1, A_2, A_3) *and* (B_1, B_2, B_3) *triadic concepts of the tricontext* $\mathbb{K} = (K_1, K_2, K_3, Y)$, *we say that* (B_1, B_2, B_3) *is* directly reachable *from* (A_1, A_2, A_3) using perspective (1) *and we write* $(A_1, A_2, A_3) \prec_1 (B_1, B_2, B_3)$ *if and only if* $(B_2, B_3) \in \mathfrak{B}(\mathbb{K}_{A_1}^{(23)})$. *Analogously, we can define direct reachability for perspectives* (2) *and* (3).

We say that (B_1, B_2, B_3) *is* directly reachable *from* (A_1, A_2, A_3) *if it is directly reachable using at least one of the three perspectives, that is, formally:*

$$(A_1, A_2, A_3) \prec (B_1, B_2, B_3) \Leftrightarrow$$
$$[(A_1, A_2, A_3) \prec_1 (B_1, B_2, B_3)] \vee [(A_1, A_2, A_3) \prec_2 (B_1, B_2, B_3)] \vee [(A_1, A_2, A_3) \prec_3 (B_1, B_2, B_3)].$$

By Proposition 3.3.1, two triconcepts having the same extent, intent, or modus are always mutually directly reachable. Hence, in a trilattice diagram, all triconcepts aligned on the same line (i.e., being equivalent with respect to one of the three preorders) are mutually directly reachable:

Proposition 3.3.3 *Let* $(A_1, A_2, A_3), (B_1, B_2, B_3)$ *be two triconcepts of the context* $\mathbb{K} = (K_1, K_2, K_3, Y)$. *If* $A_i = B_i$ *for an* $i \in \{1, 2, 3\}$ *then* $(A_1, A_2, A_3) \prec_i (B_1, B_2, B_3)$ *and* $(B_1, B_2, B_3) \prec_i (A_1, A_2, A_3)$.

Definition 3.3.2 (Reachability) *We define the* reachability relation *between two triconcepts as being the transitive closure of the direct reachability relation* $\prec$. *We denote this relation by* $\lhd$.

Definition 3.3.3 (Reachability cluster) *The equivalence class of a triconcept* (A_1, A_2, A_3) *with respect to the preorder* $\triangleleft$ *on* $\mathfrak{T}(\mathbb{K})$ *will be called a* reachability cluster *and denoted by* $[(A_1, A_2, A_3)]$. *The corresponding equivalence relation is denoted by* $\sim$.

Intuitively, the reachability cluster of (A_1, A_2, A_3) contains all triconcepts which are mutually reachable from (A_1, A_2, A_3). However, when a triconcept $T2$ is reachable from a triconcept $T1$ it is not necessarily the case that $T1$ is also reachable from $T2$, since the reachability relation is not symmetric. The following definitions briefly introduce some notions of directed graphs which are necessary in order to describe the structure of the reachable triconcepts.

Definition 3.3.4 (Directed graph) *A directed graph* $G = (V, E)$ *is an ordered pair comprising a set of vertices* V *and a set* E *of directed edges* $E \subseteq V \times V$.

Definition 3.3.5 (Subgraph) *A graph* $G' = (V', E')$ *is a subgraph of graph* $G = (V, E)$ *if and only if* $V' \subseteq V$ *and* $E' \subseteq E$, *i.e. its vertices, respectively edges, are a subset of the vertices, respectively edges, of the other graph.*

Definition 3.3.6 (Strongly connected) *A directed graph* $G = (V, E)$ *is called strongly connected if between any two vertices* $x, y \in V$ *there is a path from* x *to* y *as well as a path from* y *to* x. *The path is defined as a sequence of distinct edges with the property that the starting point of an edge is the ending point of the previous edge, i.e. the edges connect a sequence of vertices.*

Definition 3.3.7 (Strongly connected component) *A strongly connected component of a directed graph* $G = (V, E)$ *is a strongly connected subgraph which is maximal with this property, i.e. there is no way of adding other vertices* $v \in V$ *or edges* $e \in E$ *which results in a new strongly connected subgraph. The set of strongly connected components of a graph form a partition of the set of vertices* V.

Observation 3.3.1 *Let* $\mathbb{K}$ *be a tricontext and* G *the graph with* $\mathfrak{T}(\mathbb{K})$ *as vertices and the edges given by the direct reachability relation. Then, the reachability clusters of* $\mathbb{K}$ *are identified by the strongly connected components of* G. *Furthermore, for triconcepts* $T_1, T_2 \in \mathfrak{T}(\mathbb{K})$, *we have that* $T_1 \triangleleft T_2$ *if, in graph* G, *there is a directed path from* T_1 *to* T_2. *Hence, we can deduce the transitive closure of the direct reachability relation from the previously defined triconcept graph* G.

The following results provide a better understanding of the reachability clusters and their structure. We prove that there exist triconcepts which are always reachable (Proposition 3.3.4). Moreover, the induced order on the set of reachability clusters always has a greatest element.

Proposition 3.3.4 *Let $\mathbb{K} = (K_1, K_2, K_3, Y)$ be a tricontext. Then, the trivial/non-proper triconcepts $\theta_1 = ((K_2 \times K_3)^{(1)}, K_2, K_3)$, $\theta_2 = (K_1, (K_1 \times K_3)^{(2)}, K_3)$ and $\theta_3 = (K_1, K_2, (K_1 \times K_2)^{(3)})$ are always reachable. Moreover, they are always directly reachable.*

Proof. Let us first assume that $(K_2 \times K_3)^{(1)} = (K_1 \times K_3)^{(2)} = (K_1 \times K_2)^{(3)} = \emptyset$. Let $(A_1, A_2, A_3) \in \mathfrak{T}(\mathbb{K})$ be a triconcept. Using perspective (3), we navigate in $\mathfrak{B}(\mathbb{K}_{A_3}^{(12)})$. The greatest and the lowest elements of $\mathbb{K}_{A_3}^{(12)}$ are $(K_1, \emptyset)$ and $(\emptyset, K_2)$, respectively. But the corresponding triconcepts of $(K_1, \emptyset)$ and $(\emptyset, K_2)$ are $(K_1, \emptyset, K_3)$, respectively $(\emptyset, K_2, K_3)$. It follows that $(A_1, A_2, A_3) \triangleleft \theta_1 = (\emptyset, K_2, K_3)$ and $(A_1, A_2, A_3) \triangleleft \theta_2 = (K_1, \emptyset, K_3)$. By choosing one of the other two perspectives, θ_3 is directly reached from (A_1, A_2, A_3).

In particular, if $(A_1, A_2, A_3) = \theta_1$, then the trivial triconcepts θ_2 and θ_3 are reachable using perspective (1).

If $(K_2 \times K_3)^{(1)} = M_1 \neq \emptyset$, then the lowest element of $\mathbb{K}_{A_3}^{(12)}$ would be (M_1, K_2) instead be of $(\emptyset, K_2)$ and again it follows that all the trivial triconcepts are directly reachable. It works analogously if $(K_1 \times K_3)^{(2)} \neq \emptyset$ or $(K_1 \times K_2)^{(3)} \neq \emptyset$. □

Corollary 3.3.4.1 *The ordered set of equivalence classes $(\mathfrak{T}(\mathbb{K})/\sim, \leq)$ has always a greatest element, the reachability cluster of the trivial concepts. We denote this cluster by ∇.*

Observation 3.3.2 *The same graph G, having the triconcepts $\mathfrak{T}(\mathbb{K})$ as vertices and the edges given by the direct reachability relation, can be used to deduce the order relation between the clusters. In Section 3.3.5 we show that the order relation between clusters is a partial order. If $T_1 \in C_1$ and $T_2 \in C_2$ are two triconcepts from different clusters s.t., in graph G, there is a path from T_1 to T_2, then we have that $C_1 \leq C_2$. Observe that, considering T_1 and T_2 belong to different clusters, in the case that there is a path from T_1 to T_2, we cannot also have a path from T_2 to T_1.*

Proposition 3.3.5 *If (A_1, A_2, A_3) is a triconcept with either $A_1 = K_1$, or $A_2 = K_2$, or $A_3 = K_3$, then $(A_1, A_2, A_3) \in \nabla$.*

Proof. Every trivial concept is directly reachable from (A_1, A_2, A_3), so $(A_1, A_2, A_3) \triangleleft \theta_3$. Let us assume that $A_1 = K_1$. Take now $\theta_3 = (K_1, K_2, (K_1 \times K_2)^{(3)})$ and choose perspective (1). We obtain the context $\mathbb{K}_{K_1}^{(23)} = (K_2, K_3, Y_{K_1}^{(23)})$. We want to prove that $(A_2, A_3) \in \mathfrak{B}(\mathbb{K}_{K_1}^{(23)})$.

We know that $A_2 = (K_1 \times A_3)^{(2)} = \{m \in K_2 \mid \forall g \in K_1, \forall b \in A_3. (g, m, b) \in Y\}$. Also, by definition, ${A_3}^{(2,3,K_1)} = \{m \in K_2 \mid \forall g \in K_1, \forall b \in A_3. (g, m, b) \in Y\}$, hence $A_2 = {A_3}^{(2,3,K_1)}$. Analogously, $A_3 = {A_2}^{(2,3,K_1)}$, so $(A_2, A_3) \in \mathfrak{B}(\mathbb{K}_{K_1}^{(23)})$.

It follows that (A_1, A_2, A_3) and θ_3 are mutually reachable, so they belong to the same cluster, i.e. $(A_1, A_2, A_3) \in \nabla$ □

Remark 3.3.1 *Let $(A_1, A_2, A_3), (B_1, B_2, B_3) \in \mathfrak{T}(\mathbb{K})$ be two triconcepts.*
If $(A_1, A_2, A_3) \in \nabla$ and $(A_1, A_2, A_3) \lhd (B_1, B_2, B_3)$ then $(B_1, B_2, B_3) \in \nabla$. The converse does not hold, i.e. if $(B_1, B_2, B_3) \lhd (A_1, A_2, A_3)$ and $(A_1, A_2, A_3) \in \nabla$ it does not follow that $(B_1, B_2, B_3) \in \nabla$. In fact, this is always true, since we showed earlier that the trivial concepts are reachable from any triconcept, hence all triconcepts in ∇ are reachable from any other triconcept.

Intuitively, this remark suggests that a tricontext can have more than one reachability cluster. Consider, for example, the following triconctext: $K_1 = \{g_1, g_2\}$, $K_2 = \{m_1, m_2\}$, $K_3 = \{b_1, b_2\}$ with $Y = \{(g_1, m_1, b_1)\}$. In this context there are exactly two reachability clusters, $\nabla = \{\theta_1, \theta_2, \theta_3\}$ and $\{(g_1, m_1, b_1)\}$.

In the next section, we will discuss in more detail the number of reachability clusters of a tricontext, but first, we will present some examples of artificial tricontexts that show that, in general, triconcepts might be structured in more than one cluster.

Example 3.3.1 (Tricontexts with more than two clusters)

b1	*m1*	*m2*
g1	×	
g2		×

b2	*m1*	*m2*
g1		
g2		×

b3	*m1*	*m2*
g1		
g2		

The concepts are partitioned in clusters the following way:
$C_1 = \{(\{g_1\}, \{m_1\}, \{b_1\})\}$
$C_2 = \{(\{g_2\}, \{m_2\}, \{b_1, b_2\})\}$
$C_3 = \{(\{g_1, g_2\}, \{m_1, m_2\}, \emptyset), (\{g_1, g_2\}, \emptyset, \{b_1, b_2, b_3\}), (\emptyset, \{m_1, m_2\}, \{b_1, b_2, b_3\})\}$
The order relation on the clusters is $C_1 \leq C_2 \leq C_3$.

b1	*m1*	*m2*	*m3*
g1	×		
g2			
g3			

b2	*m1*	*m2*	*m3*
g1	×	×	
g2	×	×	
g3			

b3	*m1*	*m2*	*m3*
g1	×	×	×
g2	×	×	×
g3	×	×	×

The concepts are partitioned in clusters the following way:
$C_1 = \{(\{g_1\}, \{m_1\}, \{b_1, b_2, b_3\})\}$

$C_2 = \{(\{g_1, g_2\}, \{m_1, m_2\}, \{b_2, b_3\})\}$
$C_3 = \{(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\}, \{b_3\}), (\{g1, g2, g3\}, \emptyset, \{b_1, b_2, b_3\}),$
$(\emptyset, \{m_1, m_2, m_3\}, \{b_1, b_2, b_3\})\}$
The order relation on the clusters is $C_1 \leq C_2 \leq C_3$.

b1	*m1*	*m2*	*m3*
g1	×		
g2			
g3			

b2	*m1*	*m2*	*m3*
g1	×		
g2		×	
g3			

b3	*m1*	*m2*	*m3*
g1	×		
g2		×	
g3			×

The concepts are partitioned in clusters the following way:
$C_1 = \{(\{g_3\}, \{m_3\}, \{b_3\})\}$
$C_2 = \{(\{g_2\}, \{m_2\}, \{b_2, b_3\})\}$
$C_3 = \{(\{g_1\}, \{m_1\}, \{b_1, b_2, b_3\}), (\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\}, \emptyset),$
$(\{g_1, g_2, g_3\}, \emptyset, \{b_1, b_2, b_3\}), (\emptyset, \{m_1, m_2, m_3\}, \{b_1, b_2, b_3\})\}$
The order relation on the clusters is $C_1 \leq C_2 \leq C_3$.
We can observe that the triconcepts $(\{g_3\}, \{m_3\}, \{b_3\})$ *and* $(\{g_2\}, \{m_2\}, \{b_2, b_3\})$ *have disjoint extents and intents, but* $(\{g_3\}, \{m_3\}, \{b_3\}) \prec_3 (\{g_2\}, \{m_2\}, \{b_2, b_3\})$.

b1	*m1*	*m2*	*m3*
g1	×		
g2			
g3			

b2	*m1*	*m2*	*m3*
g1			
g2			
g3		×	

b3	*m1*	*m2*	*m3*
g1	×	×	
g2			
g3			

The concepts are partitioned in clusters the following way:
$C_1 = \{(\{g_3\}, \{m_2\}, \{b_2\})\}$
$C_2 = \{(\{g_1\}, \{m_1\}, \{b_1.b_3\}), (\{g_1\}, \{m_1.m_2\}, \{b_3\})\}$
$C_3 = \{(\{g_1, g_2\}, \{m_1, m_2\}, \emptyset), (\{g_1, g_2\}, \emptyset, \{b_1, b_2, b_3\}), (\emptyset, \{m_1, m_2\}, \{b_1, b_2, b_3\})\}$
The order relation on the clusters is $C_1 \leq C_2 \leq C_3$.

Example 3.3.2 (Tricontext with two clusters)

$b1$	$m1$	$m2$
$g1$	×	
$g2$		×

$b2$	$m1$	$m2$
$g1$		
$g2$		×

$b3$	$m1$	$m2$
$g1$		
$g2$	×	

The concepts are partitioned in clusters the following way:
$C_1 = \{(\{g_1\}, \{m_1\}, \{b_1\}), (\{g_2\}, \{m_2\}, \{b_1, b_2\}), (\{g_2\}, \{m_1\}, \{b_3\})\}$
$C_2 = \{(\{g_1, g_2\}, \{m_1, m_2\}, \emptyset), (\{g_1, g_2\}, \emptyset, \{b_1, b_2, b_3\}), (\emptyset, \{m_1, m_2\}, \{b_1, b_2, b_3\})\}$
The order relation between the clusters is $C_1 \leq C_2$.

Example 3.3.3 (Tricontexts with one cluster)

$b1$	$m1$	$m2$	$m3$
$g1$	×	×	×
$g2$			
$g3$			

$b2$	$m1$	$m2$	$m3$
$g1$	×	×	
$g2$	×	×	
$g3$			

$b3$	$m1$	$m2$	$m3$
$g1$	×		
$g2$	×		
$g3$	×		

The concepts are the following:
$C = \{(\{g_1\}, \{m_1\}, \{b_1, b_2, b_3\}), (\{g_1, g_2, g_3\}, \{m_1\}, \{b_3\}), (\{g_1, g_2\}, \{m_1, m_2\}, \{b_2\}),$
$(\{g_1\}, \{m_1, m_2\}, \{b_1, b_2\}), (\{g_1, g_2\}, \{m_1\}, \{b_2, b_3\}), (\{g_1\}, \{m_1, m_2, m_3\}, \{b_1\}),$
$(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\}, \emptyset), (\{g_1, g_2, g_3\}, \emptyset, \{b_1, b_2, b_3\}),$
$(\emptyset, \{m_1, m_2, m_3\}, \{b_1, b_2, b_3\})\}$

$b1$	$m1$	$m2$	$m3$
$g1$	×		
$g2$			
$g3$			

$b2$	$m1$	$m2$	$m3$
$g1$	×	×	
$g2$	×		
$g3$			

$b3$	$m1$	$m2$	$m3$
$g1$	×	×	×
$g2$	×	×	
$g3$	×		

The concepts are the following:
$C = \{(\{g_1\}, \{m_1\}, \{b_1, b_2, b_3\}), (\{g_1, g_2\}, \{m_1\}, \{b_2, b_3\}), (\{g_1, g_2\}, \{m_1, m_2\}, \{b_3\}),$
$(\{g_1\}, \{m_1, m_2\}, \{b_2, b_3\}), (\{g_1, g_2, g_3\}, \{m_1\}, \{b_3\}), (\{g_1\}, \{m_1, m_2, m_3\}, \{b_3\}),$
$(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\}, \emptyset), (\{g_1, g_2, g_3\}, \emptyset, \{b_1, b_2, b_3\}),$
$(\emptyset, \{m_1, m_2, m_3\}, \{b_1, b_2, b_3\})\}$

$b1$	$m1$	$m2$	$m3$
$g1$	×		
$g2$			
$g3$			

$b2$	$m1$	$m2$	$m3$
$g1$	×		
$g2$	×	×	
$g3$			

$b3$	$m1$	$m2$	$m3$
$g1$	×		
$g2$	×	×	
$g3$	×	×	×

The concepts are the following:

$C = \{(\{g_1\}, \{m_1\}, \{b_1, b_2, b_3\}), (\{g_1, g_2\}, \{m_1\}, \{b_2, b_3\}), (\{g_1, g_2, g_3\}, \{m_1\}, \{b_3\}),$
$(\{g_2\}, \{m_1, m_2\}, \{b_2, b_3\}), (\{g_2, g_3\}, \{m_1, m_2\}, \{b_3\}), (\{g_3\}, \{m_1, m_2, m_3\}, \{b_3\}),$
$(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\}, \emptyset), (\{g_1, g_2, g_3\}, \emptyset, \{b_1, b_2, b_3\}),$
$(\emptyset, \{m_1, m_2, m_3\}, \{b_1, b_2, b_3\})\}$

$b1$	$m1$	$m2$	$m3$
$g1$	×		
$g2$	×		
$g3$			

$b2$	$m1$	$m2$	$m3$
$g1$	×	×	
$g2$	×	×	
$g3$			

$b3$	$m1$	$m2$	$m3$
$g1$	×	×	
$g2$			
$g3$			

The concepts are the following:

$C = \{(\{g_1\}, \{m_1\}, \{b_1, b_2, b_3\}), (\{g_1, g_2\}, \{m_1\}, \{b_1, b_2\}), (\{g_1\}, \{m_1, m_2\}, \{b_2, b_3\}),$
$(\{g_1, g_2\}, \{m_1, m_2\}, \{b_2\}), (\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\}, \emptyset),$
$(\{g_1, g_2, g_3\}, \emptyset, \{b_1, b_2, b_3\}), (\emptyset, \{m_1, m_2, m_3\}, \{b_1, b_2, b_3\})\}$

$b1$	$m1$	$m2$	$m3$
$g1$	×		
$g2$			
$g3$			

$b2$	$m1$	$m2$	$m3$
$g1$		×	
$g2$			
$g3$		×	

$b3$	$m1$	$m2$	$m3$
$g1$	×	×	
$g2$			
$g3$			

The concepts are the following:

$C = \{(\{g_1\}, \{m_1\}, \{b_1, b_3\}), (\{g_1\}, \{m_2\}, \{b_2, b_3\}), (\{g_1\}, \{m_1, m_2\}, \{b_3\}),$
$(\{g_3\}, \{m_2\}, \{b_2\}), (\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\}, \emptyset), (\{g_1, g_2, g_3\}, \emptyset, \{b_1, b_2, b_3\}),$
$(\emptyset, \{m_1, m_2, m_3\}, \{b_1, b_2, b_3\})\}$

This is an example of a tricontext with two concepts that have disjoint extents, intents and modi, but are in the same cluster: $(\{g_1\}, \{m_1\}, \{b_1, b_3\})$ *and* $(\{g_3\}, \{m_2\}, \{b_2\})$

3.3.4 Reachability in Composed Tricontexts

In this section we investigate whether there is a correlation between the dimensions of the tricontext and the number of reachability clusters. For this purpose, we study what happens with the reachability clusters in the case of a composition of tricontexts and we show that there is a way of composing several tricontexts such that the reachability clusters of the composed tricontext coincide with the union of the reachability clusters of the constituents, except for the greatest cluster ∇ [**Rudolph *et al.*, 2015c**].

Definition 3.3.8 (Composition of tricontexts) *Given tricontexts* $\mathbb{K}_1 = (K_1^1, K_2^1, K_3^1, Y^1), \ldots, \mathbb{K}_n = (K_1^n, K_2^n, K_3^n, Y^n)$*, with* K_j^i *and* K_k^i *being disjoint for all* $j \neq k$ *and all* $i \in \{1, 2, 3\}$*, their* composition $\mathbb{K}_1 \uplus \ldots \uplus \mathbb{K}_n$ *is the tricontext* $\mathbb{K} = (K_1, K_2, K_3, Y)$ *with* $K_i = \bigcup_{k=1}^n K_i^k$ *and* $Y = \bigcup_{k=1}^n Y^k$.

Observation 3.3.3 *From the definition of the composed tricontext it follows that there cannot be a triple in the relation* Y *that has elements from more than one of the component contexts, i.e. if* $(g, m, b) \in Y$ *with* $g \in K_1^i, m \in K_2^j, b \in K_3^k$ *then* $i = j = k$.

Proposition 3.3.6 *Let* $(K_1, K_2, K_3, Y) = \mathbb{K}_1 \uplus \ldots \uplus \mathbb{K}_n$ *be a composed tricontext with* $n \geq 2$ *and all* K_j^i *being non-empty. Then* (A_1, A_2, A_3) *is a triconcept of* (K_1, K_2, K_3, Y) *if and only if*

- A_1, A_2, A_3 *are all non-empty and* (A_1, A_2, A_3) *is a triconcept of some* $\mathbb{K}_j$ *or*
- (A_1, A_2, A_3) *is one of* $(\emptyset, K_2, K_3)$ *or* $(K_1, \emptyset, K_3)$ *or* $(K_1, K_2, \emptyset)$.

Proof. "If": First, consider a triconcept (A_1, A_2, A_3) of some $\mathbb{K}_j$ with A_1, A_2, and A_3 nonempty. Now suppose (A_1, A_2, A_3) were not a triconcept of $\mathbb{K}$, i.e., at least one of A_1, A_2, A_3 can be enlarged. Without loss of generality assume some $a \in K_1 \setminus A_1$ with $(A_1 \cup \{a\}) \times A_2 \times A_3 \subseteq Y$. Now, for $a_2 \in A_2$ and $a_3 \in A_3$, we have $(a, a_2, a_3) \in Y$. Observation 3.3.3 implies that $a \in K_j$ and thus $(A_1 \cup \{a\}) \times A_2 \times A_3 \in Y_j$, contradicting the maximality condition of the triconcept (A_1, A_2, A_3) of $\mathbb{K}_j$.

Second, $(A_1, A_2, A_3) = (\emptyset, K_2, K_3)$ is a triconcept of $\mathbb{K}$ if it satisfies the maximality condition, i.e. unless for some a holds $\{a\} \times K_2 \times K_3 \subseteq Y$. Yet this contradicts the construction of Y. The cases of $(K_1, \emptyset, K_3)$ and $(K_1, K_2, \emptyset)$ follow by symmetry.

"Only if": For any triconcept (A_1, A_2, A_3) of (K_1, K_2, K_3, Y) with nonempty A_1, A_2, A_3, we find an $(a_1, a_2, a_3) \in A_1 \times A_2 \times A_3$. By construction, for every such (a_1, a_2, a_3) must exist some j with $a_1 \in K_1^j$ and $a_2 \in K_2^j$ and $a_3 \in K_3^j$. Consequently, $A_1 \subseteq K_1^j$ and $A_2 \subseteq K_2^j$ and $A_3 \subseteq K_3^j$. Moreover, maximality of (A_1, A_2, A_3) in (K_1, K_2, K_3, Y) implies maximality in $\mathbb{K}_j$.

Finally if one of the components of (A_1, A_2, A_3) is empty, the other two must be maximal by definition. □

Proposition 3.3.7 *Let $\mathbb{K} = (K_1, K_2, K_3, Y) = \mathbb{K}_1 \uplus \ldots \uplus \mathbb{K}_n$ with $n \geq 2$ and all K_j^i being non-empty, and $(A_1, A_2, A_3), (B_1, B_2, B_3) \in \mathfrak{T}(\mathbb{K})$. Then (B_1, B_2, B_3) is directly reachable from (A_1, A_2, A_3) in $\mathbb{K}$ if and only if*

- *they are triconcepts of the same $\mathbb{K}_j$ and (B_1, B_2, B_3) is directly reachable from (A_1, A_2, A_3) in $\mathbb{K}_j$ or*
- *one of B_1, B_2, B_3 is empty, i.e. (B_1, B_2, B_3) is one of the trivial triconcepts.*

Proof. "If": First assume (B_1, B_2, B_3) is directly reachable from (A_1, A_2, A_3) in $\mathbb{K}_j$ and both are are triconcepts of the same $\mathbb{K}_j$. Without loss of generality let (1) be the corresponding perspective. Then $A_1 \subseteq B_1$. Moreover, none of A_1, A_2, A_3 is empty, otherwise (A_1, A_2, A_3) cannot be a triconcept of $\mathbb{K}_j$ due to Proposition 3.3.6. We find that $(B_2, B_3) \in \mathfrak{B}(\mathbb{K}_{j\,A_1}^{(23)})$. However, due to the composition process, $\mathbb{K}_{A_1}^{(23)} = \mathbb{K}_{j\,A_1}^{(23)}$. This implies $(B_2, B_3) \in \mathfrak{B}(\mathbb{K}_{A_1}^{(23)})$, thus (B_1, B_2, B_3) is directly reachable from (A_1, A_2, A_3) in $\mathbb{K}$.

Next, assume that one of B_1, B_2, B_3 is empty and without loss of generality let $B_1 = \emptyset$. By Proposition 3.3.6, this entails $B_2 = K_2$ and $B_3 = K_3$. Then $(\emptyset, K_3) \in \mathfrak{B}(\mathbb{K}_{A_2}^{(13)})$ whenever $A_2 \neq \emptyset$ and $(\emptyset, K_2) \in \mathfrak{B}(\mathbb{K}_{A_3}^{(12)})$ whenever $A_3 \neq \emptyset$ (it is not possible that $A_2 = \emptyset = A_3$), therefore $(A_1, A_2, A_3) \prec (B_1, B_2, B_3)$ holds in $\mathbb{K}$.

"Only if": Assume $(A_1, A_2, A_3) \prec_i (B_1, B_2, B_3)$ in $\mathbb{K}$ and all of B_1, B_2, B_3 are nonempty. Without loss of generality assume $i = 1$, i.e., $(B_2, B_3) \in \mathfrak{B}(\mathbb{K}_{A_1}^{(23)})$. Proposition 3.3.6 implies that (B_1, B_2, B_3) must be a triconcept of some $\mathbb{K}_j$. Then, due to $\emptyset \neq A_1 \subseteq B_1 \subseteq K_1^j$ we find that (A_1, A_2, A_3) cannot be a trivial triconcept, thus it is a triconcept of $\mathbb{K}_j$. Then $(B_2, B_3) \in \mathfrak{B}(\mathbb{K}_{A_1}^{(23)})$ implies $(B_2, B_3) \in \mathfrak{B}(\mathbb{K}_{j\,A_1}^{(23)})$ thus $(A_1, A_2, A_3) \prec_1 (B_1, B_2, B_3)$ holds in $\mathbb{K}_j$. □

Corollary 3.3.7.1 *Let $\mathbb{K} = (K_1, K_2, K_3, Y) = \mathbb{K}_1 \uplus \ldots \uplus \mathbb{K}_n$ with $n \geq 2$ and all K_j^i being non-empty. Then (B_1, B_2, B_3) is reachable from (A_1, A_2, A_3) in $\mathbb{K}$ iff*

- *they are triconcepts of the same $\mathbb{K}_j$ and (B_1, B_2, B_3) is reachable from (A_1, A_2, A_3) in $\mathbb{K}_j$ or*
- *one of B_1, B_2, B_3 is empty.*

Proof. This is a straightforward consequence of the previous proposition and the fact that all trivial triconcepts (those having one empty component) are together in the maximal cluster. □

Using the above results, we investigate if there is any correlation between the cardinality of the three sets of a tricontext and the number of the reachability clusters we obtain. The first observation was that we can find qubic tricontexts with $|K_1| = |K_2| = |K_3| = n$, where the number of clusters equals $n + 1$. This observation is formally described in the following Proposition.

Proposition 3.3.8 *Let $\mathbb{K} = (K_1, K_2, K_3, Y)$ be a tricontext of size $n \times n \times n$ with $K_1 = \{k_i^1 \mid 1 \leq i \leq n\}$, $K_2 = \{k_i^2 \mid 1 \leq i \leq n\}$, $K_3 = \{k_i^3 \mid 1 \leq i \leq n\}$. Let the relation Y be the spatial main diagonal of the tricontext, meaning that a triple $(k_i^1, k_j^2, k_l^3) \in Y \Leftrightarrow i = j = k$. Then there are n+1 clusters, namely n minimal clusters and the maximal cluster.*

Proof. Considering Proposition 3.3.7, the conclusion is immediate, since $\mathbb{K} = (\{k_1^1\}, \{k_1^2\}, \{k_1^3\}, \{(k_1^1, k_1^2, k_1^3)\}) \uplus \ldots \uplus (\{k_n^1\}, \{k_n^2\}, \{k_n^3\}, \{(k_n^1, k_n^2, k_n^3)\})$ □

Based on this example, we assume next that the number of clusters is bounded by the minimal dimension of the tricontext plus one. However, this assumption is contradicted by the following example.

Example 3.3.4 *Consider the following $4 \times 6 \times 6$ tricontext $\mathbb{K}_{466}$.*

b_1	m_1	m_2	m_3	m_4	m_5	m_6
g_1	×					
g_2		×				
g_3			×			
g_4						
g_5						
g_6						

b_2	m_1	m_2	m_3	m_4	m_5	m_6
g_1	×					
g_2						
g_3						
g_4				×		
g_5					×	
g_6						

b_3	m_1	m_2	m_3	m_4	m_5	m_6
g_1						
g_2		×				
g_3						
g_4				×		
g_5						
g_6						×

b_4	m_1	m_2	m_3	m_4	m_5	m_6
g_1						
g_2						
g_3			×			
g_4						
g_5					×	
g_6						×

Besides the maximal cluster, we have six minimal ones which are all singletons consisting of the following triconcepts:

$C_1 = (\{g_1\},\{m_1\},\{b_1, b_2\})$,
$C_2 = (\{g_2\},\{m_2\},\{b_1, b_3\})$,
$C_3 = (\{g_3\},\{m_3\},\{b_1, b_4\})$,
$C_4 = (\{g_4\},\{m_4\},\{b_2, b_3\})$,
$C_5 = (\{g_5\},\{m_5\},\{b_2, b_4\})$,
$C_6 = (\{g_6\},\{m_6\},\{b_3, b_4\})$.
Hence, the $4 \times 6 \times 6$ *tricontext has the smallest dimension equal to* 4*, but contains* 7 *reachability clusters.*

Another assumption that follows naturally from the previous example is that the number of cluster does not exceed the maximal dimension of the tricontext plus one. This assumption, however, is also disproven by the following example.

Example 3.3.5 *Given the tricontext* $\mathbb{K}_{466} = (G, M, B, Y)$ *from Example 3.3.4 and considering that the sets of objects, attributes and conditions are disjoint (for the purpose of composition), we define* $\mathbb{K}_{646} = (B, G, M, \{(b, g, m)|(g, m, b) \in Y\})$ *as well as* $\mathbb{K}_{664} = (M, B, G, \{(m, b, g)|(g, m, b) \in Y\})$.
Intuitively, we obtain $\mathbb{K}_{646}$ *and* $\mathbb{K}_{664}$ *by rotating* $\mathbb{K}_{466}$ *twice. We now let* $\mathbb{K}_{16^3} = \mathbb{K}_{466} \uplus \mathbb{K}_{646} \uplus \mathbb{K}_{664}$ *be the* $16 \times 16 \times 16$ *context built by composing the three. Combining Example 3.3.4 with Corollary 3.3.7.1, we obtain that* $\mathbb{K}_{16^3}$ *has 19 clusters, namely the maximal one and* $6 + 6 + 6 = 18$ *minimal ones.*

Remark 3.3.2 *The issue of whether the total number of clusters or the number of minimal clusters is bounded and what could be an estimation of that bound remains an open question, since our initial conjectures about upper bounds had to be refuted by counterexamples, which nevertheless provided some interesting structural insights. As of yet, the only (and trivial) upper bound for the number of reachability clusters is the number of triconcepts, which may be exponential in the size of the tricontext. We, however, still conjecture that there is a polynomial bound.*

3.3.5 Properties of Reachability Clusters

This section is devoted to the study of several properties of reachability clusters. With this purpose, we study the dyadic context having the set of triconcepts as object and attribute set with the reachability relation as incidence relation [**Rudolph *et al.*, 2015c**]. This dyadic context is formally described in the next definition.

Definition 3.3.9 (Dyadic context of reachability)
Let $\mathbb{K} = (K_1, K_2, K_3, Y)$ *be a triadic context. Then we denote with* $\mathbb{K}_{\triangleleft} = (\mathfrak{T}(\mathbb{K}), \mathfrak{T}(\mathbb{K}), \triangleleft)$ *the formal context of triconcepts with the reachability relation.*

We observe that the concepts of $\mathbb{K}_{\triangleleft}$ are exactly the pairs (M, N), whith M, N being sets of triconcepts, having the property that every triconcept from N is reachable from any triconcept of M and (M, N) is maximal with this property. In what follows, we study some properties of the reachability clusters using the reachability context $\mathbb{K}_{\triangleleft}$.

Proposition 3.3.9 *Let $(A_1, A_2, A_3), (B_1, B_2, B_3) \in \mathfrak{T}(\mathbb{K})$ be two triconcepts for which $(A_1, A_2, A_3) \prec_3 (B_1, B_2, B_3)$. Then $Y^{12}_{B_3} \subseteq Y^{12}_{A_3}$.*

Proof. Let $(g, m) \in Y^{12}_{B_3}$. Then, for every $b \in B_3$, we have $(g, m, b) \in Y$. From $(A_1, A_2, A_3) \prec_3 (B_1, B_2, B_3)$ we have that $A_3 \subseteq B_3$, so for every $b \in A_3, (g, m, b) \in Y$, hence $(g, m) \in Y^{12}_{A_3}$. □

Proposition 3.3.10 *Let $(M, N) \in \mathbb{K}_{\triangleleft}$ be a concept and denote by $C = M \cap N$. If $C \neq \emptyset$, then C is a reachability cluster.*

Proof. Let $T_1, T_2 \in C$ be two triconcepts. Since $C = M \cap N$, on the one side we have that $T_1 \in M, T_2 \in N$, so $T_1 \triangleleft T_2$ and on the other $T_1 \in N, T_2 \in M$, so $T_2 \triangleleft T_1$. It follows that all triconcepts from C are mutually reachable and there exists a reachability cluster C' s.t. $C \subseteq C'$. Assume that $C \neq C'$ and let $T_3 \in C' \setminus C$. From $T_1 \in C \subseteq C'$ we deduce that $T_1 \triangleleft T_3$ and $T_3 \triangleleft T_1$. From the transitivity of the reachability relation and the fact that (M, N) is a concept, it follows that $T_3 \triangleleft T_1 \triangleleft T, \forall T \in N$ and $T \triangleleft T_1 \triangleleft T_3, \forall T \in M$. This contradicts the maximality condition of the dyadic concept (M, N). We conclude that $C = C'$ and the intersection $M \cap N$ is a reachability cluster. □

Observation 3.3.4 *If we denote with $\mathcal{C}$ the set of reachability clusters from $\mathbb{K}$ and with $\mathcal{I} = \{M \cap N \mid (M, N) \in \mathfrak{B}(\mathbb{K}_{\triangleleft}), M \cap N \neq \emptyset\}$ the set of all dyadic concepts of $\mathbb{K}_{\triangleleft}$ having non disjoint extent and intent, then the previous proposition states that $\mathcal{I} \subseteq \mathcal{C}$.*

One might expect that there exists a one-to-one correspondence between dyadic concepts in the context of reachability $\mathbb{K}_{\triangleleft}$ and reachability clusters from $\mathbb{K}$. This would mean that the structure of reachability clusters is a concept lattice. However, the following example shows that there exist dyadic concepts in $\mathbb{K}_{\triangleleft}$, having disjoint extent and intent, meaning they do not correspond to a reachability cluster.

Example 3.3.6 *Consider the following $4 \times 4 \times 4$ tricontext:*

b_1	m_1	m_2	m_3	m_4
g_1	×			
g_2				
g_3		×		
g_4		×		

b_2	m_1	m_2	m_3	m_4
g_1				
g_2				
g_3		×		
g_4		×		

b_3	m_1	m_2	m_3	m_4
g_1			×	
g_2			×	
g_3				
g_4				

b_4	m_1	m_2	m_3	m_4
g_1			×	
g_2			×	
g_3				
g_4				×

We have the following reachability relations among the nontrivial triconcepts:

$(\{g_1\},\{m_1\},\{b_1\}) \lhd (\{g_1,g_2\},\{m_3\},\{b_3,b_4\})$
$(\{g_1\},\{m_1\},\{b_1\}) \lhd (\{g_3,g_4\},\{m_2\},\{b_1,b_2\})$
$(\{g_4\},\{m_4\},\{b_4\}) \lhd (\{g_1,g_2\},\{m_3\},\{b_3,b_4\})$
$(\{g_4\},\{m_4\},\{b_4\}) \lhd (\{g_3,g_4\},\{m_2\},\{b_1,b_2\})$

The dyadic context of reachability $\mathbb{K}_\lhd$ is given by:

	T_1	T_2	T_3	T_4	T_5	T_6	T_7
$T_1 = (\{g_1\},\{m_1\},\{b_1\})$	×		×	×	×	×	×
$T_2 = (\{g_4\},\{m_4\},\{b_4\})$		×	×	×	×	×	×
$T_3 = (\{g_1,g_2\},\{m_3\},\{b_3,b_4\})$			×		×	×	×
$T_4 = (\{g_3,g_4\},\{m_2\},\{b_1,b_2\})$				×	×	×	×
$T_5 = (\{g_1,g_2,g_3,g_4\},\{m_1,m_2,m_3,m_4\},\emptyset)$					×	×	×
$T_6 = (\{g_1,g_2,g_3,g_4\},\emptyset,\{b_1,b_2,b_3,b_4\})$					×	×	×
$T_7 = (\emptyset,\{m_1,m_2,m_3,m_4\},\{b_1,b_2,b_3,b_4\})$					×	×	×

We observe that the dyadic concept $(\{T_1,T_2\},\{T_3,T_4,T_5,T_6,T_7\})$ has disjoint extent and intent.

Intuitively, this suggests that, considering the dyadic concepts of the reachability context $\mathbb{K}_\lhd$, their intersection of extent and intent is either the empty set or a reachability cluster. Conversely, we can show that every reachability cluster from $\mathbb{K}$ corresponds to the intersection of extent and intent of exactly one dyadic concept in $\mathbb{K}_\lhd$.

Proposition 3.3.11 *Let C be a reachability cluster of triconcepts from $\mathbb{K}_\lhd$. Then there exists a concept in $(M,N) \in \mathbb{K}_\lhd$ with $C = M \cap N$.*

Proof. Consider (C'',C') where $()'$ is the derivation operator of the dyadic context $\mathbb{K}_\lhd$. □

Furthermore, there are no two dyadic concepts of $\mathbb{K}_\lhd$ that correspond to the same reachability cluster, i.e. that have the same intersection of extent and intent. This is formally described in the following Proposition.

Proposition 3.3.12 *If* $(M_1, N_1), (M_2, N_2) \in \mathfrak{B}(\mathbb{K}_\triangleleft)$ *are two different concepts of the context* $\mathbb{K}_\triangleleft$ *of reachable triconcepts with* $M_1 \cap N_1 \neq \emptyset$ *and* $M_2 \cap N_2 \neq \emptyset$*, then* $M_1 \cap N_1 \neq M_2 \cap N_2$*.*

Proof. Let $\mathbb{K}_\triangleleft = \{\mathfrak{T}(\mathbb{K}), \mathfrak{T}(\mathbb{K}), I\}$, s.t. $(T_1, T_2) \in I \Leftrightarrow T_1 \triangleleft T_2$ in $\mathbb{K}$, be the dyadic reachability context and $(M_1, N_1), (M_2, N_2) \in \mathfrak{B}(\mathbb{K}_\triangleleft)$ two different dyadic concepts. We assume $M_1 \cap N_1 = M_2 \cap N_2 = P \neq \emptyset$. Since they are different concepts, we can conclude that they have different extents and intents, so $M_1 \neq M_2$ and $N_1 \neq N_2$. It follows that at least one of the extents and one of the intents is larger than P.

If $M_1 \neq P$, $N_1 \neq P$, $M_2 = P$ and $N_2 = P$, it contradicts the fact that $(M_2, N_2) \in \mathfrak{B}(\mathbb{K}_\triangleleft)$ because it is not maximal since it could be extended to (M_1, N_1). We can conclude that at least the extent of one concept and the intent of the other concept are larger than P. Without loss of generality, we assume $M_1 \neq P$ and $N_2 \neq P$. Let $T_1 \in M_1 \setminus P, T_2 \in P, T_3 \in N_2 \setminus P$. Since $T_2 \in P \subseteq N_1$ it follows $(T_1, T_2) \in I \Rightarrow T_1 \triangleleft T_2$. Since $T_2 \in P \subseteq M_1$ it follows $(T_2, T_3) \in I \Rightarrow T_2 \triangleleft T_3$. From the transitivity of the relation $\triangleleft$ we have $T_1 \triangleleft T_3$. Herefrom we conclude that for every $T \in M_1 \setminus P$, we have $(T, T_3) \in I$, but since $P \subseteq M_2$ and $T_3 \in N_2$, we also have that for every $T \in P$, we have $(T, T_3) \in I$. It follows that T_3 is reachable from any triconcept in M_1, so T_3 should be in the intent of the concept (M_1, N_1), i.e. $T_3 \in N_1$. Since $T_3 \in N_2 \Rightarrow T_3 \in N_1 \cap N_2 = P$ which contradicts the fact that we chose $T_3 \in N_2 \setminus P$. Therefore, the two different concepts in $\mathfrak{B}(\mathbb{K}_\triangleleft)$ cannot have the same intersection of the extent and intent. □

We can go further and state that for dyadic concepts of $\mathbb{K}_\triangleleft$, their intersections of extent and intent are not just distinct, but disjoint.

Proposition 3.3.13 *Let* $(M_1, N_1), (M_2, N_2) \in \mathfrak{B}(\mathbb{K}_\triangleleft)$*. Let* $P = M_1 \cap N_1 \neq \emptyset$ *and* $Q = M_2 \cap N_2 \neq \emptyset$*. Then* $P \cap Q = \emptyset$*.*

Proof. This is a direct consequence of Proposition 3.3.10 and the fact that two reachability clusters are disjoint. Obviously if two clusters would have a common triconcept, then it would result due to the transitivity of $\triangleleft$ that they have to be merged into the same cluster.

□

Proposition 3.3.14 *The sets defined in Observation 3.3.4,* $\mathcal{C}$ *and* $\mathcal{I}$*, are equal, i.e.* $\mathcal{C} = \mathcal{I}$*.*

Proof. The first part of the equivalence, $\mathcal{C} \subseteq \mathcal{I}$ was proven in Proposition 3.3.11, while the second part $\mathcal{I} \subseteq \mathcal{C}$ in Proposition 3.3.10. It follows $\mathcal{C} = \mathcal{I}$.

□

Intuitively, the previous Proposition states that there is a one to one correlation between reachability clusters of $\mathbb{K}$ and the nonempty intersections of the extent and the intent of dyadic concepts from the reachability context $\mathbb{K}_\triangleleft$.

Observation 3.3.5 *We can deduce from Proposition 3.3.14 that there is a partial order relation on the cluster set.*

In what follows, we will study object concepts of the dyadic reachability context $\mathbb{K}_\triangleleft$ and their correlation to the reachability clusters.

Proposition 3.3.15 *Let $T_1, T_2 \in \mathfrak{T}(\mathbb{K})$ be two triconcepts of the same cluster. Then $T_1' = T_2'$ with respect to the dyadic derivation operator of the context $\mathbb{K}_\triangleleft$.*

Proof. Assume $T_1' \neq T_2'$ and without loss of generality let $T_3 \in T_2' \setminus T_1'$ be a triconcept. It follows from the derivation definition that $T_2 \triangleleft T_3$. But T_1 and T_2 belong to the same cluster, so $T_1 \triangleleft T_2$ and from the transitivity of $\triangleleft$ we have that $T_1 \triangleleft T_3$. This means that $T_3 \in T_1'$ which is in contradiction with the choice of T_3. We can conclude that $T_1' = T_2'$. □

Observation 3.3.6 *We can deduce from the previous proposition that triconcepts $T \in \mathfrak{T}(\mathbb{K})$ from the same reachability cluster generate the same object concept $\gamma T = (T'', T')$ in $\mathbb{K}_\triangleleft$.*

Proposition 3.3.16 *Let $T \in \mathfrak{T}(\mathbb{K})$ be a triconcept. Then the cluster $[T]$ of T is generated by the object concept $\gamma T = (T'', T')$ of $\mathbb{K}_\triangleleft$ by $T'' \cap T' = [T]$.*

Proof. " $\subseteq$ ": For the first part $T'' \cap T' \subseteq [T]$ consider $T_1 \in T'' \cap T'$. From $T_1 \in T' \Rightarrow T \triangleleft T_1$. From $T_1 \in T'' = (T')' \Rightarrow T_1 \triangleleft T_2, \forall T_2 \in T'$ and it follows that also $T_1 \triangleleft T$. Hence, the two triconcepts T and T_1 are mutually reachable, i.e. $T_1 \in [T]$.

" $\supseteq$ ": For the second part $[T] \subseteq T'' \cap T'$, let $T_3 \in [T]$. It follows that $T \triangleleft T_3$ and $T_3 \triangleleft T$. From $T \triangleleft T_3 \Rightarrow T_3 \in T'$. Moreover, we know that the reachability relation is reflexive, so $T \in T'' \Rightarrow T \triangleleft T_4, \forall T_4 \in T'$. But $T_3 \triangleleft T$ and from the transitivity of $\triangleleft$ it follows that $T_3 \triangleleft T \in T'' \Rightarrow T \triangleleft T_4, \forall T_4 \in T'$, hence $T_3 \in T''$. We conclude that $T_3 \in T' \cap T''$. □

Observation 3.3.7 *We can conclude from Proposition 3.3.15 and Proposition 3.3.16 that clusters are generated by object concepts of $(\mathfrak{T}(\mathbb{K}), \mathfrak{T}(\mathbb{K}), \triangleleft)$. Moreover, after computing one cluster using element $T \in \mathfrak{T}(\mathbb{K})$, we do not have to compute the object concepts of the other elements of the cluster, since they would generate the same cluster. Furthermore, from Proposition 3.3.10 and Proposition 3.3.12 we can conclude that for all the other*

proper concepts $(M, N) \in \mathfrak{B}(\mathbb{K}_\triangleleft)$*, i.e. concepts which are not object concepts, we have* $M \cap N = \emptyset$*. The intuition behind is that we cannot have two different concepts of* $\mathbb{K}_\triangleleft$ *with the same intersection of extent and intent and since all the clusters are generated by the object concepts, any other proper concepts must have disjoint extents and intents.*

We will demonstrate these results on an example, using one of the tricontexts with more than two reachability clusters, presented earlier.

Example 3.3.7 (Generating clusters with object concepts)

b1	*m1*	*m2*	*m3*
g1	×		
g2			
g3			

b2	*m1*	*m2*	*m3*
g1			
g2			
g3		×	

b3	*m1*	*m2*	*m3*
g1	×	×	
g2			
g3			

We recall that the concepts are partitioned in clusters the following way:

$C_1 = \{(\{g_3\}, \{m_2\}, \{b_2\})\}$

$C_2 = \{(\{g_1\}, \{m_1\}, \{b_1.b_3\}), (\{g_1\}, \{m_1.m_2\}, \{b_3\})\}$

$C_3 = \{(\{g_1, g_2\}, \{m_1, m_2\}, \emptyset), (\{g_1, g_2\}, \emptyset, \{b_1, b_2, b_3\}), (\emptyset, \{m_1, m_2\}, \{b_1, b_2, b_3\})\}$

In order to compute $\mathbb{K}_\triangleleft$ *we denote all the triconcepts as follows:*

$T_1 = (\{g_3\}, \{m_2\}, \{b_2\})$

$T_2 = (\{g_1\}, \{m_1\}, \{b_1.b_3\})$

$T_3 = (\{g_1\}, \{m_1.m_2\}, \{b_3\})$

$T_4 = (\{g_1, g_2\}, \{m_1, m_2\}, \emptyset)$

$T_5 = (\{g_1, g_2\}, \emptyset, \{b_1, b_2, b_3\})$

$T_6 = (\emptyset, \{m_1, m_2\}, \{b_1, b_2, b_3\})$

Then, we represent $\mathbb{K}_\triangleleft$*:*

$\mathbb{K}_\triangleleft$	T_1	T_2	T_3	T_4	T_5	T_6
T_1	×	×	×	×	×	×
T_2		×	×	×	×	×
T_3		×	×	×	×	×
T_4				×	×	×
T_5				×	×	×
T_6				×	×	×

Now we compute the object concepts:

$\mu T_1 = (\{T_1\}, \{T_1, T_2, T_3, T_4, T_5, T_6\})$

$\mu T_2 = \mu T_3 = (\{T_2, T_3\}, \{T_2, T_3, T_4, T_5, T_6\})$

$\mu T_4 = \mu T_5 = \mu T_6 = (\{T_1, T_2, T_3, T_4, T_5, T_6\}, \{T_4, T_5, T_6\})$

When computing the intersection of their extent and intent we observe that for μT_1 we obtain cluster C_1, for μT_2 and μT_3 we obtain cluster C_2 and for μT_4, μT_5 and μT_6 we obtain cluster C_3.

The results presented in this section give rise to a display method of all reachability clusters, along with a navigation support in a concept lattice, by highlighting the object concepts and ignoring all the others.

3.3.6 Exploration Strategy

Considering the theoretical aspects introduced in the previous sections, we propose a strategy for navigating among triconcepts inside a reachability cluster as well as between clusters [**Rudolph *et al.*, 2015c**]. The purpose of this approach is to obtain a tool that can be used for navigation and local visualization of a triadic context. Basically, starting from a triconcept, a navigation step consists of navigating to another directly reachable triconcept. By following a navigation path of several navigation steps one can explore the triadic conceptual knowledge landscape.

The only preprocessing step necessary for the navigation is to compute all triconcepts, for example using `Trias` [Jäschke *et al.*, 2006]. Once we have the triconcept set, one can choose a triconcept as a starting point and navigate by choosing one perspective. The local navigation paradigm is described by the following steps:

- choose a triconcept $T = (A_1, A_2, A_3)$ and a perspective (i) with $i \in \{1, 2, 3\}$
- compute the derived context $\mathbb{K}_{A_i}^{(jk)}$ of the triadic relation with $j, k \in \{1, 2, 3\} \setminus \{i\}$ s.t. $j < k$
- generate the concept lattice of $\mathbb{K}_{A_i}^{(jk)}$
- attach as labels to the dyadic concepts in the lattice the corresponding triconcepts by adding the third component
- the user can choose one of the triconcepts that are represented by the nodes in the dyadic lattice as a next step

For computing the derived context $\mathbb{K}_{A_i}^{(jk)}$ one must select from the triadic relation the pairs of elements $(a_j, a_k) \in A_j \times A_k$ which are in relation with all elements from A_i, i.e. $(a_i, a_j, a_k) \in Y, \forall a_i \in A_i$, assuming without loss of generality that $i < j < k$. Afterwards, the third step consisting of generating the concept lattice of the derived context

can be done using one of the existing tools for dyadic formal concept analysis, for example the `ToscanaJ` suite [Becker *et al.*, 2002; Becker and Correia, 2005]. In the next step, assuming w.l.o.g. that $i < j < k$, for a dyadic concept (B_j, B_k) of the derived context $\mathbb{K}_{A_i}^{(jk)}$ we must identify the corresponding triconcept $(B_i, B_j, B_k) \in \mathfrak{T}(\mathbb{K})$. Theoretically, this would be done by using the corresponding derivation operator to compute the third component of a triconcept. However, considering we have already computed all triconcepts it is more efficient to select the triconcept having the two components B_j and B_k from the triconcept set.

The previously described local navigation and visualization paradigm solves the problem of navigation in triadic contexts. However, it is sometimes helpful to have an overall view of the navigation clusters in order to understand whereto one can navigate, since we have seen that not every triconcept can be reached from every other triconcept, although this seems to be the case in most practical scenarios. With that purpose, we use the lattice structure of the dyadic reachability context $\mathbb{K}_{\triangleleft} = (\mathfrak{T}(\mathbb{K}), \mathfrak{T}(\mathbb{K}), \triangleleft)$ defined in Section 3.3.5 as follows:

- compute the direct reachability relation between triconcepts
- compute the transitive closure of the direct reachability relation
- represent the dyadic lattice of clusters

For the first step we can use Algorithm 3.1 that outputs whether triconcept (B_1, B_2, B_3) is directly reachable from triconcept (A_1, A_2, A_3) or not. The derivations used in the description of the algorithm are the simple dyadic derivations and the index was added just to highlight that each dyadic derivation corresponds to a different dyadic context. For example, $(B_2)'_{P_e} = B_3$ uses the dyadic derivation operator of the context P_e.

Listing 3.1: Procedure directlyReachable((A_1, A_2, A_3), (B_1, B_2, B_3))

If $A_1 = B_1$ **or** $A_2 = B_2$ **or** $A_3 = B_3$ **then**
    Return true

If $A_1 \subset B_1$ **then**
    $P_e = \mathbb{K}^{(23)}_{A_1}$
    If $(B_2)'_{P_e} = B_3$ **and** $(B_3)'_{P_e} = B_2$ **then**
        Return true

If $A_2 \subset B_2$ **then**
    $P_i = \mathbb{K}^{(13)}_{A_2}$
    If $(B_1)'_{P_e} = B_3$ **and** $(B_3)'_{P_e} = B_1$ **then**
        Return true

If $A_3 \subset B_3$ **then**
    $P_m = \mathbb{K}^{(12)}_{A_3}$
    If $(B_1)'_{P_e} = B_2$ **and** $(B_2)'_{P_e} = B_1$ **then**
        Return true

Return false

For the second step, the transitive closure of the direct reachability relation can be computed using one of the existing algorithms: `Warshall` algorithm, `Warren` algorithm, `Schnorr`'s algorithm, `Schmitz`'s algorithm, `Blocked Warshall` algorithm, `Blocked Warren` algorithm and many others [Agrawal and Jagadish, 1987].

After computing the reachability context $\mathbb{K}_\triangleleft$, we can compute the clusters and the partial order on the cluster set using the concept lattice of $\mathbb{K}_\triangleleft$. It follows from Propositions 3.3.10, 3.3.11 and 3.3.12 that each reachability cluster is uniquely identified by the intersection of extent and intent of exactly one dyadic concept of $\mathbb{K}_\triangleleft$. Hence, we can compute the concept lattice of $\mathbb{K}_\triangleleft$ and label each node with the corresponding cluster when the corresponding intersection of extent and intent is not empty. In the obtained lattice one can visualize the partial order between the clusters.

For example, the lattice corresponding to the reachability context $\mathbb{K}_\triangleleft$ from Example 3.3.7 is represented in Figure 3.11. In this lattice, the upper concept corresponds to cluster $C_3 = \{T_4, T_5, T_6\}$, the one in the middle to cluster $C_2 = \{T_2, T_3\}$ and the lower concept to cluster $C_1 = \{T_1\}$. The partial order between the clusters can be read from the lattice the following way: if one can navigate from cluster C_1 to cluster C_2 going upwards,

then $C_1 \leq C_2$.

Alternatively, we can make use of the directed graph having the triconcepts as vertices and the edges given by the direct reachability relation. Here, we can identify the reachability clusters, as well as deduce the transitive closure of the direct reachability relation as described in Observation 3.3.1. Furthermore, we can compute the partial order relation on the cluster set (see Observation 3.3.2). However, this dos not give us the lattice representation of clusters, hence a formal concept analysis tool has to be used if we want to obtain a visualization of the cluster structure.

In conclusion, to support exploration through the dataset, we suggest using both the local visualization method for triconcepts and the visualization of the cluster structure as follows. We compute the lattice showing the cluster structure at the beginning of the navigation and make it available to the user throughout the whole exploration process. Then, at each step of the navigation, in the dyadic lattice showing the possible next steps from a triconcept T, we highlight all the triconcepts belonging to the same cluster as T. In that way, the user can easily choose to navigate within the same cluster or to a different one. Furthermore, looking at the cluster structure in parallel, the user can navigate more easily towards a potential goal of the navigation.

3.4 An n-adic Navigation Paradigm based on Membership Constraints

3.4.1 Motivation

As we have seen in the previous chapters, formal concept analysis provides a powerful and elegant mathematical tool for understanding and investigating multi-dimensional datasets. Although FCA is based on efficient and intuitive methods for knowledge representation, acquiring and retrieval, the visualization and navigation developed so far have limitations when it comes to datasets of arity higher than two. Therefore, the previous chapter described a navigation method based on conveniently defined local projections of the data in a dyadic setting. In this chapter, we consider a different approach, which focuses on narrowing down the space of "interesting" concepts by applying different constraints on the data. With that purpose, we consider the problem of satisfiability of membership constraints, in order to determine if there is a formal concept that satisfies the given constraints, i.e. whose components include or exclude particular elements. This is an important feature of conceptual knowledge management applications, in order to enable the user to focus on a subset of data he is interested in.

Moreover, visualization of conceptual spaces in higher-adic datasets remained an open problem so far. With this second approach, we try to offer a possible solution to this problem by enabling the visualization of concepts based on elements lists, where the user has the option of choosing elements he is interested in, hence narrowing down the space of concepts. Although this visualization method has a local character, we believe that a holistic graphical representation would be inefficient and too complex, since the number of formal concepts is, in general, very large. Note that the number of concepts may be exponential in the size of the formal context.

In the following chapters, we define membership constraints and the corresponding satisfiability problems, and study the computational complexity of these problems in general and for restricted forms of membership constraints. We perform this analysis for dyadic, triadic and n-adic FCA. After introducing the theoretical aspects, we focus on the implementation of memebership constraints. For that purpose, we present a generic asnwer set programming encoding of membership constraints, allowing us to use ASP tools, which are highly optimized to solve satisfiability problems.

Finally, we describe a navigation paradigm based on membership constraints in order to highlight their importance in data analysis and implement the paradigm using two different strategies. We compare the first strategy based on the previously mentioned ASP encoding to the second approach based on an exhaustive search in the concept set. The navigation paradigm based on ASP has the great advantage of being easily extendable to any n-ary dataset, however, with the corresponding limitations of visualizing multiple dimension on a single screen.

The results presented in this chapter are described partly in a conference paper published in 2015 [**Rudolph *et al.*, 2015a**] and partly in a second paper, which is currently under review [**Rudolph *et al.*, 2016**].

3.4.2 Membership Constraints

In this section, we introduce membership constraints for formal concepts and discuss the computational complexity of different membership constraint satisfiability problems [**Rudolph *et al.*, 2015a**]. In the first subsection we discuss the case of dyadic formal contexts, continuing in the other two subsections with triadic and finally general (n-adic) contexts.

3.4.2.1 Membership Constraints in Dyadic Formal Concept Analysis

A membership constraint on a dyadic formal context is defined as follows:

Definition 3.4.1 *Let $\mathbb{K} = (G, M, I)$ be a dyadic formal context. We define a* **(dyadic) membership constraint** *$\mathbb{C}$ on $\mathbb{K}$ as a quadruple $\mathbb{C} = (G^+, G^-, M^+, M^-)$ with $G^+ \subseteq G$ being the set of* **required objects**, *$G^- \subseteq G$ the set of* **forbidden objects**, *$M^+ \subseteq M$ the set of* **required attributes**, *and $M^- \subseteq M$ the set of* **forbidden attributes**.

A formal concept (A, B) of $\mathbb{K}$ is said to **satisfy** *such a membership constraint if all the following conditions hold:*

- $G^+ \subseteq A$,
- $G^- \cap A = \emptyset$,
- $M^+ \subseteq B$,
- $M^- \cap B = \emptyset$.

Furthermore, a membership constraint is said to be (properly) **satisfiable** *with respect to the context $\mathbb{K}$, if it is satisfied by one of its (proper) formal concepts.*

Similar to the satisfiability problem SAT defined in Chapter 2.2, we define the (general) decision problem of membership constraint satisfiability, denoted MCSAT.

problem MCSAT
input: dyadic formal context $\mathbb{K} = (G, M, I)$,
membership constraint $\mathbb{C} = (G^+, G^-, M^+, M^-)$
output: YES if $\mathbb{C}$ satisfiable with respect to $\mathbb{K}$, NO otherwise.

Next, we study the complexity of MCSAT, considering different types of membership constraints when one or more of the required or forbidden sets are missing, i.e. they are the empty set. In the general case, the complexity of the problem 3MCSAT turns out to be intractable as shown by the following theorem.

Theorem 3.4.1 *MCSAT is* NP*-complete, even when restricting to membership constraints of the form $\mathbb{C} = (\emptyset, G^-, \emptyset, M^-)$.*

Proof. In order to prove that it is in NP, we consider the following statement: after guessing a pair (A, B) from $2^G \times 2^M$ (where 2^G denotes, as usual, the set of all subsets of

G), it can be checked in polynomial time if (A, B) is a formal concept of the context $\mathbb{K}$ and if it satisfies the membership constraint $\mathbb{C}$.

We prove NP-hardness showing that the NP-hard 3SAT problem can be polynomially reduced to our MCSAT problem. Let $\mathcal{L} = \{L_1, \ldots, L_n\}$ be a set of propositional literal sets over the set $\{p_1, \ldots, p_k\}$ of propositional variables, and let $P^+ = \{p_1, \ldots, p_k\}$, $P^- = \{\neg p_1, \ldots, \neg p_k\}$ and $\tilde{P} = \{\tilde{p}_1, \ldots, \tilde{p}_k\}$, where $\tilde{P}$ represents a set of copies for the literals $\{p_1, \ldots, p_k\}$ that helps us impose the necessary conditions. Now define the formal context $\mathbb{K}_\mathcal{L} = (G, M, I)$ with

- $G = \mathcal{L} \cup P^+ \cup P^-$
- $M = P^+ \cup P^- \cup \tilde{P}$
- $I = \{(L_i, m) \mid L_i \in \mathcal{L}, m \in M \setminus L_i\}$
  $\cup \ \mathcal{L} \times \tilde{P}$
  $\cup \ \{(l_1, l_2) \mid l_1, l_2 \in P^+ \cup P^-, l_1 \neq l_2\}$
  $\cup \ \{(p_i, \tilde{p}_j) \mid i \neq j\}$
  $\cup \ \{(\neg p_i, \tilde{p}_j) \mid i \neq j\}$

Furthermore, let $\mathbb{C}_\mathcal{L}$ denote the membership constraint $(\emptyset, \mathcal{L}, \emptyset, \tilde{P})$.

Note that both $\mathbb{K}_\mathcal{L}$ and $\mathbb{C}_\mathcal{L}$ can be computed in polynomial time and are of polynomial size with respect to $|\mathcal{L}|$.

We will now show that $\mathcal{L}$ is satisfiable exactly if $\mathbb{C}$ is satisfiable with respect to $\mathbb{K}$.

"$\Rightarrow$": If $\mathcal{L}$ is satisfiable, then there must be a valuation $v : \{p_1, \ldots p_k\} \to \{\mathit{true}, \mathit{false}\}$ under which $\mathcal{L}$ evaluates to *true*. Let L_v be the set of literals such that $p \in L_v$ whenever $v(p) = \mathit{true}$ and $\neg p \in L_v$ whenever $v(p) = \mathit{false}$. Next, we show that $((P^+ \cup P^-) \setminus L_v, L_v)$ is a formal concept of $\mathbb{K}_\mathcal{L}$. On one hand we have, from the definition of the relation I:

$((P^+ \cup P^-) \setminus L_v)' = \{m \in P^+ \cup P^- \mid m \notin (P^+ \cup P^-) \setminus L_v\} \cup$
$\{\tilde{p} \in \tilde{P} \mid p, \neg p \notin (P^+ \cup P^-) \setminus L_v\} = L_v \cup \emptyset = L_v$.

On the other hand:

$L'_v = \{L_i \mid L_v \subseteq M \setminus L_i\} \cup \{l \in P^+ \cup P^- \mid l \notin L_v\} = \{L_i \mid L_i \cap L_v = \emptyset\} \cup$
$\{l \mid l \in (P^+ \cup P^-) \setminus L_v\} = \emptyset \cup (P^+ \cup P^-) \setminus L_v = (P^+ \cup P^-) \setminus L_v$.

Next, we observe that this formal concept satisfies the membership constraint $\mathbb{C}_\mathcal{L}$, since none of the L_i are contained in its extent and none of the $\tilde{p}_j$ are contained in its intent. Therefore, we have found a concept that proves the satisfiability of $\mathbb{C}_\mathcal{L}$ with respect to the context $\mathbb{K}_\mathcal{L}$.

"$\Leftarrow$": Assume that $\mathbb{C}_\mathcal{L}$ is satisfiable with respect to the context $\mathbb{K}_\mathcal{L}$. Then there must be a formal concept (A, B) of $\mathbb{K}_\mathcal{L}$ satisfying it, i.e. having $\mathcal{L} \cap A = \emptyset$ as well as

$\tilde{P} \cap B = \emptyset$. Observe that A must contain one of p or $\neg p$ for each propositional variable p, since otherwise $\tilde{p} \in B$ would hold. Consequently B cannot contain both p and $\neg p$ for any propositional variable p. Moreover, for every $L_i \in \mathcal{L}$ there must be one $l \in B \cap (P^+ \cup P^-)$ with $l \in L_i$. Next, let $\hat{B} = P^+ \cap B \cup \{\neg p \mid p \notin B\}$. By our observation above, we know that $B \subseteq \hat{B}$ therefore $\hat{B}$ still contains at least one literal from every L_i. On the other hand, $\hat{B}$ directly corresponds to a valuation $v_{\hat{B}} : \{p_1, \dots p_k\} \to \{true, false\}$ mapping p_i to *true* if $p_i \in \hat{B}$ and to *false* if $\neg p_i \in \hat{B}$. Consequently, $v_{\hat{B}}$ is a valuation making φ *true* and hence showing that it is satisfiable. □

In what follows, we demonstrate the reduction of 3SAT to MCSAT on an example, using the 3SAT problem described in Example 2.2.1.

Example 3.4.1 *Consider the* $\mathcal{L} = \{L_1, L_2, L_3\}$ *with* $L_1 = \{r, s, \neg q\}$, $L_2 = \{s, \neg q, \neg r\}$, *and* $L_3 = \{\neg q, \neg r, \neg s\}$. *The corresponding 3SAT problem amounts to checking if* $\varphi_{\mathcal{L}} = (\neg q \vee r \vee s) \wedge (\neg q \vee \neg r \vee s) \wedge (\neg q \vee \neg r \vee \neg s)$ *is satisfiable. However, this 3SAT problem can be reduced to the question if the membership constraint* $(\emptyset, \{L_1, L_2, L_3\}, \emptyset, \{\tilde{q}, \tilde{r}, \tilde{s}\})$ *is satisfiable in the following context:*

	q	r	s	$\neg q$	$\neg r$	$\neg s$	$\tilde{q}$	$\tilde{r}$	$\tilde{s}$
$\{r, s, \neg q\}$	×				×	×	×	×	×
$\{s, \neg q, \neg r\}$	×	×				×	×	×	×
$\{\neg q, \neg r, \neg s\}$	×	×	×				×	×	×
q		×	×	×	×	×		×	×
r	×		×	×	×	×	×		×
s	×	×		×	×	×	×	×	
$\neg q$	×	×	×		×	×		×	×
$\neg r$	×	×	×	×		×	×		×
$\neg s$	×	×	×	×	×		×	×	

This encoding ensures that the intent of the formal concepts corresponds to a valid valuation for the formula $\varphi_{\mathcal{L}}$, *since it will not contain both* p *and* $\neg p$ *for any propositional variable* p. *Moreover, concepts satisfying the membership constraint, will not contain any elements of* $\tilde{P}$ *either. We observe that there is a bijection between the valuations making the formula true and the concepts satisfying the membership constraint. The corresponding concept for the valuation* $v = \{q \mapsto true, r \mapsto false, s \mapsto true\}$, *making* $\varphi_{\mathcal{L}}$ *true, is* $(\{r, \neg q, \neg s\}, \{q, s, \neg r\})$.

Corollary 3.4.1.1 *It follows from the previous theorem that also membership constraints of the form* $(G^+, G^-, \emptyset, M^-)$, $(\emptyset, G^-, M^+, M^-)$, (G^+, G^-, M^+, M^-) *are* NP*-complete.*

When analyzing the problem further, it turns out that the simultaneous presence of forbidden objects and forbidden attributes in the membership constraint is the only reason for the established intractability and in all other cases the complexity is lower. For membership constraints where one of the forbidden sets becomes empty, the complexity drops to AC^0 and, for some cases, the problem even becomes trivially true.

Theorem 3.4.2 *When restricted to membership constraints of the form* $(G^+, \emptyset, M^+, M^-)$ *or* $(G^+, G^-, M^+, \emptyset)$*, MCSAT is in* AC^0*.*

Proof. We prove the statement for constraints of the form $\mathbb{C} = (G^+, \emptyset, M^+, M^-)$, the other case follows by duality. First observe that $((M^+)', (M^+)'')$ is a formal concept of $\mathbb{K}$ and it is subset-maximal with respect to its extent and subset-minimal with respect to its intent among all formal concepts whose intent contains M^+. Therefore, $(G^+, \emptyset, M^+, M^-)$ is satisfiable with respect to $\mathbb{K}$ if and only if it is satisfied by the concept $((M^+)', (M^+)'')$. By definition, this is the case if and ony if:

- (1) $G^+ \subseteq (M^+)'$ and
- (2) $(M^+)'' \cap M^- = \emptyset$.

Condition (1) can be rewritten as $G^+ \times M^+ \subseteq I$, while the second condition is equivalent to the condition that for every $m \in M^-$ there exists some $g \in (M^+)'$ with $(g, m) \notin I$.

We now define a first-order-logic interpretation $\mathcal{I}_{\mathbb{K},\mathbb{C}} = (\Delta, ()^{\mathcal{I}})$ over the unary predicates p_G, p_M, p_{G^+}, p_{M^+}, p_{M^-} and the binary predicate p_I as follows: $\Delta = G \cup M$, for every $X \in \{G, M, G^+, M^+, M^-\}$ we let $p_X^{\mathcal{I}} = X$, and $p_I^{\mathcal{I}} = I$. Obviously, $\mathcal{I}$ is an immediate representation of the MCSAT problem.

Next we observe that condition (1) can be expressed by the first-order formula φ_1 defined as:

$$\forall x, y.(p_{G^+}(x) \wedge p_{M^+}(y) \rightarrow p_I(x, y)),$$

while condition (2) can be expressed by φ_2 defined as:

$$\forall x.p_{M^-}(x) \rightarrow \exists y.(\forall z.(p_{M^+}(z) \rightarrow p_I(y, z)) \wedge \neg p_I(y, x)).$$

Intuitively, the part $\forall z.(p_{M^+}(z) \rightarrow p_I(y, z))$ from φ_2 expresses that $y \in (M^+)'$.

Consequently, satisfiability of $\mathbb{C}$ with respect to $\mathbb{K}$ coincides with the satisfaction of the fixed first-order-logic formula $\varphi = \varphi_1 \wedge \varphi_2$ in $\mathcal{I}_{\mathbb{K},\mathbb{C}}$. It follows from the corresponding result from descriptive complexity theory (see Remark 2.2.1) that the considered restricted version of MCSAT is in AC^0. □

As a consequence, all subclasses of the described membership constraint class from Theorem 3.4.2 are also in AC^0.

Corollary 3.4.2.1 *We can deduce that for membership constraints of the form* $(G^+, G^-, \emptyset, \emptyset)$, $(\emptyset, \emptyset, M^+, M^-)$, $(G^+, \emptyset, M^+, \emptyset)$, $(G^+, \emptyset, \emptyset, M^-)$, $(\emptyset, G^-, M^+, \emptyset)$, $(\emptyset, G^-, \emptyset, \emptyset)$, $(\emptyset, \emptyset, \emptyset, M^-)$, $(G^+, \emptyset, \emptyset, \emptyset)$, $(\emptyset, \emptyset, M^+, \emptyset)$ *the corresponding MCSAT problems are also in* AC^0.

Proof. Considering that these are subclasses of the membership constraint class described in Theorem 3.4.2, a proof would not be necessary. However, in the following, we highlight the fact that membership constraints of these forms can be expressed by the same or similar first-order formulae as in the previous theorem. For the first two membership constraints, we consider the case of $(\emptyset, \emptyset, M^+, M^-)$, the other one follows from duality. Following the same ideas from the previous proof, we deduce that this membership constraint is satisfiable if and only if it is satisfiable by the concept $((M^+)', (M^+)'')$. However, the only condition that needs to be checked now is that $(M^+)'' \cap M^- = \emptyset$, which is identical to condition (2) from the previous proof, hence it can be expressed by the first-order formula φ_2.

Similarly, for membership constraints of the type $(G^+, \emptyset, M^+, \emptyset)$, the only condition that needs to be checked is equivalent to the previous condition (1) and can be expressed by the first-order formula φ_1.

For the membership constraints of the form $(G^+, \emptyset, \emptyset, M^-)$, $(\emptyset, G^-, M^+, \emptyset)$, we analyze the second one, while the other follows from duality. Here, we cannot deduce that $(\emptyset, G^-, M^+, \emptyset)$ is satisfiable if and only if it is satisfied by $((M^+)', (M^+)'')$, since this concept is subset-maximal with respect to its extent among concepts whose intent contains M^+. Therefore, it can be the case that $((M^+)', (M^+)'')$ does not satisfy the constraint, but there is a concept (A, B) with $A \subset (M^+)'$ and $(M^+)'' \subset B$ that satisfies it. However, the trivial concept (M', M) (we know that $M = M''$ since the attribute set can no longer be extended) has the subset-minimal extent of all concepts. So we can deduce that the membership constraint is satisfied if and only it is satisfied by (M', M). Obviously $M^+ \subseteq M$, so the condition that needs to be checked is $M' \cap G = \emptyset$ which is equivalent to the condition that for every $g \in G^-$ there exists some $m \in M$ with $(g, m) \notin I$. This can be expressed by the following first-order formula φ_3:

$$\forall x.p_{G^-}(x) \rightarrow \exists y.(p_M(y) \wedge \neg p_I(x, y)).$$

For the membership constraint of the form $(\emptyset, G^-, \emptyset, \emptyset)$, we use the same trivial concept (M', M) which is minimal with respect to its extent among all concepts. Hence, the membership constraint is satisfied if and only if it is satisfied by the concept (M', M). The condition that needs to be checked is $M' \cap G^- = \emptyset$ which can be expressed by φ_3. The other case follows by symmetry.

Finally, membership constraints of the form $(G^+, \emptyset, \emptyset, \emptyset)$ and $(\emptyset, \emptyset, M^+, \emptyset)$ are satisfied by the formal concepts $((G^+)'', (G^+)')$, respectively$((M^+)', (M^+)'')$, without the need to check additional conditions.

Using the result from descriptive complexity theory, we can conclude the proof that for all types of membership constraints described in this corollary the MCSAT problems are in AC^0. □

Finally, in some cases, namely when only required objects or only required attributes are given, the MCSAT problem becomes trivial.

Theorem 3.4.3 *When restricted to membership constraints of the form* $(G^+, \emptyset, \emptyset, \emptyset)$ *or* $(\emptyset, \emptyset, M^+, \emptyset)$*, MCSAT is trivially true.*

Proof. For both types of membership constraints we observe that we have one formal concepts always satisfying the constraint. This is the case for the formal concept $((G^+)'', (G^+)')$ for the first type of constraint, and $((M^+)', (M^+)'')$ for the second. Hence, the two MCSAT problems become trivially true. □

3.4.2.2 Membership Constraints in Triadic Formal Concept Analysis

Next we define and investigate membership constraints and the corresponding satisfiability problem for the triadic case, which was theoretically described in Section 2.1.3 [**Rudolph *et al.*, 2015a**].

Definition 3.4.2 *Let* $\mathbb{K} = (G, M, B, Y)$ *be a triadic formal context. We define a* **triadic membership constraint** $\mathbb{C}$ *on the tricontext* $\mathbb{K}$ *as a sextuple:*
$\mathbb{C} = (G^+, G^-, M^+, M^-, B^+, B^-)$ *with* $G^+ \subseteq G$ *called* **required objects**, $G^- \subseteq G$ *called* **forbidden objects**, $M^+ \subseteq M$ *called* **required attributes**, $M^- \subseteq M$ *called* **forbidden attributes**, $B^+ \subseteq B$ *called* **required conditions**, *and* $B^- \subseteq B$ *called* **forbidden conditions**.

A triconcept (A_1, A_2, A_3) *of* $\mathbb{K}$ *is said to* **satisfy** *such a membership constraint if all the following conditions hold:*

- $G^+ \subseteq A_1$,
- $G^- \cap A_1 = \emptyset$,
- $M^+ \subseteq A_2$,
- $M^- \cap A_2 = \emptyset$,

- $B^+ \subseteq A_3$,
- $B^- \cap A_3 = \emptyset$.

A triadic membership constraint is said to be (properly) **satisfiable** *with respect to* $\mathbb{K}$*, if it is satisfied by one of its (proper) triconcepts.*

problem TMCSAT
input: tricontext $\mathbb{K} = (G, M, B, Y)$,
triadic membership constraint $\mathbb{C} = (G^+, G^-, M^+, M^-, B^+, B^-)$
output: YES if $\mathbb{C}$ satisfiable with respect to $\mathbb{K}$, NO otherwise.

When studying the complexity of TMCSAT, we observe that in the triadic case, analogously to the dyadic one, the membership constraint satisfiability problem can be NP-complete or in AC^0. However, in the triadic case there are two possible sources for intractability. The first one, namely two nonempty forbidden sets, is tightly related to the NP-complete case observed in dyadic MCSAT problems, while the other one, namely a required and forbidden set of the same type, is tractable for classical FCA and becomes intractable only when considering the triadic case.

Theorem 3.4.4 *TMCSAT is* NP*-complete, even when restricting to triadic membership constraints of the following form:*

- $(\emptyset, G^-, \emptyset, M^-, \emptyset, \emptyset)$, $(\emptyset, G^-, \emptyset, \emptyset, \emptyset, B^-)$, $(\emptyset, \emptyset, \emptyset, M^-, \emptyset, B^-)$,
- $(G^+, G^-, \emptyset, \emptyset, \emptyset, \emptyset)$, $(\emptyset, \emptyset, M^+, M^-, \emptyset, \emptyset)$, $(\emptyset, \emptyset, \emptyset, \emptyset, B^+, B^-)$.

Proof. NP membership is straightforward: after guessing a triple (A_1, A_2, A_3) from $2^G \times 2^M \times 2^B$, it can be checked in polynomial time if (A_1, A_2, A_3) is a triconcept of $\mathbb{K}$ and if it satisfies $\mathbb{C}$.

We proceed by showing hardness for the restricted cases. As mentioned previously, the first type of membership constraints, with two forbidden sets, are tightly related to membership constraints in dyadic contexts. Therefore, we will show NP-hardness for these types by proving that the corresponding NP-complete MCSAT problem can be reduced to a TMCSAT problem.

Given some (dyadic) formal context $\mathbb{K} = (G, M, I)$, we define its triadic version $\mathbb{T}(\mathbb{K}) = (G, M, \{*\}, I \times \{*\})$. Then, the set of all triconcepts of $\mathbb{T}(\mathbb{K})$ is $\{(G, M, \emptyset)\} \cup \{(A_1, A_2, \{*\}) \mid (A_1, A_2)$ concept of $\mathbb{K}\}$. We deduce that every MCSAT problem of the form $(\emptyset, G^-, \emptyset, M^-)$, defined on the context $\mathbb{K}$, can be reduced to the TMCSAT problem

of the form $(\emptyset, G^-, \emptyset, M^-, \emptyset, \emptyset)$, defined on the tricontext $\mathbb{T}(\mathbb{K})$. Since the former problem is NP-complete due to Theorem 3.4.1, the latter must be NP-hard. By symmetry, this argument carries over to constraints of the form $(\emptyset, G^-, \emptyset, \emptyset, \emptyset, B^-)$ and $(\emptyset, \emptyset, \emptyset, M^-, \emptyset, B^-)$. Herefrom, we can conclude that the TMCSAT problem for constraints having two forbidden sets is NP-complete.

Next, we show NP-hardness for constraints of the form $(G^+, G^-, \emptyset, \emptyset, \emptyset, \emptyset)$. In order to prove this, we use a reduction from 3SAT. Given a set $\mathcal{L} = \{L_1, \ldots, L_n\}$ of propositional literal sets over the set $\{p_1, \ldots, p_k\}$ of propositional variables, we define the tricontext $\mathbb{K}_\mathcal{L} = (G, M, B, Y)$ with

- $G = \{*\} \cup \mathcal{L}$,
- $M = \{*, p_1, \ldots, p_k\}$
- $B = \{*, \neg p_1, \ldots, \neg p_k\}$
- $Y = G \times M \times B \setminus \big(\{(*, p_i, \neg p_i) \mid 1 \leq i \leq k\} \cup \{(L_j, p_i, *) \mid p_i \in L_j\}$
  $\cup \{(L_j, *, \neg p_i) \mid \neg p_i \in L_j\}\big)$

Furthermore, let $\mathbb{C}_\mathcal{L}$ denote the membership constraint $(\{*\}, \mathcal{L}, \emptyset, \emptyset, \emptyset, \emptyset)$.

Note that both $\mathbb{K}_\mathcal{L}$ and $\mathbb{C}_\mathcal{L}$ can be computed in polynomial time and are of polynomial size with respect to $|\mathcal{L}|$.

We will now show that $\mathcal{L}$ is satisfiable exactly if $\mathbb{C}$ is satisfiable with respect to $\mathbb{K}$.

"$\Rightarrow$": If $\mathcal{L}$ is satisfiable, then there must be a valuation $v : \{p_1, \ldots p_k\} \to \{\mathit{true}, \mathit{false}\}$ under which $\mathcal{L}$ evaluates to *true*. Let L_v be the set of literals such that $p \in L_v$ whenever $v(p) = \mathit{true}$ and $\neg p \in L_v$ whenever $v(p) = \mathit{false}$. Next, we show that $(A_1, A_2, A_3) = (\{*\}, \{*\} \cup (L_v \cap M), \{*\} \cup (L_v \cap B))$ is a triconcept of $\mathbb{K}_\mathcal{L}$. First, $A_1 \times A_2 \times A_3 \subseteq Y$, since L_v denotes a valuation and it cannot both contain some p_i and its negation $\neg p_i$. We now show that (A_1, A_2, A_3) is also maximal, i.e., no component can be extended while maintaining $A_1 \times A_2 \times A_3 \subseteq Y$. Since L_v already contains for every $i \in \{1, \ldots, n\}$ either p_i or $\neg p_i$, extending A_2 or A_3 would lead to some i satisfying $p_i \in A_2$ and $\neg p_i \in A_3$ which contradicts $(*, p_i, \neg p_i) \notin Y$. It remains to show that A_1 cannot be extended. Let us assume that it can be extended, i.e., for some $L_j \in \mathcal{L}$ we have $L_j \in A_1$. It follows that $\{L_j\} \times A_2 \times A_3 \subseteq Y$ holds. However, by construction, we know that $L_j \cap L_v$ is non-empty. On one hand, assuming there is some $p_i \in L_j \cap L_v$, we conclude $p_i \in A_2$ and thus $(L_j, p_i, *) \in Y$ which contradicts the definition of the tricontext. On the other hand, assuming there is some $\neg p_i \in L_j \cap L_v$, we conclude $\neg p_i \in A_3$ and thus $(L_j, *, \neg p_i) \in Y$ which, again, contradicts the construction of the tricontext. Hence A_1 cannot be extended either and (A_1, A_2, A_3) is indeed a triconcept, which obviously also satisfies $\mathbb{C}_\mathcal{L}$.

"⇐": Assume $\mathbb{C}_{\mathcal{L}}$ is satisfiable with respect to $\mathbb{K}_{\mathcal{L}}$. Then there must be a triconcept of $\mathbb{K}_{\mathcal{L}}$ satisfying it, i.e. a triconcept of the form $(\{*\}, A_2, A_3)$. Since $(*, p_i, \neg p_i) \notin Y$, we know that for no p_i holds $p_i \in A_2$ and $\neg p_i \in A_3$ at the same time. On the other hand, by maximality, for every p_i one of $p_i \in A_2$ and $\neg p_i \in A_3$ must hold. Therefore, we can define a valuation v by letting $v(p_i) = \mathit{true}$ whenever $p_i \in A_2$ and letting $v(p_i) = \mathit{false}$ whenever $\neg p_i \in A_3$. We now show that v is a valuation mapping $\mathcal{L}$ to *true* and thus proving the satisfiability of $\mathcal{L}$. By assumption, $(\{*\}, A_2, A_3)$ is maximal, thus, by maximality of the first component, for every $L_j \in \mathcal{L}$ must hold that $\{L_j\} \times A_2 \times A_3 \not\subseteq Y$. Then, by construction of $\mathbb{K}_{\mathcal{L}}$ there must be either some $p_i \in L_j$ with $p_i \in A_2$ or there must be some $\neg p_i \in L_j$ with $\neg p_i \in A_3$. In any case, this means that L_j is mapped to *true* under v. Since the same argument applies to every $L_j \in \mathcal{L}$ we find that v is indeed a valuation proving the satisfiability of $\mathcal{L}$. □

In what follows, we use the same 3SAT problem (see Example 2.2.1) as in the previous chapter, to demonstrate the reduction of 3SAT to TMCSAT.

Example 3.4.2 *Consider $\mathcal{L} = \{L_1, L_2, L_3\}$ with $L_1 = \{r, s, \neg q\}$, $L_2 = \{s, \neg q, \neg r\}$, and $L_3 = \{\neg q, \neg r, \neg s\}$. The corresponding 3SAT problem amounts to checking if $\varphi_{\mathcal{L}} = (\neg q \vee r \vee s) \wedge (\neg q \vee \neg r \vee s) \wedge (\neg q \vee \neg r \vee \neg s)$ is satisfiable. However, this 3SAT problem can be reduced to the question if the membership constraint $(\{*\}, \{L_1, L_2, L_3\}, \emptyset, \emptyset, \emptyset, \emptyset)$ is satisfiable in the following tricontext:*

$*$	$*$	q	r	s
$*$	×	×	×	×
$\neg q$	×		×	×
$\neg r$	×	×		×
$\neg s$	×	×	×	

L_1	$*$	q	r	s
$*$	×	×		
$\neg q$		×	×	×
$\neg r$	×	×	×	×
$\neg s$	×	×	×	×

L_2	$*$	q	r	s
$*$	×	×	×	
$\neg q$		×	×	×
$\neg r$		×	×	×
$\neg s$	×	×	×	×

L_3	$*$	q	r	s
$*$	×	×	×	×
$\neg q$		×	×	×
$\neg r$		×	×	×
$\neg s$		×	×	×

This encoding ensures that for the union U of the intent and modus of the formal concepts the following holds: for every p_i, either $p_i \in U$ or $\neg p_i \in U$. Hence, the union U contains a valid valuation for the formula $\varphi_{\mathcal{L}}$. Moreover, there is a bijection between the valuations making the formula true and the triconcepts satisfying the membership constraint. The corresponding triconcept for the valuation $v = \{q \mapsto \mathit{true}, r \mapsto \mathit{false}, s \mapsto \mathit{true}\}$, making $\varphi_{\mathcal{L}}$ true, is $(\{*\}, \{*, q, s\}, \{*, \neg r\})$.

Corollary 3.4.4.1 *If we add other required or forbidden sets to any of the membership constraints described in Theorem 3.4.4, the satisfiability problem remains* NP*-complete.*

Similar to the dyadic case, when excluding the critical cases, i.e. membership constraints described in Theorem 3.4.4 and Corollary 3.4.4.1, the satisfiability problem becomes tractable and, for some cases, even trivially true.

Theorem 3.4.5 *When restricted to membership constraints of the form* $(\emptyset, G^-, M^+, \emptyset, B^+, \emptyset)$, $(G^+, \emptyset, \emptyset, M^-, B^+, \emptyset)$, *and* $(G^+, \emptyset, M^+, \emptyset, \emptyset, B^-)$, *TMCSAT is in* AC^0.

Proof. Let us consider membership constraints of the form $(\emptyset, G^-, M^+, \emptyset, B^+, \emptyset)$. We observe that the trivial triconcept (G_U, M, B) with $G_U = \{g \mid \{g\} \times M \times B \subseteq Y\}$ is maximal with respect to its intent and modus, and minimal with respect to its extent, i.e. for every triconcept (A_1, A_2, A_3) of $\mathbb{K}$ we have that $G_U \subseteq A_1$ and (trivially) $A_2 \subseteq M$ as well as $A_3 \subseteq B$. Therefore the membership constraint $\mathbb{C}$ is satisfiable with respect to $\mathbb{K}$ if and only if (G_U, M, B) satisfies it. For checking this, it suffices to check if $G_U \cap G^- = \emptyset$ which amounts to checking if for every $g \in G^-$ there are $m \in M$ and $b \in B$ with $(g, m, b) \notin Y$. This, in turn is equivalent to $\mathcal{I}_{\mathbb{K},\mathbb{C}}$ satisfying the first-order formula

$$\forall x.p_{G^-}(x) \rightarrow \exists y, z.(p_M(y) \wedge p_B(z) \wedge \neg p_Y(x, y, z)).$$

From the descriptive theory result stated in Remark 2.2.1, it follows that the considered restricted version of TMCSAT is in AC^0.

AC^0 membership for the other forms of membership constraints follows by symmetry. □

Corollary 3.4.5.1 *We can deduce that the satisfiability problems for membership constraints having one forbidden set and the corresponding required set empty are in* AC^0. *As subclasses of the membership constraint class described in Theorem 3.4.5, the following types of membership constraints are also in* AC^0*:*

- $(\emptyset, G^-, M^+, \emptyset, B^+, \emptyset)$, $(G^+, \emptyset, \emptyset, M^-, B^+, \emptyset)$, $(G^+, \emptyset, M^+, \emptyset, \emptyset, B^-)$,
- $(\emptyset, G^-, M^+, \emptyset, \emptyset, \emptyset)$, $(\emptyset, G^-, \emptyset, \emptyset, B^+, \emptyset)$, $(G^+, \emptyset, \emptyset, M^-, \emptyset, \emptyset)$, $(\emptyset, \emptyset, \emptyset, M^-, B^+, \emptyset)$, $(G^+, \emptyset, \emptyset, \emptyset, \emptyset, B^-)$, $(\emptyset, \emptyset, M^+, \emptyset, \emptyset, B^-)$,
- $(\emptyset, G^-, \emptyset, \emptyset, \emptyset, \emptyset)$, $(\emptyset, \emptyset, \emptyset, M^-, \emptyset, \emptyset)$, $(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, B^-)$,
- $(\emptyset, \emptyset, M^+, \emptyset, B^+, \emptyset)$, $(G^+, \emptyset, \emptyset, \emptyset, B^+, \emptyset)$, $(G^+, \emptyset, M^+, \emptyset, \emptyset, \emptyset)$,
- $(G^+, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$, $(\emptyset, \emptyset, M^+, \emptyset, \emptyset, \emptyset)$, $(\emptyset, \emptyset, \emptyset, \emptyset, B^+, \emptyset)$.

As in the dyadic case, there are membership constraints which are trivially true, since they already suggest the formal concept that satisfies the constraint. While for membership constraints on dyadic contexts this was the case when they had one required set, for triadic contexts this is the case when membership constraints contain at most two required sets.

Theorem 3.4.6 *When restricting to membership constraints that have all forbidden sets empty and at least one of the required sets empty, TMCSAT is trivially true. These membership constraints have the following form:*

- $(\emptyset, \emptyset, M^+, \emptyset, B^+, \emptyset)$, $(G^+, \emptyset, \emptyset, \emptyset, B^+, \emptyset)$, $(G^+, \emptyset, M^+, \emptyset, \emptyset, \emptyset)$, *or*
- $(G^+, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$, $(\emptyset, \emptyset, M^+, \emptyset, \emptyset, \emptyset)$, $(\emptyset, \emptyset, \emptyset, \emptyset, B^+, \emptyset)$.

Proof. Let us consider the first membership constraint $(\emptyset, \emptyset, M^+, \emptyset, B^+, \emptyset)$, while the other cases follow by symmetry. We observe that the trivial concept (G_U, M, B) with $G_U = \{g \mid \{g\} \times M \times B \subseteq Y\}$ satisfies any constraint of this form, thus satisfiability is always ensured. □

3.4.2.3 Membership Constraints in n-adic Formal Concept Analysis

Classical FCA and triadic FCA, the two cases considered in the last sections, can be seen as two instances of a general framework that we call n-adic FCA. The n-adic case of formal concept analysis, also called polyadic FCA, was introduced in Section 2.1.4. In this section, we consider the general case and observe that the already identified causes of intractability are the only ones also when increasing the arity of the incidence relation further [**Rudolph *et al.*, 2015a**].

Definition 3.4.3 *Let $\mathbb{K} = (K_1, \ldots, K_n, R)$ be an n-adic context. We define an* **n-adic membership constraint** *$\mathbb{C}$ on the context $\mathbb{K}$ as a $2n$-tuple $\mathbb{C} = (K_1^+, K_1^-, \ldots, K_n^+, K_n^-)$ with $K_i^+ \subseteq K_i$ called* **required sets** *and $K_i^- \subseteq K_i$ called* **forbidden sets**.

An n-concept $(A_1, \ldots, A_n)$ of $\mathbb{K}$ is said to **satisfy** *such a membership constraint if the following conditions hold:*

- $K_i^+ \subseteq A_i$ *for all* $i \in \{1, \ldots, n\}$ *and*
- $K_i^- \cap A_i = \emptyset$ *for all* $i \in \{1, \ldots, n\}$.

Furthermore, we let $\mathrm{Mod}(\mathbb{K}, \mathbb{C})$, *respectively* $\mathrm{Mod}_p(\mathbb{K}, \mathbb{C})$, *denote the set of all n-concepts, respectively proper n-concepts, of $\mathbb{K}$ that satisfy $\mathbb{C}$.*

An n-adic membership constraint is said to be **satisfiable** *with respect to $\mathbb{K}$, if it is satisfied by one of its n-concepts, that is if* $\mathrm{Mod}(\mathbb{K}, \mathbb{C}) \neq \emptyset$.

Similarly, an n-adic membership constraint is said to be **properly satisfiable** *with respect to $\mathbb{K}$, if it is satisfied by one of its proper n-concepts, that is if* $\mathrm{Mod}_p(\mathbb{K}, \mathbb{C}) \neq \emptyset$.

Let nMCSAT denote the decision problem of membership constraint satisfiability for an n-adic context $\mathbb{K}$:

problem nMCSAT
input: n-context $\mathbb{K} = (K_1, \ldots, K_n, R)$,
n-adic membership constraint $\mathbb{C} = (K_1^+, K_1^-, \ldots, K_n^+, K_n^-)$
output: YES if $\mathbb{C}$ satisfiable with respect to $\mathbb{K}$, NO otherwise.

It turns out that the triadic case exhibits all necessary information needed to settle the general case, taking into account some straightforward adaptations, hence the following theorem is immediate.

Theorem 3.4.7 *For a fixed $n > 2$, the nMCSAT problem is*

- NP-*complete for any class of constraints that allows for*
  - *the arbitrary choice of at least two forbidden sets or*
  - *the arbitrary choice of at least one forbidden set and the corresponding required set,*
- *in* AC^0 *for the class of constraints with at most one forbidden set and the corresponding required set empty,*
- *trivially true for the class of constraints with all forbidden sets and at least one required set empty.*

Proof. Let us first observe that NP-membership for the general case is straightforward: we guess an n-uple $({K_1}^+, \ldots, {K_n}^+)$ from $2^{K_1} \times \ldots \times 2^{K_n}$ and check (in polynomial time) if it is a concept of $\mathbb{K}$ and if it satisfies the constraint.

For the first two types we can study membership constraints of the form $(\emptyset, {K_1}^-, \emptyset, {K_2}^-, \emptyset, \ldots, \emptyset,)$ and $({K_1}^+, {K_1}^-, \emptyset, \ldots, , \emptyset)$. If these categories of membership constraints are NP-hard, then membership constraints with more forbidden or required sets remain NP-hard.

First, we study NP-completeness for membership constraints of the form $(\emptyset, {K_1}^-, \emptyset, {K_2}^-, \emptyset, \ldots, \emptyset,)$. We show NP-hardness by proving that the corresponding NP-complete TMCSAT problem can be reduced to a nMCSAT problem of this type.

Given some triadic context $\mathbb{K} = (G, M, B, Y)$, we define the n-adic correspondent $\mathbb{K}_n = (G, M, B, \{*\}, \ldots, \{*\}, Y \times \{*\} \ldots \times \{*\})$, where $\{*\}$ appears $n-3$ times. Herefrom it follows that the set of all n-concepts of $\mathbb{K}_n$ is the union of $\{(A_1, A_2, A_3, \{*\}, \ldots, \{*\}) \mid (A_1, A_2, A_3)$ triconcept of $\mathbb{K}\}$ and all the trivial n-concepts having all components maximal

and one of the sets at the position $4 \leq i \leq n$ empty. We deduce that every TMCSAT problem of the form $(\emptyset, G^-, \emptyset, M^-, \emptyset, \emptyset)$ can be reduced to the nMCSAT problem of the form $(\emptyset, {K_1}^-, \emptyset, {K_2}^-, \emptyset, \ldots, \emptyset,)$, defined on the context $\mathbb{K}_n$. Since the former problem is NP-complete due to Theorem 3.4.4, the latter must be NP-hard. All the other cases of membership constraints with at least two forbidden sets follow by symmetry.

Moreover, using the same argument, we can show that the NP-complete TMCSAT problem $(G^+, G^-, \emptyset, \emptyset, \emptyset, \emptyset)$ can be reduced to a nMCSAT problem of the type $({K_1}^+, {K_1}^-, \emptyset, \ldots, , \emptyset)$. It follows from Theorem 3.4.4 that the nMCSAT problem is NP-complete when allowing for the arbitrary choice of at least one fortbidden set and the corresponding required set.

For the class of constraints with at most one forbidden set and the corresponding required set empty we consider constraints of the following form:

$$\mathbb{C} = (\emptyset, {K_1}^-, {K_2}^+, \emptyset, {K_3}^+, \emptyset, \ldots, {K_n}^+, \emptyset).$$

The other cases follow by symmetry. We observe that the trivial concept $(K_{min}, K_2, K_3, \ldots, K_n)$ with $K_{min} = \{g \mid \{g\} \times K_2 \times \ldots \times K_n \subseteq R\}$ is minimal with respect to its extent and maximal with respect to all the other components. Therefore, the membership constraint $\mathbb{C}$ is satisfiable with respect to $\mathbb{K}$ if and only if it is satisfied by $(K_{min}, K_2, K_3, \ldots, K_n)$. For checking this, it suffices to check if $K_{min} \cap {K_1}^- = \emptyset$. As we have seen in previous proofs, this condition is equivalent to $\mathcal{I}_{\mathbb{K},\mathbb{C}}$ satisfying the first-order formula

$$\forall x. p_{{K_1}^-}(x) \rightarrow \exists y_2, \ldots, y_n. (p_{K_2}(y_2) \wedge \ldots \wedge p_{K_n}(y_n) \wedge \neg p_R(x, y_2, \ldots, y_n)).$$

From the descriptive theory result stated in Remark 2.2.1, it follows that the considered type of nMCSAT is in AC^0.

Finally, for the class of constraints with all forbidden sets and at least one required set empty, we observe that the trivial concepts (for example $(K_{min}, K_2, K_3, \ldots, K_n)$ with $K_{min} = \{g \mid \{g\} \times K_2 \times \ldots \times K_n \subseteq R\}$ when the first required set is empty) always satisfy the constraints, thus satisfiability is ensured. □

3.4.2.4 A Discussion on Proper Satisfiability

We introduce the following decision problems for proper satisfiability.

problem MCSAT_p
input: dyadic formal context $\mathbb{K} = (G, M, I)$,
membership constraint $\mathbb{C} = (G^+, G^-, M^+, M^-)$
output: YES if $\mathbb{C}$ properly satisfiable with respect to $\mathbb{K}$, NO otherwise.

problem TMCSAT$_p$
input: tricontext $\mathbb{K} = (G, M, B, Y)$,
triadic membership constraint $\mathbb{C} = (G^+, G^-, M^+, M^-, B^+, B^-)$
output: YES if $\mathbb{C}$ properly satisfiable with respect to $\mathbb{K}$, NO otherwise.

problem nMCSAT$_p$
input: n-context $\mathbb{K} = (K_1, \ldots, K_n, R)$,
n-adic membership constraint $\mathbb{C} = (K_1^+, K_1^-, \ldots, K_n^+, K_n^-)$
output: YES if $\mathbb{C}$ properly satisfiable with respect to $\mathbb{K}$, NO otherwise.

We observe that some of the results regarding satisfiability of memberships presented so far, easily carry over to proper satisfiability.

Remark 3.4.1 *The proper satisfiability problems* MCSAT$_p$*,* TMCSAT$_p$ *and* nMCSAT$_p$ *having corresponding satisfiability problems which are* NP*-complete remain intractable for proper satisfiaibility as well.*

However, for some satisfiability problems which are trivial or in AC^0, it is not straight forward to show that the corresponding proper satisfiability problems remain tractable. In the following we discuss the tractable cases (from simple satisfiability) for the MCSAT$_p$ problem. First, we observe that the trivial case of MCSAT satisfiability remains tractable, however it is not trivial any more for proper satisfiability.

Example 3.4.3 *Consider the following dyadic context:*

	m1	*m2*
g1		
g2	×	

Now we analyze the satisfiability and proper satisfiability problems for the following membership constraint: $(\{g_1\}, \emptyset, \emptyset, \emptyset)$*. The formal concept generated by object* g_1 *is* $({g_1}'', {g_1}') = (\{g_1, g_2\}, \emptyset)$*. We observe that, as stated in Thereom 3.4.3, the satifiability problem is trivially true for concept* $({g_1}'', {g_1}')$*, while the proper satisfiability problem is no longer true. Furthermore, in order to be able to state whether this concept properly satisfies the problem or not, we need to compute it and check if* ${g_1}' \neq \emptyset$*, so, in general, the proper satisfiability problem of this type is in* AC^0*.*

Lemma 3.4.8 *When restricted to membership constraints of the form* $(G^+, \emptyset, \emptyset, \emptyset)$ *or* $(\emptyset, \emptyset, M^+, \emptyset)$*,* MCSAT$_p$ *is in* AC^0*.*

Proof. Similarly to the proof of the trivial MCSAT case, we observe that the membership constraints are properly satisfiable if and only if they are properly satisfied by the formal concepts $((G^+)'', (G^+)')$, respectively $((M^+)', (M^+)'')$. However, as opposed to the satisfiability case, here, it is not trivial since we still need to check that these are proper concepts. Hence, the conditions that need to be checked are that $(G^+)' \neq \emptyset$, respectively $(M^+)' \neq \emptyset$. The first condition can be expressed by the following first-order formula (while the second condition by the dual formula):

$$\exists y.(p_M(y) \wedge (\forall x.p_{G^+}(x) \rightarrow p_I(x, y))).$$

Using the result from descriptive complexity theory, it follows that the described types of MCSAT_p are in AC^0. □

Next we show that for all the types of membership constraints for which the (simple) satisfiability problem is in AC^0, the corresponding proper satisfiability problem remains in AC^0.

Theorem 3.4.9 *When restricted to membership constraints of the form* $(G^+, \emptyset, M^+, M^-)$ *or* $(G^+, G^-, M^+, \emptyset)$, MCSAT_p *is in* AC^0.

Proof. We prove the statement for constraints of the form $\mathbb{C} = (G^+, \emptyset, M^+, M^-)$, the other case follows by duality. For this purpose, we distinguish the following two cases: $M^+ \neq \emptyset$ and $M^+ = \emptyset$.

"$M^+ \neq \emptyset$": In this case the idea is similar to the one used for proving (simple) satisfiability. We observe that $(G^+, \emptyset, M^+, M^-)$ is properly satisfiable with respect to $\mathbb{K}$ if and only if it is properly satisfied by the concept $((M^+)', (M^+)'')$. The conditions that need to be checked are:

- (1) $G^+ \subseteq (M^+)'$,
- (2) $(M^+)'' \cap M^- = \emptyset$ and
- (3) $(M^+)' \neq \emptyset$ (which can be left out if $G^+ \neq \emptyset$).

All three conditions can be expressed by first-order formulae as we have seen in the proofs of Theorem 3.4.2 and Remark 3.4.8.

"$M^+ = \emptyset$": In this case we have to find a different method for constructing the concept satisfying the membership constraints. For that reason we need to check that $\exists m \in M$ with the following properties:

- (1) $G^+ \subseteq m'$,

- (2) $m'' \cap M^- = \emptyset$ and
- (3) $m' \neq \emptyset$ (which can be left out if $G^+ \neq \emptyset$).

These conditions, being analog to the ones from the first case, can also be expressed by first-order-formulae.

We can conclude (using the corresponding result from descriptive complexity theory described in Remark 2.2.1) that for the described class of membership constraints MCSAT_p is in AC^0. □

Similarly as for (simple) satisfiability, it follows that the subclasses of the previous membership constraints class are in AC^0 for proper satisfiability as well.

Corollary 3.4.9.1 *We can deduce that membership constraints of the form* $(G^+, G^-, \emptyset, \emptyset)$, $(\emptyset, \emptyset, M^+, M^-)$, $(G^+, \emptyset, M^+, \emptyset)$, $(G^+, \emptyset, \emptyset, M^-)$, $(\emptyset, G^-, M^+, \emptyset)$, $(\emptyset, G^-, \emptyset, \emptyset)$, $(\emptyset, \emptyset, \emptyset, M^-)$, $(G^+, \emptyset, \emptyset, \emptyset)$, $(\emptyset, \emptyset, M^+, \emptyset)$ *are also in* AC^0 *when considering the problem of proper satisfiability.*

Proof. The types of membership constraints described here fall into one or both of the cases (or their dual correspondent) described in the previous proof an hence similar conditions need to be checked. It follows from the first-order expressibility that all the subclasses are in AC^0. □

Observation 3.4.1 *When considering triadic and n-adic membership constraints, it is not clear whether the results from (simple) satisfiability carry over to proper satisfiability for the tractable cases. We observe that for tractable TMCSAT or nMCSAT problems we have used trivial concepts in order to show tractability. These proofs do not hold for proper satisfiability, since the trivial concepts might not be proper. Furthermore, we conjecture that some of these problems become intractable for proper satisfiability. In conclusion, the computational complexities of these proper satisfiability problems need to be further analyzed and remain an open question.*

3.4.3 Encoding for Membership Constraints in Answer Set Programming

Given that satisfiability of membership constraints can in general be NP-complete, it is nontrivial to find efficient algorithms. However, the problem can be nicely expressed with answer set programming. We present in what follows the encoding for the n-adic case [**Rudolph *et al.*, 2015a**]. We assume that the specific problem is given by the following set of ground facts $F_{\mathbb{K},\mathbb{C}}$:

- $\texttt{set}_i(a)$ for all $a \in K_i$,
- $\texttt{rel}(a_1, \ldots, a_n)$ for all $(a_1, \ldots, a_n) \in R$,
- $\texttt{required}_i(a)$ for all $a \in K_i^+$, and
- $\texttt{forbidden}_i(a)$ for all $a \in K_i^-$.

Let P denote the following fixed answer set program (with rules for every $i \in \{1, \ldots, n\}$):

$$
\begin{aligned}
\texttt{in}_i(x) &\leftarrow \texttt{set}_i(x) \wedge {\sim}\texttt{out}_i(x) \\
\texttt{out}_i(x) &\leftarrow \texttt{set}_i(x) \wedge {\sim}\texttt{in}_i(x) \\
&\leftarrow \textstyle\bigwedge_{j \in \{1,\ldots,n\}} \texttt{in}_j(x_j) \wedge {\sim}\texttt{rel}(x_1, \ldots, x_n) \\
\texttt{exc}_i(x_i) &\leftarrow \textstyle\bigwedge_{j \in \{1,\ldots,n\} \setminus \{i\}} \texttt{in}_j(x_j) \wedge {\sim}\texttt{rel}(x_1, \ldots, x_n) \\
&\leftarrow \texttt{out}_i(x) \wedge {\sim}\texttt{exc}_i(x) \\
&\leftarrow \texttt{out}_i(x) \wedge \texttt{required}_i(x) \\
&\leftarrow \texttt{in}_i(x) \wedge \texttt{forbidden}_i(x)
\end{aligned}
$$

$$
|\{x \mid \texttt{in}_i(x)\}| \geq 1, \text{ for all } i \in \{1, \ldots, n\}
$$

Intuitively, the first two lines "guess" an n-concept candidate by starting with an empty constraint and stipulating for each element of each K_i if they are in or out, hence reaching a membership constraint of the form $\mathbb{C} = (K_1^+, K_1^-, \ldots, K_n^+, K_n^-)$ with $K_i^+ \cup K_i^- = K_i$ for every i. The third rule eliminates a candidate if it violates the condition $K_1^+ \times K_2^+ \times \ldots \times K_n^+ \subseteq Y$, while the fourth and fifth rule ensure the maximality condition for n-concepts. The sixth and the seventh rule eliminate n-concepts violating the given membership constraint by checking if the required and forbidden elements are assigned correspondingly in the obtained membership constraint, i.e. required elements belong to K_i^+, forbidden elements belong to K_j^-, for some $i, j \in \{1, \ldots, n\}$. The last rule eliminates non-proper concepts, hence it is optional and must be used only when dealing with proper satisfiability.

An n-concept can be read from an answer set X as follows: $(\{a \mid \texttt{in}_1(a) \in X\}, \ldots, \{a \mid \texttt{in}_n(a) \in X\})$. There is a one-to-one correspondence between the answer sets X of $F_{\mathbb{K},\mathbb{C}} \cup P$ and the n-concepts of $\mathbb{K}$ satisfying $\mathbb{C}$, obtained as described previously. Consequently, optimized ASP tools can be used for checking satisfiability but also for enumerating all satisfying n-concepts.

3.4.4 Navigation in Conceptual Spaces based on Membership Constraints

In this section we describe an interactive search scenario where membership constraints can be put to use to support a user in finding a proper n-concept with desired properties [**Rudolph *et al.*, 2015a; Rudolph *et al.*, 2016**]. This is particularly useful in cases where the number of n-concepts is very large. In what follows, we explain in more detail the intuition behind the approach and present the formal algorithms, as well as their practical optimized versions for implementation.

First, let us remind that, given an n-context $\mathbb{K} = (K_1, \ldots, K_n, R)$ and a corresponding membership constraint $\mathbb{C}$, $\mathrm{Mod}_p(\mathbb{K}, \mathbb{C})$ denotes the set of all proper n-concepts of $\mathbb{K}$ that satisfy $\mathbb{C}$. We observe that for the "zero-constraint" $\mathbb{C}_\emptyset = (\emptyset, \ldots, \emptyset)$, the set $\mathrm{Mod}_p(\mathbb{K}, \mathbb{C}_\emptyset)$ contains all proper n-concepts of $\mathbb{K}$. Next, for two membership constraints $\mathbb{C}_1$ and $\mathbb{C}_2$, let $\mathbb{C}_1 \preceq \mathbb{C}_2$ denote componentwise $\subseteq$, read as $\mathbb{C}_1$ is "more general than" $\mathbb{C}_2$, or $\mathbb{C}_2$ is "more specific than" $\mathbb{C}_1$. We observe that, from $\mathbb{C}_1 \preceq \mathbb{C}_2$ it follows that $\mathrm{Mod}_p(\mathbb{K}, \mathbb{C}_2) \subseteq \mathrm{Mod}_p(\mathbb{K}, \mathbb{C}_1)$. Finally, every proper n-concept $\mathcal{C} = (A_1, \ldots, A_n)$ of $\mathbb{K}$ gives rise to the characteristic membership constraint $\mathbb{C}_\mathcal{C} = (A_1, K_1 \setminus A_1, \ldots, A_n, K_n \setminus A_n)$ with $\mathrm{Mod}_p(\mathbb{K}, \mathbb{C}_\mathcal{C}) = \{\mathcal{C}\}$.

We now want to describe the identification of a proper n-concept by a user as an iterated approximation process starting from $\mathbb{C}_\emptyset$ and going along a chain of ever more specific (but satisfiable) membership constraints until $\mathbb{C}_\mathcal{C}$ is reached for some n-concept $\mathcal{C}$. Non-proper concepts are considered out of scope for knowledge exploration, thus we exclude them from our consideration. The described navigation would, however, also work if these concepts were taken into account.

Given a current satisfiable constraint $\mathbb{C} = (K_1^+, K_1^-, \ldots, K_n^+, K_n^-)$, the next constraint is determined by the user by picking some $a \in K_i \setminus (K_i^+ \cup K_i^-)$ for some i and adding it either to K_i^+ or K_i^-. Intuitively, for some element, whose set membership in the looked-for n-concept is not yet determined, the user has to decide to include or exclude it. In order to avoid that the membership constraint turns unsatisfiable as a consequence of the user's refinement decision, we will perform constraint propagation on $\mathbb{C}$ before the interaction: for every $a \in K_i \setminus (K_i^+ \cup K_i^-)$, for some i, if adding a to K_i^+, respectively K_i^-, would result in an unsatisfiable constraint, we add it to K_i^-, respectively K_i^+. Note that not both can be the case at the same time, since otherwise $\mathbb{C}$ itself would be unsatisfiable. Furthermore, we deduce that an initial propagation phase is also necessary, i.e. a propogation of the "zero-constraint" $\mathbb{C}_\emptyset = (\emptyset, \ldots, \emptyset)$, in order to ensure that the first step of the user does not lead to an unsatisfiable constraint.

The method is formally specified in Algorithm 1, which calls Algorithm 2. Algorithm 2

is based on an nMCSAT solving procedure called NMCSAT, which relies on the ASP encoding described in the previous section, including the constraint regarding the cardinality of the concept's components in order to ensure proper satisfiability.

Algorithm 1 interactive n-concept finding algorithm

function FINDNCONCEPTINTERACTIVE($\mathbb{K}$)
Input: n-context $\mathbb{K} = (K_1, \ldots, K_n, R)$
Output: n-concept searched by user
Data: membership constraint $\mathbb{C} = (K_1^+, K_1^-, \ldots, K_n^+, K_n^-)$

$\mathbb{C} = (\emptyset, \ldots, \emptyset)$
$\mathbb{C} =$PROPAGATE$(\mathbb{K}, \mathbb{C})$
while $K_i \neq K_i^+ \cup K_i^-$ for some $i \in \{1, \ldots, n\}$ **do**
have user pick one such i and $a \in K_i \setminus (K_i^+ \cup K_i^-)$
have user pick some *decision* $\in \{in, out\}$
if *decision* $= in$ **then**
update $\mathbb{C}$ by $K_i^+ = K_i^+ \cup \{a\}$
else
update $\mathbb{C}$ by $K_i^- = K_i^- \cup \{a\}$
end if
$\mathbb{C} =$PROPAGATE$(\mathbb{K}, \mathbb{C})$
end while
return $(K_1^+, \ldots, K_n^+)$
end function

Algorithm 2 propagation of user decisions

function PROPAGATE($\mathbb{K}, \mathbb{C}$)
Input: n-context $\mathbb{K} = (K_1, \ldots, K_n, R)$,
membership constraint $\mathbb{C} = (K_1^+, K_1^-, \ldots, K_n^+, K_n^-)$
Output: updated membership constraint $\mathbb{C}$
Data: membership constraint $\mathbb{C}'$

```
for all i ∈ {1, ..., n} do
    for all a ∈ K_i \ (K_i^+ ∪ K_i^-) do
        obtain ℂ' from ℂ by adding a to K_i^+
        if NMCSAT(𝕂, ℂ') = NO then
            update ℂ by adding a to K_i^-
        end if
        obtain ℂ' from ℂ by adding a to K_i^-
        if NMCSAT(𝕂, ℂ') = NO then
            update ℂ by adding a to K_i^+
        end if
    end for
end for
return ℂ
```

end function

Algorithm 3 propagation of user decisions optimized

function PROPAGATEOPTIMIZED($\mathbb{K}, \mathbb{C}$)
Input: n-context $\mathbb{K} = (K_1, \ldots, K_n, R)$,
membership constraint $\mathbb{C} = (K_1^+, K_1^-, \ldots, K_n^+, K_n^-)$
Output: updated membership constraint $\mathbb{C}$
Data: $L_1^+, L_1^-, \ldots, L_n^+, L_n^-$

for all $i \in \{1, \ldots, n\}$ **do**

$$L_i^+ = \bigcap_{(A_1,\ldots,A_n)\in \mathrm{Mod}_p(\mathbb{K},\mathbb{C})} A_i$$

$$L_i^- = \bigcap_{(A_1,\ldots,A_n)\in \mathrm{Mod}_p(\mathbb{K},\mathbb{C})} K_i \setminus A_i.$$

end for
$\mathbb{C} = (L_1^+, L_1^-, \ldots, L_n^+, L_n^-)$
return $\mathbb{C}$
end function

Intuitively, Algorithm 2 determines for each element $a \in K_i$ in an undetermined state, i.e. $a \notin {K_i}^+ \cup {K_i}^-$, if adding this element either to ${K_i}^+$ or to ${K_i}^-$ would result in an unsatisfiable constraint. If this is the case, we propagate the opposite constraint, so that the problem remains satisfiable regardless of the user's next step.

Note that the interactive algorithm sketched here does not need to compute all (possibly exponentially many) n-concepts upfront, however, it relies on polynomially many subsequent nMCSAT checks. For that reason, implementing it as described in Algorithm 2, would result in a slow and non-efficient algorithm, since we need to call the NMCSAT procedure multiple times in the propagation phase. Hence, we try to optimize the propagation algorithm. First observe that the purpose of the propagation phase is to avoid arriving at an empty "subspace" corresponding to a membership constraint $\mathbb{C}$, meaning that $\mathbb{C}$ is not satisfied by any proper n-concept, i.e. $\mathrm{Mod}_p(\mathbb{K}, \mathbb{C}) = \emptyset$. Since the proper n-concepts in the "subspace" associated with a constraint $\mathbb{C}$ are exactly the n-concepts from $\mathrm{Mod}_p(\mathbb{K}, \mathbb{C})$, we obtain the optimized version of Algorithm 2 as described in Algorithm 3. It is then clear that after such an update, for every element e of some K_i which is still undetermined by $\mathbb{C}'$, there exist proper n-concepts $(E_1, \ldots, E_n)$ and $(F_1, \ldots, F_n)$ in $\mathrm{Mod}_p(\mathbb{K}, \mathbb{C}')$ with $e \in E_i$ but $e \notin F_i$. Consequently, whatever undetermined element the user chooses to include or exclude in the next step, the resulting membership constraint will be properly satisfiable. If the updated constraint $\mathbb{C}' = (L_1^+, L_1^-, \ldots, L_n^+, L_n^-)$ determines for every element if it is included or excluded (i.e., if $L_i^+ \cup L_i^- = K_i$ holds for every i), the user's navigation has narrowed down the space to the one proper n-concept $(L_1^+, \ldots, L_n^+)$.

Example 3.4.4 *We consider a triadic context (K_1, K_2, K_3, Y) where the object set K_1 consists of authors of scientific papers, the attribute set K_2 contains conference names/journal names while the conditions K_3 are the publication years. This context was formed using a small selection of data from the dblp[3] database. For this small selection we obtain the $2 \times 4 \times 2$ triadic context from Figure 3.12.*

There are exactly six triconcepts of this context, i.e. maximal 3D cuboids full of incidences:

- $(\{Rumpe, Alouni\}, \{Corr\}, \{2014, 2015\})$,
- $(\{Alouni\}, \{Corr, ICC, PIMRC\}, \{2014\})$,
- $(\{Alouni\}, \{Corr, ICC\}, \{2014, 2015\})$,
- $(\{Rumpe\}, \{Corr, HICSS\}, \{2015\})$,

[3] `http://dblp.l3s.de/dblp++.php`

- $(\emptyset, \{Corr, ICC, PIMRC, HICSS\}, \{2014, 2015\})$ *and*
- $(\{Rumpe, Alouni\}, \{Corr, ICC, PIMRC, HICSS\}, \emptyset)$.

The first four of these triconcepts are proper.

Let us assume that we start the navigation from this context. In the initial propagation phase, i.e. propagation of the "zero-constraint" $\mathbb{C}_\emptyset$*, we need to compute:*

$$L_i^+ = \bigcap_{(A_1,\ldots,A_n)\in Mod_p(\mathbb{K},\mathbb{C}_\emptyset)} A_i$$

and

$$L_i^- = \bigcap_{(A_1,\ldots,A_n)\in Mod_p(\mathbb{K},\mathbb{C}_\emptyset)} K_i \setminus A_i.$$

Herefrom, we obtain the first updated constraint

$$\mathbb{C} = (\emptyset, \emptyset, \{Corr\}, \emptyset, \emptyset, \emptyset),$$

meaning that there is no proper concept that does not include the attribute Corr. Let us assume that in the first step, the user specifies the exclusion of the attribute ICC from the intent, i.e.,

$$\mathbb{C}' = (\emptyset, \emptyset, \{Corr\}, \{ICC\}, \emptyset, \emptyset).$$

The proper 3*-concepts of* $\mathbb{K}$ *satisfying* $\mathbb{C}'$ *are*

$$C_1 = (\{Rumpe, Alouni\}, \{Corr\}, \{2014, 2015\}) \text{ and}$$
$$C_2 = (\{Rumpe\}, \{Corr, HICSS\}, \{2015\}),$$

therefore, we would obtain the updated constraint

$$\mathbb{C}'' = (\{Rumpe\}, \emptyset, \{Corr\}, \{ICC, PIMRC\}, \{2015\}, \emptyset).$$

If the user now decides to additionally exclude 2014 *from the modus, leading to the constraint*

$$\mathbb{C}''' = (\{Rumpe\}, \emptyset, \{Corr\}, \{ICC, PIMRC\}, \{2015\}, \{2014\}),$$

the only proper 3*-concept satisfying this constraint is* C_2*. Consequently,* $\mathbb{C}'''$ *will be updated to*

$$\mathbb{C}'''' = (\{Rumpe\}, \{Alouni\}, \{Corr, HICSS\}, \{ICC, PIMRC\}, \{2015\}, \{2014\}),$$

which then represents the final state of the navigation.

3.4.5 Implementation of Exploration and Navigation Tool based on Membership Constraints

Following the general scheme described in the previous section, we implement a navigation and exploration tool using different strategies. The first, straightforward implementation method is based on Answer Set Programming and the membership constraint encoding presented in Section 3.4.3 [**Rudolph *et al.*, 2015a**]. However, for evaluation purposes, we analyze alternative implementation methods and present another strategy that uses an exhaustive search in the formal concept set, which must be precomputed with a different FCA tool [**Rudolph *et al.*, 2016**].

The propagation described in Algorithm 2 tests for all elements, that are still in an undetermined state, which of the possible decisions on that element (included or excluded) give rise to a satisfiable membership constraint problem. In case one of the decisions generates an unsatisfiable problem, the complementary choice is automatically made. Remember that, as discussed in the previous sections, when starting from a satisfiable setting, it cannot be the case that both choices generate an unsatisfiable program.

The alternative to explicitly testing all the possible choices for every element in an undetermined state, as described in Algorithm 3, is to compute all the membership constraints corresponding to a formal concept, that additionaly satisfy the already added constraints, and to obtain their intersection. This intersection contains the *included* and *excluded* choices that need to be propagated, since their complementary constraints are not contained in any membership constraint that satisfies the problem and hence, would generate an unsatisfiable program. For the ASP approach we implement both propagation algorithms and run experiments in order to compare them and prove that Algorithm 3 is the more efficient propagation algorithm. However, for the brute-force approach we directly use the optimized version of the propagation.

The framework of the tools follows Algorithm 1. Moreover, for all implementations (ASP approach with simple propagation, ASP approach with optimized propagation, brute-force approach with optimized propagation) the user interface is the same: the first column includes possible actions and information about the state of the navigation process (*intermediate* or *final*), while the next columns each correspond to one dimension $i \in \{1, \ldots, n\}$ of the context. These columns consist of labeled lists containing the elements of K_i with elements from K_i^+ colored green and labeled with "in", elements from K_i^- colored red and labeled with "out", and elements from $K_i \setminus (K_i^+ \cup K_i^-)$ having no particular label or color. By clicking on one of the "unknown" elements, the user may switch it to "in" or "out". Subsequent constraint propagation as described in Algorithm 2 or Algorithm 3 then will possibly turn other "unknown" labels into "in" or "out" as a

ramification of the user's decision. When no more "unknown" labels are left, the target concept has been identified and the state of the navigation becomes "DONE". Until the final state is reached, the navigation is considered to be in an "INTERMIDIATE" state.

Let us consider Example 3.4.4 again, where we navigate in a $2 \times 4 \times 2$ context containg data from the dblp database about authors, journals/conferences and the years of publication. Figure 3.13 depicts a screenshot of the navigation example described in Example 3.4.4, namely the step corresponding to the post propagation constraint $\mathbb{C}'' = (\{Rumpe\}, \emptyset, \{Corr\}, \{ICC, PIMRC\}, \{2015\}, \emptyset)$. Here, we can see that required elements, i.e. object *Rumpe*, attribute *Corr* and condition *2015*, are marked with green, forbidden elements, i.e. attributes *ICC* and *PIMRC*, with red, while elements in an undetermined state are unmarked. Furthermore, required and forbidden elements have the *in*, respectively the *out* column checked. In this step, the navigation is considered to be in an intermediate state, since we did not reach a single formal concept and there still are elements in an unknown state. Figure 3.14 shows the final state of the navigation, that corresponds to the membership constraint:

$$\mathbb{C}''' = (\{Rumpe\}, \{Alouni\}, \{Corr, HICSS\}, \{ICC, PIMRC\}, \{2015\}, \{2014\}).$$

Although the graphical interface and the general framework deduced from Algorithm 1 are the same in both approaches, the implementation of the propagation algorithm (either the simple one or the optimized one) as well as the method for computing the formal concepts, however, differ for each of the chosen strategies. In what follows we describe the implementations of the ASP and the brute-force approach in more detail.

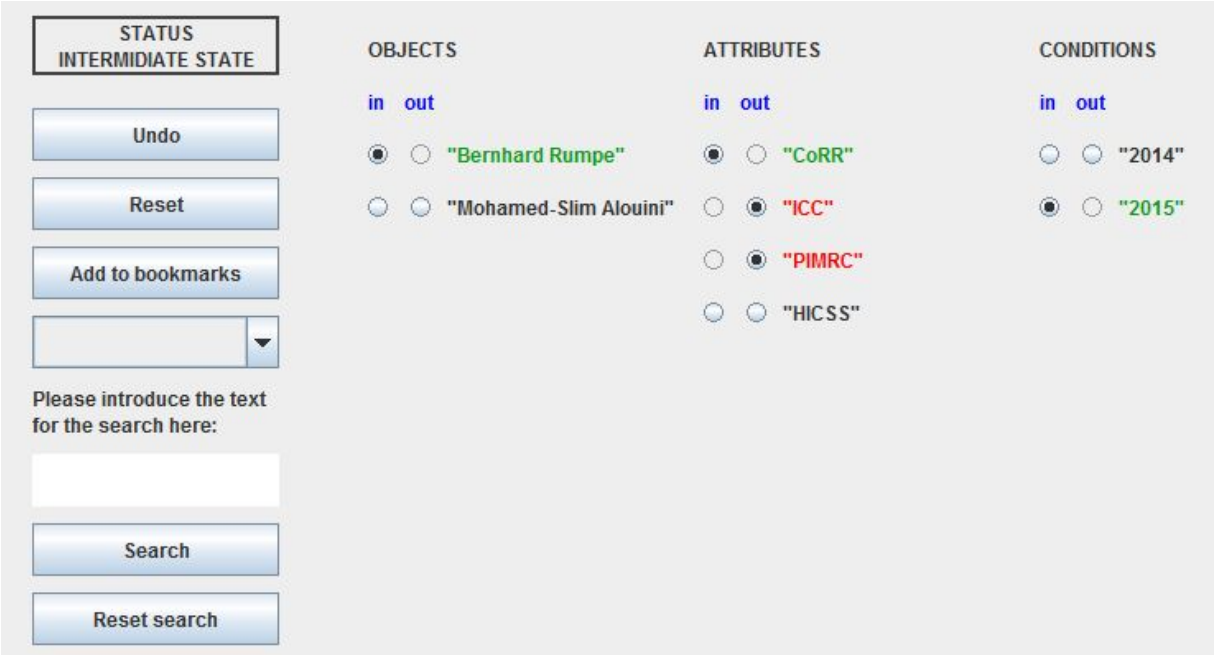


Figure 3.13: Screenshot navigation tool: intermediate state

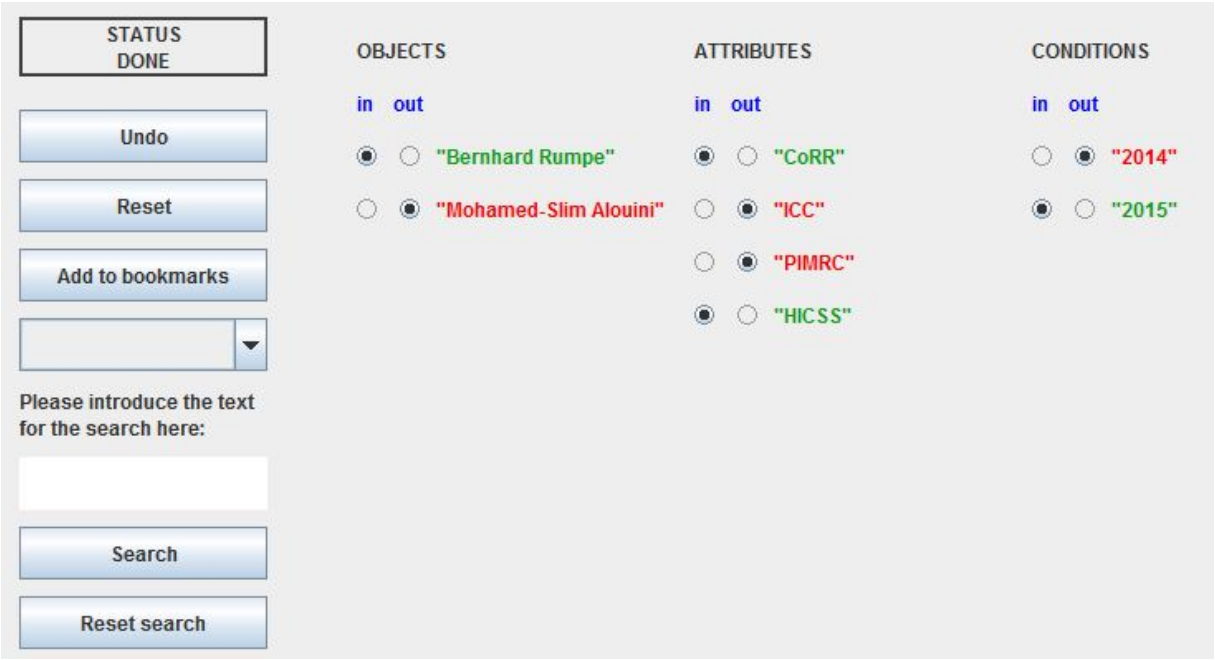


Figure 3.14: Screenshot navigation tool: final state

3.4.5.1 `ASP Navigation Tool`

The first navigation paradigm[4] uses the capabilities of Answer Set Programming for computing concepts and solving the corresponding membership constraint satisfaction problem [**Rudolph *et al.*, 2016**]. The encoding of membership constraints in ASP ([**Rudolph *et al.*, 2015a**]), described in Section 3.4.3, is such that given $\mathbb{K}$ and $\mathbb{C}$, an answer set program is created, such that there is a one-to-one correspondence between the answer sets and the n-concepts of $\mathbb{K}$ satisfying $\mathbb{C}$. For grounding and solving in the `ASP navigation tool` we used `Clingo` from the Potassco collection [Gebser *et al.*, 2011] (see Chapter 2.3 for more details about `Clingo`).

The simple propagation algorithm (Algorithm 2) needs to make multiple calls to the ASP solver, namely for each element that is in an undetermined state, two membership constraint satisfiability problems are generated, checking whether adding the element to the required objects, respectively to the forbidden objects, generates an unsatisfiable program. The end effect is a very long computation time for each propagation step.

However, the implementation of the ASP based tool can explore optimization strategies offered by ASP. Therefore, when analyzing the optimized propagation algorithm (Algorithm 3) we observe that cautious entailment with respect to the answer sets of an ASP program, computes the intersection over all answer sets, which is exactly what is needed for the propagation of the constraints. As a consequence, the implementation of the optimized propagation algorithm using the *cautious* option for `Clingo` requires a single call to the ASP solver. In comparison to the simple propagation algorithm, the optimized version proves to drastically decrease the computation time as well as the memory usage, hence improving the performance of the whole interactive navigation tool. Detailed experimental results are described in the evaluation section.

One of the main advantages of the ASP navigation approach is that it can be easily extended to any n-ary dataset. In order to prove this, we implement the `ASP navigation tool` for the cases $n \in \{2, 3, 4\}$. In the implementation we observe that the only modifications that need to be made when extending the dyadic case to the triadic and tetradic one are the following:

- update the ASP encoding for a specific n (easily done by following the general encoding description or by symmetry from the other smaller-dimensional cases)
- update the context loader, which depends, besides on the dimension of the context, also on the format of the n-adic context for a given n; for dimensions $n \geq 3$ we use the `csv`-format, however for dyadic contexts we use the common `cxt`-format

[4] tool available at `https://sourceforge.net/projects/asp-concept-navigation`

- update the graphical interface to contain a column for each element set of the context with similar functionalities, i.e. the option of choosing the state of each element as "in" or "out"

Following the model of the tetradic case, which was implemented as a proof of concept for extension, one can easily perform the necessary changes described here for navigating in n-ary formal contexts for any $n \geq 2$.

3.4.5.2 `Brute Force Navigation Tool`

The second approach is a brute force implementation based on an exhaustive search in the whole formal concept space, therefore we call it brute force navigation [**Rudolph *et al.*, 2016**]. The first main difference lies in the method of computing the formal concepts as well as in which navigation step the concepts are computed. Instead of computing them at each step using a declarative approach, in the brute force approach all formal concepts are computed in the preprocessing phase. Furthermore, the concept set is no longer computed using ASP, but by one of the existing FCA tools. In a navigation step an exhaustive search is necessary in order to select the subset of formal concepts that satisfy the constraints and compute the intersection. This subset of formal concepts is successively pruned in each navigation step until it contains a single concept, which represents the final state of the navigation.

With the purpose of comparing the two approaches, we implement the triadic case of the brute force navigation[5] with optimized propagation and use `Trias` [Jäschke *et al.*, 2006] to compute the triadic formal concepts in the preprocessing phase. For that reason, the input for the navigation tool is adapted to `Trias`' output format.

The main disadvantage of the brute force approach is that a different tool has to be available in order to compute the set of formal concepts. Furthermore, this tools usually do not have a high performance and the time needed for the preprocessing phase is drastically increased. A consequence of using an external FCA tool is the fact that a possible extension to an n-adic dataset depends on the extension of the tool used to compute the concepts. For `Trias`, the tool used in the triadic case, there is no extension available for higher-adic cases. However, Cerf et al. propose two algorithms `Data-Peeler` ([Cerf *et al.*, 2008; Cerf *et al.*, 2009]) and `Fenster` ([Cerf *et al.*, 2013]), claiming that these can compute formal concepts of n-ary contexts. From the two algorithms the latter was developed in 2013 as an extended, optimized version of the first. In our future work, we plan to implement the brute force navigation using `Fenster` or the preprocessing

[5] `https://sourceforge.net/projects/brute-force-concept-navigation`

phase and to analyze whether using this algorithm improves the extensibility of this approach. However, if the computation time for the formal concepts can not be drastically decreased, this approach becomes unusable, especially when dealing with dynamic and rapidly changing datasets.

3.4.6 Evaluation and Comparison of the ASP and the Brute Force Approach

In this section, we evaluate the performance of the ASP based and the brute force navigation paradigms in terms of implementation and computation speed [**Rudolph *et al.*, 2016**]. In order to evaluate the implemented tools, we ran experiments on the dblp database[6]. The dblp database indexes conference and journal publications and contains information such as author, title, year, volume, and journal/conference name. In order to compare the `ASP navigation tool` to the implemented `brute force navigation tool` one needs triadic datasets. The triadic structure that we chose for the experiments contains the author's name, conference/journal name and year of the publication. We extracted the described triadic dataset from the dblp mysql dump[7] and selected subsets of different dimensions. The subsets were selected by imposing restrictions on the number of publications per journal/conference, publication year and number of publications per author. For example, the dataset with 28 triples used in the following experiments can be obtained by following the next steps:

- eliminate all journals/conferences having less than 15000 publications
- eliminate all publications before the year 2014
- eliminate all entries for authors that published less than 150 papers

After selecting a triadic data subset, no preprocessing phase for the `ASP navigation tool` is needed, since its input must contain only the triadic relation. However, the `brute force navigation tool` requires a preprocessing phase. First the triconcept set needs to be computed with the `Trias` algorithm, hence the `Trias` tool[8] needs to be installed separately. If using the `Trias` algorithm without a database connection, the standard input file requires numeric data. Hence, in order to format the data according to the `Trias` tool input format, the elements of the dataset need to be indexed. After running `Trias` to obtain the triconcepts, the output needs to be formatted again before using the

[6] `http://dblp.uni-trier.de/`
[7] `http://dblp.l3s.de/dblp++.php`
[8] `https://github.com/rjoberon/trias-algorithm`

Table 3.2: Datasets used in the experiments

dataset	object nr.	attribute nr.	condition nr.	triples nr.
1	2	15	2	28
2	14	62	5	680
3	41	67	7	2514
4	68	67	8	4478
5	83	67	9	5987
6	108	67	10	8133

`brute force navigation tool`. Mainly the dimensions and encodings of the object, attribute and condition sets need to be added, so that the navigation tool can output the names of the elements and not their indexes in the user interface. Only after these preprocessing steps can a user interactively navigate in the tricontext using the `brute force navigation tool`. Obviously, different formats for the input of the navigation tool can be implemented, but for the purpose of comparing the two tools we implemented one single input format based on the standard `Trias` output.

For measuring the runtimes of the two navigation tools, we evaluated their performance on six different datasets (obtained from the dblp database as described previously) containing between 28 and 8133 triples. The datasets described in Table 3.2 contain triples where objects are identified with author names, attributes with conferences/journal names and conditions with the publication years. For each dataset we chose some random navigation paths through the data, which contain between 4 and 13 navigation steps and end when a final state, i.e., a formal concept, is reached. By navigation step we understand not only the action of a user choosing an element as *in* or *out*, but also the subsequent propagation phase. In order to compare the approaches we computed the average navigation step time (following the same navigation paths in all implementations) for each dataset and measured the time used for loading the data. This information can be obtained from the file *statistics.log* which is created as an output by the navigation tools. Furthermore, for the brute force navigation, we also measured the preprocessing time, i.e. the time that `Trias` needs to compute the triconcepts. Note that the time needed to index the dataset for the `Trias` input, as well as to add the encodings to the `Trias` output to obtain the input for the navigation tool, were excluded from this analysis, since this processing phase could be avoided by implementing different input/output formats for the `Trias` tool or for the `brute force navigation tool`. We denote the data

loading time plus the preprocessing time as offline time. In case of the `ASP navigation tool`, the offline time equals the data loading time, since no preprocessing is needed. The experiments were run on an Intel(R) Core(TM) I7-3630QM CPU @ 2.40 GHz machine with 4 GB RAM and 6M Cache.

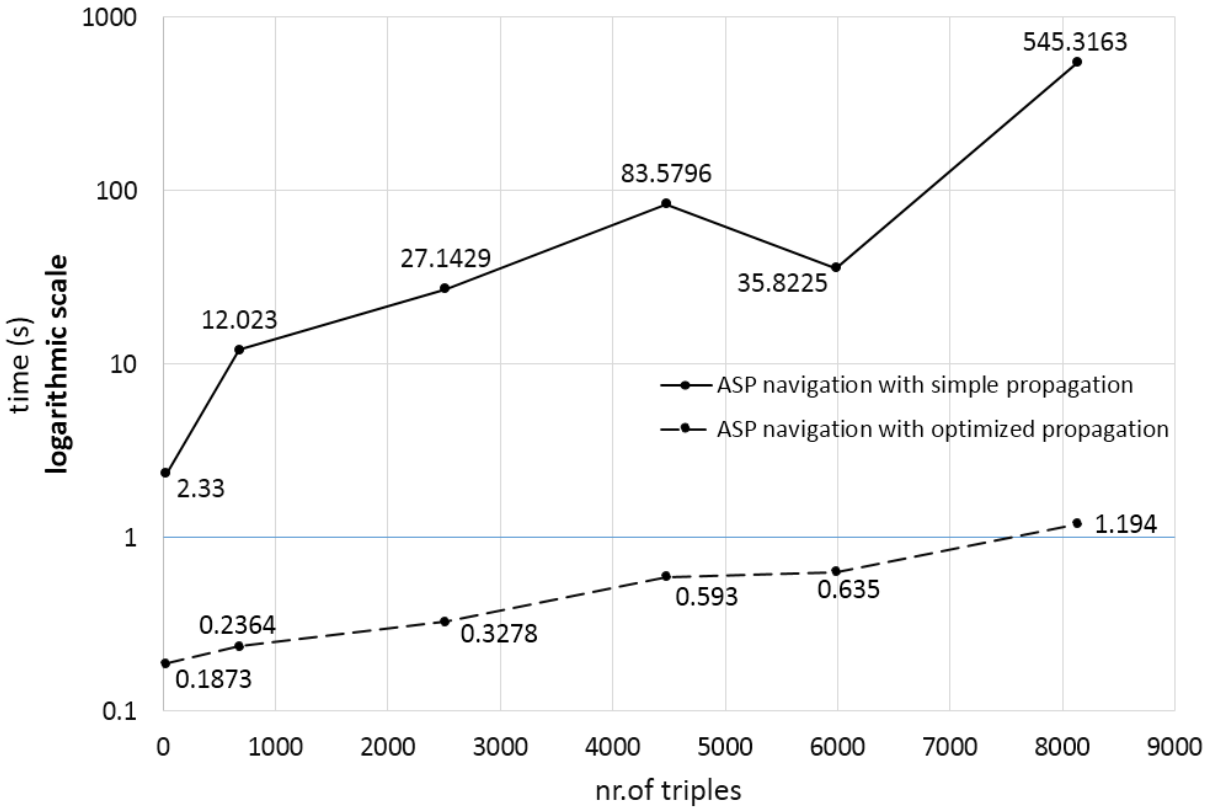


Figure 3.15: Average step time for ASP navigation with simple propagation vs optimized propagation

First we compare the different propagation implementations for the ASP approach: simple propagation vs. optimized propagation. The results are shown in Figure 3.15, where the y-axis depicts the logarithmically scaled time of execution, while the x-axis corresponds to the size of the relation. Besides the big difference in the execution time of each step, the `ASP navigation tool` with simple propagation uses a lot of memory. For the context with 8133 triples after a few navigation steps the execution was stopped by the system because it reached the limit memory of 4 GB RAM, hence the average time was computed on the steps executed so far. In comparison, this problem did not occur for the navigation tool with optimized propagation.

Table 3.3: ASP Navigator with optimized propagation experiments

object nr.	attribute nr.	condition nr.	triples nr.	ASP navigation data loading time (s)	ASP navigation average step time (s)
2	15	2	28	0.015	0.1873
14	62	5	680	0.109	0.2315
41	67	7	2514	0.374	0.3278
68	67	8	4478	0.546	0.593
83	67	9	5987	0.66	0.635
108	67	10	8133	1.07	1.194

Table 3.4: Brute Force Navigator with optimized propagation experiments

object nr.	attribute nr.	condition nr.	triples nr.	Trias preprocessing time (s)	brute force navigation data loading time (s)	brute force navigation average step time (s)
2	15	2	28	0.27	0.016	0.006
14	62	5	680	1.04	0.421	0.0047
41	67	7	2514	23.24	1.95	0.0219
68	67	8	4478	644.758	4.384	0.053
83	67	9	5987	2152.839	6.992	0.16
108	67	10	8133	$>$ 2h		

Next, we ran experiments on the same datasets (see Table 3.2) to compare the `ASP navigation tool` to the `brute force navigation tool`, both with optimized propagation. Table 3.3 describes the experimental results for the ASP navigation, while in Table 3.4 we can see the results for the brute-force navigation. Moreover, Figure 3.16 depicts the offline time of the `ASP navigation tool` vs. `the brute force navigation tool` on the logarithmically scaled y-axis in relation to the number of triples represented on the x-axis. As the chart shows, the offline time for the brute force navigation has a massive growth with respect to the size of the triadic relation, while the offline time for the `ASP navigation tool` has a more linear growth. When comparing the average step time, the `brute force navigation tool` has slightly better results than the `ASP navigation tool`, but, as shown in Figure 3.17, for subsets with less than 6000 triples the average step time is under 1 second for both approaches. Furthermore, from the experiments ran on the larger dataset, containing 8133 triples, it follows that the `ASP navigation tool` is still usable, with an average step time of 1.194 seconds, as opposed to the `brute force`

`navigation tool`, which turns out to have a very time consuming preprocessing phase: the `Trias` algorithm does not manage to compute the triconcept set in two hours.

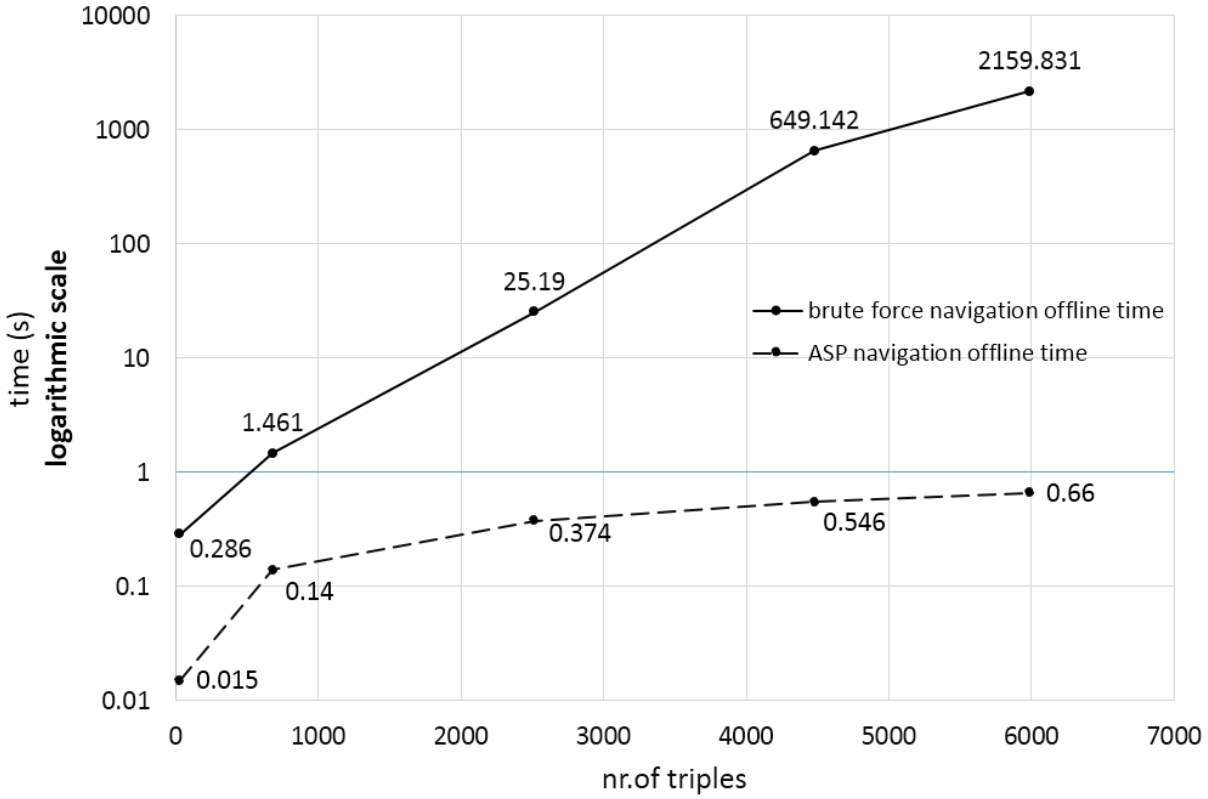
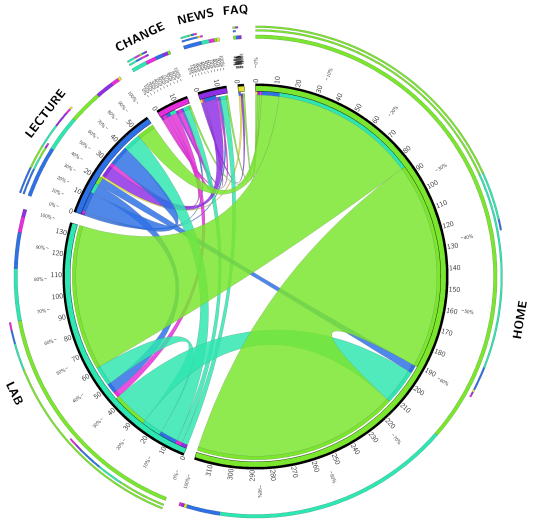


Figure 3.16: Offline time for `ASP` vs `brute force navigation tool` with respect to the number of triples in the relation

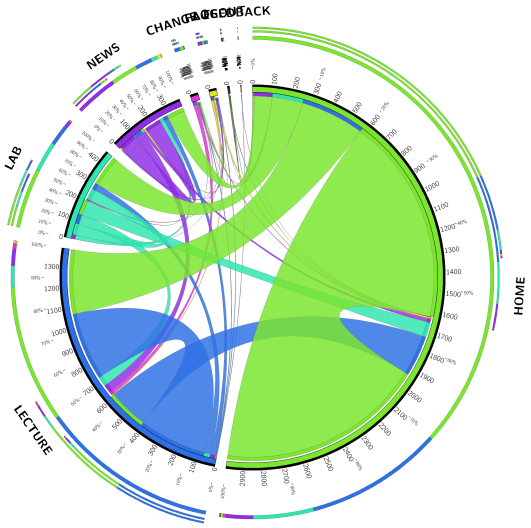
The experiments lead us to believe that for larger datasets the `ASP navigation tool` should be the preferred one, since it has a small execution time for loading the data, as well as for each navigation step, both of which are important for an interactive tool. Furthermore, in case of dynamic datasets that change frequently, it makes sense to use the `ASP navigation tool` which requires no preprocessing of the data.

Figure 3.6: Behavior types for the “ar” group in the SO1 course



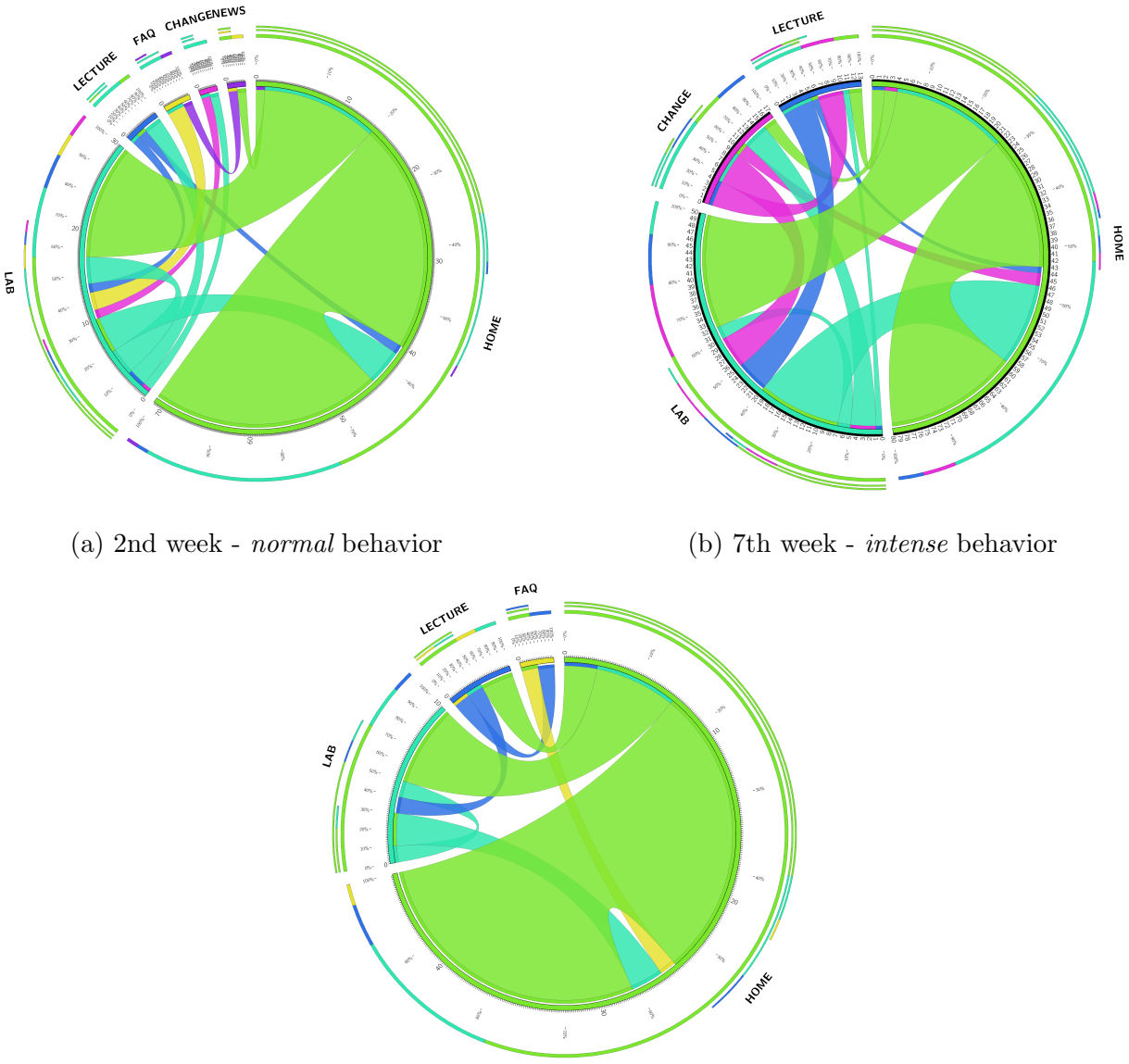
(a) 1st week - *normal* behavior

(b) 10th week - *relaxed* behavior



(c) 17th week - *intense* behavior

Figure 3.7: Behavior types for the "ie" group in the WDO course



(a) 2nd week - *normal* behavior

(b) 7th week - *intense* behavior

(c) 14th week - *relaxed* behavior

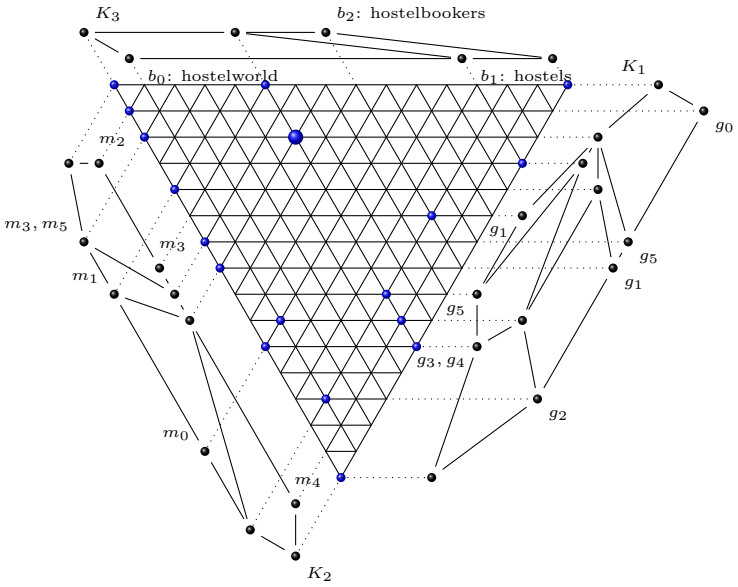


Figure 3.8: Trilattice of the tricontext "Hostels"

Figure 3.11: Lattice of reachability context

2014	*Corr*	*ICC*	*PIMRC*	*HICSS*
Rumpe	×			
Alouni	×	×	×	

2015	*Corr*	*ICC*	*PIMRC*	*HICSS*
Rumpe	×			×
Alouni	×	×		

Figure 3.12: dblp data: author, conference/journal, year

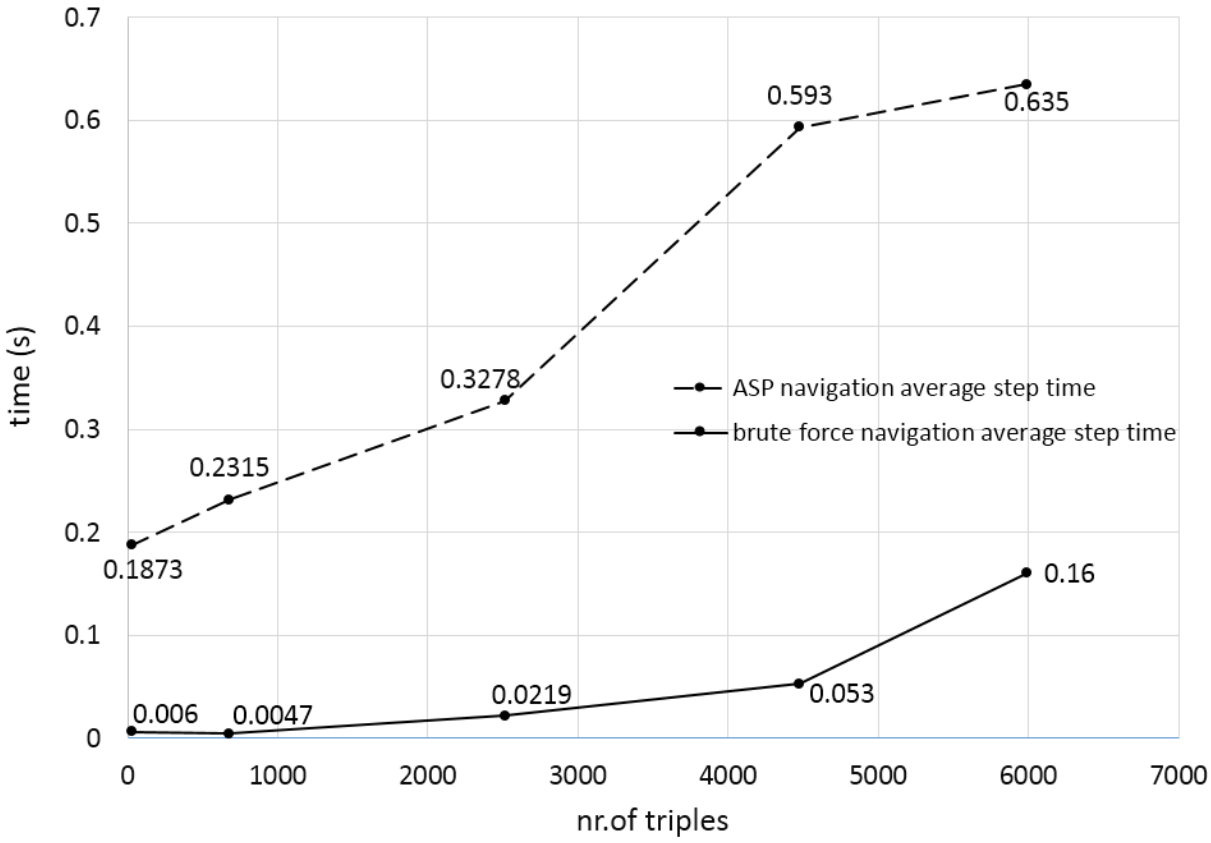


Figure 3.17: Average step time for `ASP` vs `brute force navigation tool` with respect to the number of triples in the relation

4. Conclusions and Future Work

4.1 Achievements

The aim of this thesis is to offer a deep insight into methods of analysis, visualization and navigation based on formal concept analysis for higher-dimensional datasets. The main scientific contributions are on one side theoretical and on the other side practical in the form of different visualization and navigation paradigms, a part of which were already implemented and evaluated through experiments. These contributions are summarized in the next paragraphs.

In this thesis, we discussed theoretical aspects of higher-adic formal contexts and we extended notions (such as clarification and reduction) from the dyadic to the triadic case. Given that the motivation was a practical one, we implemented the clarification and reduction processes and tested them on a cancer registry database. These experiments highlighted the importance of a preprocessing phase of the data, that includes removing redundant data through processes such as clarification and reduction, before any further analysis. This not only improves the efficiency of the analysis, but it can also improve the readability of the data since the size of the dataset is drastically decreased.

Driven by practical requirements, the rest of the thesis focused on visualization and navigation methods of higher-adic datasets. The first method we proposed addressed the problem of visualizing triadic data in a form that enables the detection of temporal patterns in the data. Our main purpose was to analyze the data logs of an e-learning platform in order to better understand the behavior of students during a semester. Therefore, we chose different triadic structures from the available dataset and used a circular layout to visualize the correlations within different data subsets. This enabled us to identify trend-setters among the students and to correlate the behavior of the students to different course-related activities.

Next we focused on paradigms that offered not only visualizations of the data, but also the possibility to navigate among concepts. Motivated by the intuitive navigation in a dyadic concept lattice, we proposed a first paradigm for triadic datasets based on

appropriately defined dyadic projections and a reachability relation among concepts. The navigation is such that the user can navigate either within the dyadic lattice corresponding to a dyadic projection or from one dyadic lattice to another one, depending on the perspective that he chooses. Herefrom, we observed that the reachability relation forms a partition on the concept set consisting of so-called reachability clusters. This led us to investigate properties of the reachability clusters. We introduced the theoretical basis for the navigation paradigm based on the reachability relation and described the exploration strategy and possible implementation methods in more detail.

The second navigation paradigm proposed in this thesis follows a general interactive framework based on the idea that the user can add constraints that gradually restrict the space of the concepts, leading in the final step to a single concept which satisfies all the constraints chosen by the user. This navigation paradigm has the important advantage that the framework was designed in a user-friendly manner, i.e. it can be used by users without any knowledge of formal concept analysis. In order to formally define the framework we have defined membership constraints and the corresponding satisfiability problems. The computational complexities of the satisfiability problems were studied for different dimensions of the datasets (dyadic, triadic and n-adic) and we have proved that although in some cases the complexity is low (trivial or AC^0), there are cases when it becomes intractable, i.e. NP-complete. This led us to consider answer set programming for the implementation of the framework, since it offers highly optimized tools for solving NP-complete satisfiability problems. For that purpose we presented an encoding in answer set programming for membership constraints. This implementation strategy has the advantage of being easily extended to any n-ary dataset. In order to prove this, we implemented the tool for dyadic, triadic and tetradic datasets.

Next we considered a second implementation strategy for the navigation framework based on membership constraints. This strategy uses an external FCA tool to compute the whole concept set and then performs exhaustive search in the concept set in order to narrow down the subset of concepts that satisfy the given constraints. However, we observed that this approach is not always scalable, since it depends on the external tool used for computing the concepts.

Finally, we ran experiments in order to compare the execution times of the two implementation strategies and analyzed optimization methods for the framework. Experimental results demonstrated that the ASP navigation tool can be successfully used for large datasets (having low execution times of under two seconds for each navigation step). The ASP-based and the exhaustive-search-based tools that implement the membership constraints navigation are among the most important achievements of the thesis, since

they represent data analysis tools that use the capabilities of FCA while at the same time being intuitive and user-friendly (i.e, using the tools does not require previous theoretical knowledge). For higher-adic FCA, these are, to the best of our knowledge, the first navigation tools which allow to explore, search, recognize, identify, analyze, and investigate polyadic concept sets, by using membership constraints together with the conceptual knowledge processing paradigm.

4.2 Open Issues and Future Work

This thesis proposes a variety of visualization and navigation methods for higher-adic datasets. For each of the proposed paradigms there are some open issues regarding either theoretical aspects, i.e. properties that arise form the defined paradigms, or practical aspects that still need to be discussed, such as different implementation methods or evaluations of the tools.

For the navigation approach based on three relatedness notions corresponding to the three dimensions of the context, we have investigated the properties arising from this reachability relation. As it turned out, in some datasets, not every concept can be reached from every other concept. In the experiments that we ran on real datasets this was not the case, which leads us to believe that the structure of real data is such that often all concepts are mutually reachable. However, in general, the reachability relation gives rise to reachability clusters obtained as maximal sets of mutually reachable concepts. Regarding reachability between two clusters C_1 and C_2, there are two possible situations: either concepts of the cluster C_2 are reachable from the concepts of the cluster C_1, but concepts of cluster C_1 are not reachable from the ones in C_2 (or the other way around), or there is no reachability relation between any elements of the two clusters. Consequently, clusters are ordered by unidirectional reachability and form a partial order which we found to always have a greatest element. Herefrom, we conclude that navigation should start in a minimal cluster. However, some examples show that it can be the case that there are several minimal clusters, hence choosing the starting point of the navigation remains an open question.

Currently, we are working on defining a navigation paradigm that combines the two navigation approaches proposed in this thesis, i.e. navigation based on the reachability relation and navigation based on membership constraints, which would offer a possible solution for choosing the starting point of the navigation. The idea is to use the constraints based navigation in order to determine the starting point of the exploration and, from then on, use the navigation based on the reachability relation for exploring the dataset.

However, in order to help the user visualize the constraints he chose in the beginning throughout the whole exploration, in each dyadic projection we highlight concepts that satisfy the constraints, using for example the green color for representing them. We intend to analyze different implementation strategies for this combined navigation paradigm and implement it in order to see if it is an efficient and appropriate solution for the above mentioned problem.

Another open issue relates to the theoretical aspects of the reachability clusters. While analyzing the properties of the clusters, we have tried to find a possible correlation between the dimension of the context and the number of reachability clusters. However, some initial conjectures about upper bounds or the existence of suprema had to be refuted by counterexamples, which nevertheless provided some interesting structural insights and may pave the way to further investigations. As of yet, the only (and trivial) upper bound for the number of reachability clusters is the number of triconcepts, which may be exponential in the size of the tricontext. We, however, still conjecture that there is a polynomial bound.

Besides these open theoretical questions, future work on the topic has to include an implementation of the described navigation paradigm and user studies in order to confirm our hypothesis that this way of displaying and browsing the space of triconcepts is indeed accessible and intuitive for human users.

For the second navigation framework, which is based on membership constraints, many directions for future work can be identified. Regarding the theoretical aspects of membership constraints, we intend to further analyze the computational complexities of the proper satisfiability problems and investigate whether some of the tractable cases of satisfiability problems become intractable for proper satisfiability. In addition, regarding aspects of implementation of the framework, as a first step in our future work we intend to test the brute force navigation tool using different FCA tools for computing the formal concepts, such as `Data-Peeler` ([Cerf *et al.*, 2008; Cerf *et al.*, 2009]) or `Fenster` ([Cerf *et al.*, 2013]). The authors of these tools claim that both are scalable and can compute the formal concepts of any n-ary dataset. We intend to run extensive experiments on datasets of different sizes in order to see if these tools improve the computation time of the concept set, turning the brute force navigation approach into a feasible exploration method for larger datasets. Moreover, we intend to compare the different implementations on several datasets and investigate possible correlations between the performance of the two main approaches, the ASP based approach and the exhaustive search based approach, and the properties of the datasets, such as size and density.

As an application of the ASP based navigation, we are currently continuing our anal-

ysis on the data logs of the e-learning platform PULSE. In Chapter 3.1, we have studied this data in a triadic setting and identified behavioral patterns of the students correlated to the timeline of the course. Now, we want to analyze the data from a tetradic and pentadic perspective, in order to detect repetitive browsing habits. Therefore, we use formal concept analysis tools along with answer set programming in order to determine trend-setters and followers. Intuitively, trend-setters are the users which firstly adhere to a specific behavior and followers are the users that copy this behavior. The trend-setter and followers can be identified from a pentadic setting in which we compare groups of students among themselves and also when the behavior that they have in common occurred for each of them. With that purpose, we extend the ASP encoding for computing formal concepts in pentadic datasets and, after discovering interesting patterns in the data, we use the ASP navigation tool in order to take a closer look at students that stand out in the previous analysis. Furthermore, we intend to investigate whether trend-setters influence the entire evolution of their followers over time. We believe that the techniques described in this analysis will reveal potential hidden patterns in the Web logs that would suggest improvements that can be made to the portal, eventually leading to a more efficient interaction between the students.

Finally, as a long term goal, we will focus, in addition to the already described paradigms, also on new methods of visualization, navigation and exploration which might improve some aspects of the approaches proposed in this thesis.

List of my Publications

[Dragoş *et al.*, 2014a] Sanda Dragoş, Diana Haliţă, Christian Săcărea, and Diana Troancă. An FCA Grounded Study of User Dynamics through Log Exploration. *Studia Universitatis Babeş-Bolyai Series Informatica*, LIX(2):82–97, 2014.

[Dragoş *et al.*, 2014b] Sanda Dragoş, Diana Haliţă, Christian Săcărea, and Diana Troancă. Applying Triadic FCA in Studying Web Usage Behaviors. In Robert Buchmann, Claudiu Vasile Kifor, and Jian Yu, editors, *Proceedings of the 7th International Conference on Knowledge Science, Engineering and Management, KSEM 2014, Sibiu, Romania*, volume 8793 of *Lecture Notes in Computer Science*, pages 73–80. Springer, 2014.

[Rudolph *et al.*, 2015a] Sebastian Rudolph, Christian Săcărea, and Diana Troancă. Membership Constraints in Formal Concept Analysis. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the 24th International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina*, pages 3186–3192. AAAI Press, 2015.

[Rudolph *et al.*, 2015b] Sebastian Rudolph, Christian Săcărea, and Diana Troancă. Reduction in Triadic Data Sets. In Sergei O. Kuznetsov, Amedeo Napoli, and Sebastian Rudolph, editors, *Proceedings of the 4th International Workshop "What can FCA do for Artificial Intelligence", FCA4AI 2015, co-located with the International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina*, volume 1430 of *CEUR Workshop Proceedings*, pages 55–62. CEUR-WS.org, 2015.

[Rudolph *et al.*, 2015c] Sebastian Rudolph, Christian Săcărea, and Diana Troancă. Towards a Navigation Paradigm for Triadic Concepts. In Jaume Baixeries, Christian Săcărea, and Manuel Ojeda-Aciego, editors, *Proceedings of the 13th International Conference on Formal Concept Analysis, ICFCA 2015, Nerja, Spania*, volume 9113 of *Lecture Notes in Computer Science*, pages 232–248. Springer, 2015.

[Rudolph *et al.*, 2016] Sebastian Rudolph, Christian Săcărea, and Diana Troancă. Conceptual Navigation for Polyadic Formal Concept Analysis. Submitted to the 4th Workshop on Artificial Intelligence for Knowledge Management, AI4KM 2016, co-located with the International Joint Conference on Artificial Intelligence, IJCAI 2016, paper available at `http://www.cs.ubbcluj.ro/~dianat/publications/RuTr1.pdf`, 2016.

Bibliography

[Agrawal and Jagadish, 1987] Rakesh Agrawal and Hosagrahar V. Jagadish. Direct Algorithms for Computing the Transitive Closure of Database Relations. In Peter M. Stocker, William Kent, and Peter Hammersley, editors, *Proceedings of the 13th International Conference on Very Large Data Bases, VLDB 1987, San Francisco, CA, USA*, pages 255–266. Morgan Kaufmann Publishers Inc., 1987.

[Andrews and Orphanides, 2010] Simon Andrews and Constantinos Orphanides. FcaBedrock, a Formal Context Creator. In Madalina Croitoru, Sébastien Ferré, and Dickson Lukose, editors, *Proceedings of the 18th International Conference on Conceptual Structures, ICCS 2010, Kuching, Sarawak, Malaysia*, volume 6208 of *Lecture Notes in Computer Science*, pages 181–184. Springer, 2010.

[Andrews, 2009] Simon Andrews. In-Close, a Fast Algorithm for Computing Formal Concepts. In Sebastian Rudolph, Frithjof Dau, and Sergei O. Kuznetsov, editors, *Supplementary Proceedings of the 17th International Conference on Conceptual Structures, ICCS 2009, Moscow, Russia*, volume 483 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[Andrews, 2011] Simon Andrews. In-Close2, a High Performance Formal Concept Miner. In Simon Andrews, Simon Polovina, Richard Hill, and Babak Akhgar, editors, *Proceedings of the 19th International Conference on Conceptual Structures, ICCS 2011, Derby, England*, volume 6828 of *Lecture Notes in Computer Science*, pages 50–62. Springer, 2011.

[Arora and Barak, 2009] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[Becker and Correia, 2005] Peter Becker and Joachim Hereth Correia. The ToscanaJ Suite for Implementing Conceptual Information Systems. In Bernhard Ganter, Gerd Stumme, and Rudolf Wille, editors, *Formal Concept Analysis, Foundations and Appli-*

cations, volume 3626 of *Lecture Notes in Computer Science*, pages 324–348. Springer, 2005.

[Becker *et al.*, 2002] Peter Becker, Joachim Hereth, and Gerd Stumme. ToscanaJ: An Open Source Tool for Qualitative Data Analysis. In Vincent Duquenne, Bernhard Ganter, Michel Liquiere, Engelbert M. Nguifo, and Gerd Stumme, editors, *Proceedings of the Workshop Advances in Formal Concept Analysis for Knowledge Discovery in Databases, FCAKDD 2002, co-located with the European Conference on Artificial Intelligence, ECAI 2002, Lyon, France*, pages 1–2, 2002.

[Berry *et al.*, 2007] Anne Berry, Jean Paul Bordat, and Alain Sigayret. A Local Approach to Concept Generation. *Annals of Mathematics and Artificial Intelligence*, 49(1-4):117–136, 2007.

[Birkhoff, 1940] Garrett Birkhoff. *Lattice Theory*. New York: American Mathematical Society, 1940.

[Borchmann, 2012] Daniel Borchmann. A Generalized Next-Closure Algorithm - Enumerating Semilattice Elements from a Generating Set. In Laszlo Szathmary and Uta Priss, editors, *Proceedings of The 9th International Conference on Concept Lattices and Their Applications, CLA 2012, Fuengirola (Málaga), Spain*, volume 972 of *CEUR Workshop Proceedings*, pages 9–20. CEUR-WS.org, 2012.

[Calimeri *et al.*, 2016] Francesco Calimeri, Martin Gebser, Marco Maratea, and Francesco Ricca. Design and Results of the Fifth Answer Set Programming Competition. *Artificial Intelligence*, 231:151–181, 2016.

[Cerf *et al.*, 2008] Loc Cerf, Jrmy Besson, Cline Robardet, and Jean-Francois Boulicaut. Data-Peeler: Constraint-based Closed Pattern Mining in n-ary Relations. In Chid Apte, Haesun Park, Ke Wang, and Mohammad J. Zaki, editors, *Proceedings of SIAM International Conference on Data Mining, SDM 2008, Atlanta, Georgia*, pages 37–48. SIAM, 2008.

[Cerf *et al.*, 2009] Loïc Cerf, Jérémy Besson, Céline Robardet, and Jean-François Boulicaut. Closed Patterns Meet n-ary Relations. *ACM Transactions on Knowledge Discovery from Data, TKDD*, 3(1):3:1–3:36, 2009.

[Cerf *et al.*, 2013] Loïc Cerf, Jérémy Besson, Kim-Ngan Nguyen, and Jean-François Boulicaut. Closed and Noise-tolerant Patterns in n-ary Relations. *Data Mining and Knowledge Discovery*, 26(3):574–619, 2013.

[Cooley *et al.*, 1999] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data Preparation for Mining World Wide Web Browsing Patterns. *Knowledge and Information Systems*, 1(1):5–32, 1999.

[Dragoş and Beldean, 2013] Sanda Dragoş and Alina Mihaela Beldean. Analysing Web Usage with Force-Directed Graphs. *Studia Universitatis Babeş-Bolyai Series Informatica*, LVIII(4):75–86, 2013.

[Dragoş and Dragoş, 2009a] Sanda Dragoş and Radu Dragoş. WATEC: A Web Analytics Tool for Educational Content. In Militon Frenţiu and Horia F. Pop, editors, *Proceedings of the 2nd Knowledge Engineering: Principles and Techniques Conference, KEPT 2009, Cluj-Napoca, Romania*, pages 320–327. Presa Universitara Clujeana, 2009.

[Dragoş and Dragoş, 2009b] Sanda Dragoş and Radu Dragoş. Web Analytics for Educational Content. *Studia Universitatis Babeş-Bolyai Series Informatica*, Special Issue KEPT-2009: Knowledge Engineering: Principles and Techniques(2):268–271, 2009.

[Dragoş and Săcărea, 2012] Sanda Dragoş and Christian Săcărea. Analysing the Usage of Pulse Portal with Formal Concept Analysis. *Studia Universitatis Babeş-Bolyai Series Informatica*, LVII(3):65–75, 2012.

[Dragoş, 2007] Sanda Dragoş. PULSE - a PHP Utility used in Laboratories for Student Evaluation. In Petros Kefalas, Anna Sotiriadou, Gordon Davies, and Andrew McGettrick, editors, *Proceedings of the 2nd International Conference on Informatics Education Europe, IEEII 2007, Thessaloniki, Greece*, pages 306–314. South-East European Research Center (SEERC), 2007.

[Dragoş, 2009] Sanda Dragoş. PULSE Extended. In Mark Perry, Hideyasu Sasaki, Matthias Ehmann, Guadalupe Ortiz Bellot, and Oana Dini, editors, *Proceedings of the 4th International Conference on Internet and Web Applications and Services, ICIW 2009, Venice/Mestre, Italy*, pages 510–515. IEEE, 2009.

[Dragoş, 2010] Sanda Dragoş. Current Extensions on PULSE. *Studia Universitatis Babeş-Bolyai Series Informatica*, LV(3):51–60, 2010.

[Dragoş, 2011] Sanda Dragoş. Why Google Analytics Cannot be Used for Educational Web Content. In Ajith Abraham, Emilio Corchado, Sang-Yong Han, Weisen Guo, Juan Corchado, and Athanasios Vasilakos, editors, *Proceedings of the 7th International Conference on Next Generation Web Services Practices, NWeSP 2011, Salamanca, Spain*, pages 113–115. IEEE, 2011.

[Eirinaki and Vazirgiannis, 2003] Magdalini Eirinaki and Michalis Vazirgiannis. Web Mining for Web Personalization. *ACM Transactions on Internet Technology, TOIT*, 3(1):1–27, 2003.

[Ferré and Ridoux, 2004] Sébastien Ferré and Olivier Ridoux. Introduction to Logical Information Systems. *Information Processing & Management*, 40(3):383–419, 2004.

[Ganter and Wille, 1999] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis - Mathematical Foundations*. Springer, 1999.

[Ganter, 1984] Bernhard Ganter. Two Basic Algorithms in Concept Analysis. FB4-Preprint 831, 1984.

[Gebser *et al.*, 2011] Martin Gebser, Benjamin Kaufmann, Roland Kaminski, Max Ostrowski, Torsten Schaub, and Marius Thomas Schneider. Potassco: The Potsdam Answer Set Solving Collection. *AI Communications*, 24(2):107–124, 2011.

[Gebser *et al.*, 2012] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers, 2012.

[Gelfond and Lifschitz, 1988] Michael Gelfond and Vladimir Lifschitz. The Stable Model Semantics For Logic Programming. In Robert Kowalski and Kenneth A. Bowen, editors, *Proceedings of the Joint International Logic Programming Conference and Symposium, JICSLP 1988, Manchester, England*, pages 1070–1080. MIT Press, 1988.

[Glodeanu, 2013] Cynthia V. Glodeanu. Tri-ordinal Factor Analysis. In Peggy Cellier, Felix Distel, and Bernhard Ganter, editors, *Proceedings of the 11th International Conference on Formal Concept Analysis, ICFCA 2013, Dresden, Germany*, volume 7880 of *Lecture Notes in Computer Science*, pages 125–140. Springer, 2013.

[Gnatyshak *et al.*, 2013] Dmitry Gnatyshak, Dmitry I. Ignatov, and Sergei O. Kuznetsov. From Triadic FCA to Triclustering: Experimental Comparison of Some Triclustering Algorithms. In Manuel Ojeda-Aciego and Jan Outrata, editors, *Proceedings of the 10th International Conference on Concept Lattices and Their Applications, CLA 2013, La Rochelle, France*, volume 1062 of *CEUR Workshop Proceedings*, pages 249–260. CEUR-WS.org, 2013.

[Godin *et al.*, 1993] 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(5):747–767, 1993.

[Godin *et al.*, 1995] Robert Godin, Rokia Missaoui, and Hassan Alaoui. Incremental Concept Formation Algorithms Based on Galois (Concept) Lattices. *Computational Intelligence*, 11(2):246–267, 1995.

[Immerman, 1999] Neil Immerman. *Descriptive Complexity*. Graduate texts in computer science. Springer, 1999.

[Jäschke *et al.*, 2006] Robert Jäschke, Andreas Hotho, Christoph Schmitz, Bernhard Ganter, and Gerd Stumme. TRIAS - An Algorithm for Mining Iceberg Tri-Lattices. In Christopher W. Clifton, Ning Zhong, Jiming Liu, Benjamin W. Wah, and Xindong Wu, editors, *Proceedings of the 6th IEEE International Conference on Data Mining, ICDM 2006, Hong Kong, China*, pages 907–911. IEEE, 2006.

[Jäschke *et al.*, 2008] Robert Jäschke, Andreas Hotho, Christoph Schmitz, Bernhard Ganter, and Gerd Stumme. Discovering Shared Conceptualizations in Folksonomies. *Journal of Web Semantics*, 6(1):38–53, 2008.

[Jelassi *et al.*, 2012] Mohamed Nader Jelassi, Sadok Ben Yahia, and Engelbert Mephu Nguifo. A Scalable Mining of Frequent Quadratic Concepts in d-Folksonomies. *Computing Research Repository, CoRR*, abs/1212.0087, 2012.

[Ji *et al.*, 2006] Liping Ji, Kian-Lee Tan, and Anthony K. H. Tung. Mining Frequent Closed Cubes in 3D Datasets. In Umeshwar Dayal, Kyu-Young Whang, David B. Lomet, Gustavo Alonso, Guy M. Lohman, Martin L. Kersten, Sang Kyun Cha, and Young-Kuk Kim, editors, *Proceedings of the 32nd International Conference on Very Large Data Bases, VLDB 2006, Seoul, Korea*, pages 811–822. ACM, 2006.

[Kosala and Blockeel, 2000] Raymond Kosala and Hendrik Blockeel. Web Mining Research: A Survey. *SIGKDD Explorations*, 2(1):1–15, 2000.

[Krajca *et al.*, 2008] Petr Krajca, Jan Outrata, and Vilem Vychodil. Parallel recursive algorithm for FCA. In Radim Belohlavek and Sergei O. Kuznetsov, editors, *Proceedings of the 6th International Conference on Concept Lattices and Their Applications, CLA 2008, Olomouc, Czech Republic*, volume 433 of *CEUR Workshop Proceedings*, pages 71–82. CEUR-WS.org, 2008.

[Krajca *et al.*, 2010] Petr Krajca, Jan Outrata, and Vilém Vychodil. Advances in Algorithms Based on CbO. In Marzena Kryszkiewicz and Sergei A. Obiedkov, editors, *Proceedings of the 7th International Conference on Concept Lattices and Their Applications, CLA 2010, Sevilla, Spain*, volume 672 of *CEUR Workshop Proceedings*, pages 325–337. CEUR-WS.org, 2010.

[Kuznetsov and Obiedkov, 2002] Sergei O. Kuznetsov and Sergei A. Obiedkov. Comparing Performance of Algorithms for Generating Concept Lattices. *Journal of Experimental and Theoretical Artificial Intelligence*, 14(2-3):189–216, 2002.

[Kuznetsov, 1999] Sergei O. Kuznetsov. Learning of Simple Conceptual Graphs from Positive and Negative Examples. In Jan M. Zytkow and Jan Rauch, editors, *Proceedings of the 3rd European Conference on Principles of Data Mining and Knowledge Discovery, PKDD 1999, Prague, Czech Republic*, volume 1704 of *Lecture Notes in Computer Science*, pages 384–391. Springer, 1999.

[Lehmann and Wille, 1995] Fritz Lehmann and Rudolf Wille. A Triadic Approach to Formal Concept Analysis. In Gerard Ellis, Robert Levinson, William Rich, and John F. Sowa, editors, *Proceedings of the 3rd International Conference on Conceptual Structures, ICCS 1995, Santa Cruz, California, USA*, volume 954 of *Lecture Notes in Computer Science*, pages 32–43. Springer, 1995.

[Lindig, 2000] Christian Lindig. Fast Concept Analysis. In *Working with Conceptual Structures - Contributions to ICCS 2000*, pages 152–161. Shaker Verlag, 2000.

[Norguet *et al.*, 2007] Jean-Pierre Norguet, Benjamin Tshibasu-Kabeya, Gianluca Bontempi, and Esteban Zimányi. A Page-Classification Approach to Web Usage Semantic Analysis. *Engineering Letters*, 14(1):120–126, 2007.

[Nourine and Raynaud, 1999] Lhouari Nourine and Olivier Raynaud. A Fast Algorithm for Building Lattices. *Information Processing Letters*, 71(5-6):199–204, 1999.

[Outrata and Vychodil, 2012] Jan Outrata and Vilém Vychodil. Fast Algorithm for Computing Fixpoints of Galois Connections Induced by Object-Attribute Relational Data. *Information Sciences*, 185(1):114–127, 2012.

[Papadimitriou, 1994] Christos H. Papadimitriou. *Computational Complexity*. Theoretical Computer Science. Addison-Wesley, 1994.

[Perkowitz and Etzioni, 1997] Mike Perkowitz and Oren Etzioni. Adaptive Web Sites: An AI Challenge. In Martha E. Pollack, editor, *Proceedings of the 15th International Joint Conference on Artificial Intelligence, IJCAI 1997, Nagoya, Aichi, Japan*, pages 16–23. Morgan Kaufmann Publishers Inc., 1997.

[Pisková and Horváth, 2013] Lenka Pisková and Tomás Horváth. Comparing Performance of Formal Concept Analysis and Closed Frequent Itemset Mining Algorithms

on Real Data. In Manuel Ojeda-Aciego and Jan Outrata, editors, *Proceedings of the 10th International Conference on Concept Lattices and Their Applications, CLA 2013, La Rochelle, France*, volume 1062 of *CEUR Workshop Proceedings*, pages 299–304. CEUR-WS.org, 2013.

[Priss, 2008] Uta Priss. FcaStone - FCA File Format Conversion and Interoperability Software. In Madalina Croitoru, Robert Jschke, and Sebastian Rudolph, editors, *Proceedings of the 3rd Conceptual Structures Tool Interoperability Workshop, CS-TIW 2008, co-located with the International Conference on Conceptual Structures , ICCS 2008, Toulouse, France*, CEUR Workshop Proceedings, pages 33–43. CEUR-WS.org, 2008.

[Romero and Ventura, 2007] Cristóbal Romero and Sebastián Ventura. Educational Data Mining: A Survey from 1995 to 2005. *Expert Systems with Applications*, 33(1):135–146, 2007.

[Romero *et al.*, 2009] Cristóbal Romero, Sebastián Ventura, Amelia Zafra, and Paul De Bra. Applying Web Usage Mining for Personalizing Hyperlinks in Web-based Adaptive Educational Systems. *Computers & Education*, 53(3):828–840, 2009.

[Romero *et al.*, 2013] Cristobal Romero, Pedro G. Espejo, Amelia Zafra, Jose Raul Romero, and Sebastian Ventura. Web Usage Mining for Predicting Final Marks of Students that Use Moodle Courses. *Computer Applications in Engineering Education*, 21(1):135–146, 2013.

[Săcărea, 2014] Christian Săcărea. Investigating Oncological Databases Using Conceptual Landscapes. In Nathalie Hernandez, Robert Jäschke, and Madalina Croitoru, editors, *Proceedings of the 21st International Conference on Conceptual Structures, ICCS 2014, Iaşi, Romania*, volume 8577 of *Lecture Notes in Computer Science*, pages 299–304. Springer, 2014.

[Spiliopoulou and Faulstich, 1998] Myra Spiliopoulou and Lukas Faulstich. WUM - A Tool for WWW Utilization Analysis. In Paolo Atzeni, Alberto O. Mendelzon, and Giansalvatore Mecca, editors, *Selected Papers of the 1st International Workshop on the Web and Databases, WebDB 1998, Valencia, Spain*, volume 1590 of *Lecture Notes in Computer Science*, pages 184–103. Springer, 1998.

[Srivastava *et al.*, 2000] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, and Pang-Ning Tan. Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. *SIGKDD Explorations*, 1(2):12–23, 2000.

[Stumme *et al.*, 2002] Gerd Stumme, Rafik Taouil, Yves Bastide, Nicolas Pasquier, and Lotfi Lakhal. Computing Iceberg Concept Lattices with TITANIC. *Data & Knowledge Engineering*, 42(2):189–222, 2002.

[Trabelsi *et al.*, 2012] Chiraz Trabelsi, Nader Jelassi, and Sadok Ben Yahia. Scalable Mining of Frequent Tri-concepts from Folksonomies. In Pang-Ning Tan, Sanjay Chawla, Chin Kuan Ho, and James Bailey, editors, *Part II of the Proceedings of the 16th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, PAKDD 2012, Kuala Lumpur, Malaysia*, volume 7302 of *Lecture Notes in Computer Science*, pages 231–242. Springer, 2012.

[Valtchev *et al.*, 2003] Petko Valtchev, David Grosser, Cyril Roume, and Mohamed Rouane Hacene. Galicia: An Open Platform for Lattices. In Aldo de Moor, Wilfried Lex, and Bernhard Ganter, editors, *Proceedings of the 11th International Conference on Conceptual Structures, ICCS 2003, Dresden, Germany*, volume 2746 of *Lecture Notes in Computer Science*, pages 241–254. Springer, 2003.

[van der Merwe *et al.*, 2004] Dean van der Merwe, Sergei A. Obiedkov, and Derrick G. Kourie. AddIntent: A New Incremental Algorithm for Constructing Concept Lattices. In Peter W. Eklund, editor, *Proceedings of the 2nd International Conference on Formal Concept Analysis, ICFCA 2004, Sydney, Australia*, volume 2961 of *Lecture Notes in Computer Science*, pages 372–385. Springer, 2004.

[Voutsadakis, 2002] George Voutsadakis. Polyadic Concept Analysis. *Order*, 19(3):295–304, 2002.

[Wille, 1995] Rudolf Wille. The Basic Theorem of Triadic Concept Analysis. *Order*, 12(2):149–158, 1995.

[Yevtushenko, 2000] Serhiy A. Yevtushenko. System of Data Analysis "Concept Explorer" (In Russian). In *Proceedings of the 7th National Conference on Artificial Intelligence, KII 2000, Russia*, pages 127–134, 2000.