A Finite State Model for On-Line Analytical Processing in Triadic Contexts

Gerd Stumme

Chair of Knowledge & Data Engineering, Department of Mathematics and Computer Science,
University of Kassel, Wilhelmshöher Allee 73, D–34121 Kassel, Germany
http://www.kde.cs.uni-kassel.de

Abstract. About ten years ago, triadic contexts were presented by Lehmann and Wille as an extension of Formal Concept Analysis. However, they have rarely been used up to now, which may be due to the rather complex structure of the resulting diagrams. In this paper, we go one step back and discuss how traditional line diagrams of standard (dyadic) concept lattices can be used for exploring and navigating triadic data.
Our approach is inspired by the *slice & dice* paradigm of On-Line-Analytical Processing (OLAP). We recall the basic ideas of OLAP, and show how they may be transferred to triadic contexts. For modeling the navigation patterns a user might follow, we use the formalisms of finite state machines. In order to present the benefits of our model, we show how it can be used for navigating the IT Baseline Protection Manual of the German Federal Office for Information Security.

1 Introduction

Concept lattices have proven their high potential for visualizing and exploring datasets in many applications during the last 25 years. This success of Formal Concept Analysis incited researchers to extend it to other types of knowledge representation. Among them are for instances logical extensions, relational data, and power context families. One of these extensions are *triadic contexts*, which were introduced ten years ago by Fritz Lehmann and Rudolf Wille in [14]. They defined a *triadic formal context* as a quadruple $\mathbb{K} := (G, M, B, Y)$ where G, M, and B are sets, and Y is a ternary relation between G, M, and B, i. e., $Y \subseteq G \times M \times B$. The elements of G, M, and B are called *(formal) objects*, *attributes*, and *conditions*, resp, and $(g, m, b) \in Y$ is read "object g has attribute m under condition b. A *triadic concept* of $\mathbb{K}$ is a triple (A_1, A_2, A_3) with $A_1 \subseteq G$, $A_2 \subseteq M$, and $A_3 \subseteq B$ where $A_1 \times A_2 \times A_3 \subseteq Y$ such that none of its three components can be enlarged without violating this condition.

Lehmann and Wille present an extension of the theory of ordered sets and (concept) lattices to the triadic case, and discuss structural properties. This approach initiated research on the theory of *concept trilattices*, which was followed by several researchers (e. g., [1–6, 8, 10, 11, 15–18, 20–22]). Already in the first paper on this topic, Lehmann and Wille elaborated also a visualization of concept trilattices in *triadic diagrams*. But even though there are applications where the natural representation of the data are triadic contexts, the visualization by triadic diagrams never made it into practice, and there exist only few visualizations of rather small concept trilattices. This is probably due to

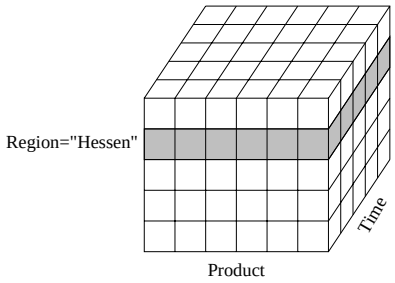

Fig. 1. A data cube

the complex structure of the diagrams. In this paper, we go one step back and discuss how traditional line diagrams of dyadic concept lattices can be used for exploring and navigating triadic data.

The idea of deriving dyadic contexts from the triadic one is not new. Lehmann and Wille present, for instance, in [14] the derived dyadic context $\mathbb{K}^{(1)} := (G, M \times B, Y^{(1)})$ with $(g,(m,b)) \in Y^{(1)} :\iff (g,m,b) \in Y$ (marked by 'W' below), and its two symmetric variations. In [8], the set B is used to define two modal operators (marked by '$\exists$' and '$\forall$' below). We will use these derivation modes later, but will set them in a common navigation framework.

Our approach for navigating triadic data is inspired by the *slice & dice* paradigm of On-Line-Analytical Processing (OLAP). We present the basic ideas of OLAP in the next section, and show how they may be transferred to triadic contexts. For modeling the navigation patterns a user might follow, we use the formalisms of finite state machine (see Section 3). In order to present the benefits of our model, we show in Section 4 how it can be used for navigating the IT Baseline Protection Manual of the German Federal Office for Information Security. As this model is only a first step to a comprehensive navigation environment for triadic (and possibly other) data, many interesting research questions remain open. They conclude the article.

2 On-line Analytical Processing and Triadic Contexts

The expression *On-Line Analytical Processing (OLAP)* has been coined by E. F. Codd et al in [7], and stands for the analysis of multi-dimensional data. We will first give a short introduction in the main features of OLAP as far as they are needed in this paper, before informally outlining how we adapt them to triadic contexts.

OLAP relies on the metaphor of a (high-dimensional) cube containing data. One might for instance want to structure sales facts along the dimensions region, product and time. These dimensions span a three-dimensional cube as shown in Fig. 1. The cube is composed of cells, one for each combination of a region, a product, and a day. The cell contains a numerical value indicating how many items of that product have been purchased in the specific region on the given day.

The analyst may ask queries like 'Give me the sales facts for all products over all days in Hessen' or 'Give me the total number of items sold of product X in Hessen within the whole time period'. Both queries reduce the dimensionality of the answer: the first query returns a two-dimensional answer (the 'slice' which is indicated in Fig. 1), while the second query returns a one-dimensional answer. This reduction of the dimensionality is known as *slicing* in OLAP. The second query applies additionally an *aggregation function* as it sums up numbers of items. There are usually predefined hierarchies on the dimensions along which the aggregation takes place. For instance, days may sum up to months, and months to years. In this paper, however, we won't make use of these hierarchies.[1] An additional feature of OLAP is *dicing*[2] which rotates the data cube. This is particularly useful if the results are presented by spreadsheets, as it allows to permute rows and columns.

The association between OLAP data cubes and triadic contexts is now straight forward. The latter can in fact be considered as an OLAP data cube with three dimensions (G, M, and B), and the content of the cells represents the membership function of the relation Y.

As in OLAP, we want to be able to dice. This means that we want to allow to use any of the three sets as the set of objects at some point in time, depending on the task on hand. Therefore, we will not fix the roles of the three sets in advance. Instead, we consider a triadic context as symmetric structure, where all three sets are of equal importance. The decision which of the sets is considered as object set, attribute set, and condition set, resp., is made later by the user. For easier handling, we will therefore denote the triadic context $\mathbb{K} := (G, M, B, Y)$ alternatively by $\mathbb{K} := (K_1, K_2, K_3, Y)$ in the sequel. We consider all (triadic) contexts in this paper to be finite.

We will usually not work on the full triadic context (K_1, K_2, K_3, Y). Instead, we allow to focus on subsets of interest. Hence, for each of the three dimensions, we allow to restrict the set K_i to any subset X_i we are currently interested in. Thus, the current triadic context is just the sub-context $(X_1, X_2, X_3, Y \cap (X_1 \times X_2 \times X_3))$. This reduction is inspired by the slicing operation in OLAP.

Dicing is modeled by a permutation on the set $\{1, 2, 3\}$, i. e., by an element σ of the full symmetric group S_3. Such a permutation indicates that currently $X_{\sigma(1)}$ is considered as set of objects, $X_{\sigma(2)}$ as set of objects, and $X_{\sigma(3)}$ as set of conditions.

The aggregation mode is determined by one of the four options '$\exists$', '$\forall$', '$\ni$', and '$\not\ni$'.

- In the first case, we consider the concept lattice of the dyadic context

$$\mathbb{K}^{\sigma,\exists}_{X_1,X_2,X_3} := (X_{\sigma(1)}, X_{\sigma(2)}, I)$$

  with $(x_{\sigma(1)}, x_{\sigma(2)}) \in I$ if and only if there exists $x_{\sigma(3)} \in X_{\sigma(3)}$ with $(x_1, x_2, x_3) \in Y$.
- In the second case, we consider the concept lattice of the dyadic context

$$\mathbb{K}^{\sigma,\forall}_{X_1,X_2,X_3} := (X_{\sigma(1)}, X_{\sigma(2)}, I)$$

[1] For a combination of these hierarchies with Formal Concept Analysis, see [19].

[2] Here the terminology is diverging in the literature. In some papers this is called *pivoting*, while 'dicing' is used for 'slicing' with resulting slices of dimension 3 or higher.

with $(x_{\sigma(1)}, x_{\sigma(2)}) \in I$ if and only if for all $x_{\sigma(3)} \in X_{\sigma(3)}$ holds $(x_1, x_2, x_3) \in Y$.
– In the third case, we consider the concept lattice of the dyadic context

$$\mathbb{K}^{\sigma,\mathfrak{S}}_{X_1,X_2,X_3} := (X_{\sigma(1)} \times X_{\sigma(3)}, X_{\sigma(2)}, I)$$

with $((x_{\sigma(1)}, x_{\sigma(3)}), x_{\sigma(2)}) \in I$ if and only if $(x_1, x_2, x_3) \in Y$.
– In the fourth case, we consider the concept lattice of the dyadic context

$$\mathbb{K}^{\sigma,\mathsf{W}}_{X_1,X_2,X_3} := (X_{\sigma(1)}, X_{\sigma(2)} \times X_{\sigma(3)}, I)$$

with $(x_{\sigma(1)}), (x_{\sigma(2)}, x_{\sigma(3)})) \in I$ if and only if $(x_1, x_2, x_3) \in Y$.

Concluding, the binary context (and its concept lattice) that we consider at a given moment depends on the following selections:

– the choice of three subsets $X_1 \subseteq K_1$, $X_2 \subseteq K_2$, and $X_3 \subseteq K_3$,
– a permutation $\sigma \in S_3$, and
– the choice of the aggregation mode $q \in \{\exists, \forall, \mathfrak{S}, \mathsf{W}\}$.

Up to now, we have discussed how single concept lattices can be derived from a triadic context. In order to support navigation, however, we need a mechanism which allows us to come from one concept lattice to the next. This will be discussed in the next section. We make use of the model of a finite state machine. Single concept lattices will correspond to states, while the navigation steps are captured by state transitions.

3 The Finite State Model

As said above, there are many binary contexts that can be derived from a triadic one. In this section, we discuss how the navigation between them may go on. We model this by a finite state machine.

We recall that a finite state machine is a model of computation consisting of a set of states, a start state, an input alphabet, and a transition function that maps pairs of input symbols and current states to a next state. Thus, it is a tuple $A = (E, S, \delta, s_0)$ where E is a finite set, the *input alphabet*, S is a finite set, the *set of states*, δ is the transition function, i. e., a mapping from $E \times S$ to S, and $s_0 \in S$ is the start state. In our approach, the contexts are considered as the states, and the navigation through the set of contexts is modeled by the transition function.

Next, we give the formal definition of our finite state machine. Readers unfamiliar with mathematical notations might first go to Section 4 in order to get a feeling for the approach, before returning here.

As discussed in Section 2, the derivation of a binary context depends on a set of choices. The combination of these choices makes up the state:

Definition 1. *Let $\mathbb{K} := (K_1, K_2, K_3, Y)$ be a triadic context. A state is then a tuple*

$$s := (X_1, X_2, X_3, \sigma, q)$$

where $X_1 \subseteq K_1$, $X_2 \subseteq K_2$, and $X_3 \subseteq K_3$, $\sigma \in S_3$, and $q \in \{\exists, \forall, \mathfrak{S}, \mathsf{W}\}$. The set of all states of a triadic context is denoted by $S(\mathbb{K})$, or simply by S if $\mathbb{K}$ is unambiguous from the context.

For a given state $s := (X_1, X_2, X_3, \sigma, q)$, we let $\mathbb{K}(s) := \mathbb{K}^{\sigma,q}_{X_1,X_2,X_3}$, and $\underline{\mathfrak{B}}(s) := \underline{\mathfrak{B}}(\mathbb{K}(s))$.

We initialize our state machine with the *start state*

$$s_0 := (K_1, K_2, K_3, \mathrm{id}, \exists) \ .$$

This setting allows a first, global overview over the data, which may be refined later. The initial choice of the existential quantifier for q follows the observation that this is the most frequently used in applications for deriving a dyadic context.

The *input alphabet* E of the machine is given by

$$\begin{gathered} E := \{\mathtt{slice}(i, A) \mid i \in \{1, 2, 3\}, A \subseteq K_i\} \\ \cup \{\mathtt{dice}(\sigma) \mid \sigma \in S_3\} \\ \cup \{\mathtt{mode}(\exists), \mathtt{mode}(\forall), \mathtt{mode}(\mathfrak{S}), \mathtt{mode}(\mathsf{W})\} \ . \end{gathered}$$

Last but not least we define the *transition function*. We split up the definition into three parts, corresponding to the three types of elements of the input alphabet listed above.

Definition 2. *Let $s := (X_1, X_2, X_3, \sigma, q)$ be a state, and $e \in E$. The transition function δ is defined as follows. If $e = \mathtt{slice}(i, A)$ with $i \in \{1, 2, 3\}$ and $A \subseteq K_i$, then $\delta(s, e) := (X'_1, X'_2, X'_3, \sigma, q)$ with $X'_i := A$ and $X'_j := X_j$, for $j \neq i$.*

For simplifying the navigation, one could restrict the set A to be a concept extent or intent of some suitable concept lattice. Experiments with the example discussed below, however, revealed the need for selecting arbitrary sets. Note that we also allow to enlarge sets again, as there is no constraint saying that A has to be a subset of X_i. This allows to extend sets again during the navigation process. In practice however, A often is a subset of X_i. We discuss some properties of this case before continuing the definition of the transition function.

Let $A \subseteq X_i$. If $\sigma(1) = i$ or $q = \mathfrak{S}$ and $\sigma(3) = i$, then the `slice` operation reduces the current set of objects. The resulting concept lattice is thus isomorphic to a $\vee$-sub-semilattice of the previous one. If $\sigma(2) = i$ or $q = \mathsf{W}$ and $\sigma(3) = i$, then the `slice` operation reduces the current set of attributes, and the resulting concept lattice is isomorphic to a $\wedge$-sub-semilattice of the previous one. In all these cases, the information presented in the lattice is thus reduced, just as a slicing operation in OLAP would do. In the two remaining cases, however, the analogy to OLAP fails. If $\sigma(3) = i$ and $q = \exists$, then the binary relation of the current dyadic context decreases; if $q = \forall$ then the binary relation increases. Simple examples show that there is no pre-determined relationship between the current and the following lattice. In both situations the concept lattice can either shrink or grow, depending on the constellation.

Definition 2 (contd.). *If $e = \mathtt{dice}(\sigma')$ with $\sigma' \in S_3$, then $\delta(s, e) := (X_1, X_2, X_3, \sigma \circ \sigma', q)$.*

We may denote the elements $\sigma \in S_3$ by $(123), (132), \ldots, (321)$ where, e. g., (132) means that the role of the object set remains unchanged while the attribute and the condition sets interchange their roles. The transition $\texttt{dice}(1,2,3)$ doesn't do anything; and the transition $\texttt{dice}(2,1,3)$ interchanges the roles of objects and attributes. This means that the concept lattice is turned upside down.

Definition 2 (contd.). *If* $e = \texttt{mode}(q)$ *with* $q \in \{\exists, \forall, \texttt{Ↄ}, \texttt{W}\}$ *then* $\delta(s,e) := (X_1, X_2, X_3, \sigma, q)$.

In practice it turns out that the '$\exists$' mode is the mostly used one. The change from either '$\exists$' or '$\forall$' to '$\texttt{Ↄ}$' or '$\texttt{W}$' can be considered as drill-down, since the resulting concept lattice provides more detailed information. A change of mode in the other direction is a roll-up, as the information becomes more summarized.

The definition of our finite state machine is now complete. Next we introduce two additional shortcuts.

Definition 3. *We let* $\texttt{delG} := \texttt{slice}(\sigma^{-1}(1), \{\{x\} \in X_{\sigma^{-1}(1)} \mid \{x\}' \neq \emptyset\}$ *and* $\texttt{delM} := \texttt{slice}(\sigma^{-1}(2), \{x \in X_{\sigma^{-1}(2)} \mid x' \neq \emptyset\}$, *where the derivation* $\cdot'$ *is computed in the current dyadic context.*

These two derived operators serve the following purpose. After a `slice` operation, one usually also wants to prune the remaining sets to the relevant elements. For instance, if one reduced the set of objects, then there may be some attributes which do not relate to any of the remaining objects. In most cases, one may want to remove these attributes (which would all be attached to the bottom concept), as they do not provide any further insight. This removing of 'superfluous' attributes is performed by `delM`, while `delG` removes all objects which are not covered by at least one attribute any more.

As known from basics about finite state machines, we can now extend the transition function such that it applies not only to single symbols of the input alphabet, but also to words. We denote the set of words over the input alphabet E by E^*, which includes the empty word λ. The transition function δ is then recursively extended to $\delta^*: E^* \times S \to S$ by $\delta^*(\lambda, s) := s$, and $\delta^*((w,e),s) := \delta^*(w, \delta(e,s))$, for $w \in E^*$ and $e \in E$.

An element w of E^* can naturally be considered as a program (which is executed from right to left). Its semantics is given by the function $[\![w]\!]: S \to S$ which is given by $[\![w]\!](s) := \delta^*(w,s)$. Specifically, one can determine the current state of the system by storing all previous interactions of the user as $w \in E^*$. The current state is then just $[\![w]\!](s_0)$. In an implementation of the framework, w may be shown as *navigation history* to the user, and supports an 'undo' function or 'back' button.

4 Navigation within the Triadic Information System

In this section, we show by an example, how our framework supports navigation in a real world dataset. The IT Baseline Security Manual [9] of the German Federal Office for Information Security provides a description of the threat scenario that is globally assumed, standard security measures for typical IT systems, and detailed descriptions of safeguards to assist with their implementation.[3]

[3] The online version of the manual can be found at http://www.bsi.de/gshb/ .

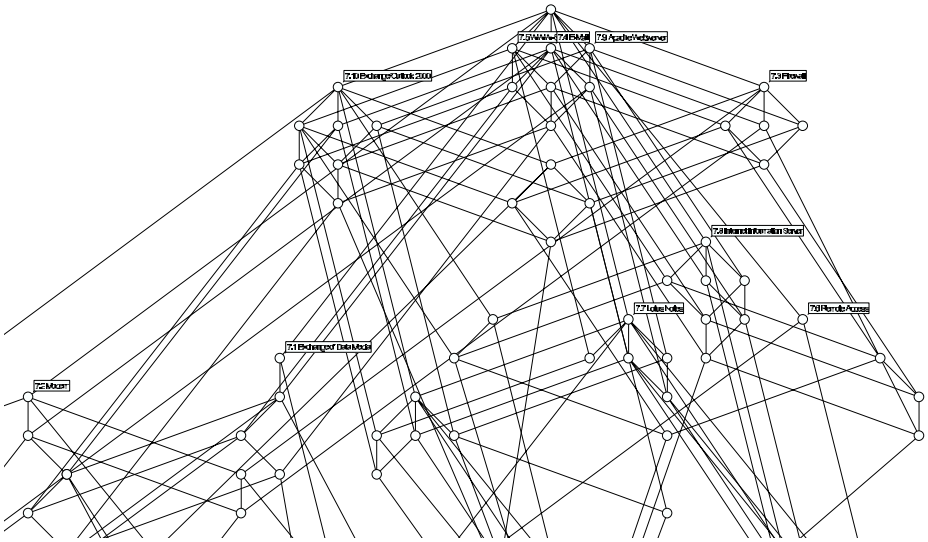

Fig. 2. The top part of the concept lattice of the initial state

The core data of the manual can be considered as a triadic context. We consider the the possible threats as objects, the IT components as attributes, and the safeguards as conditions.[4] For presentation purposes, we restrict the example as follows. K_1 :={Exchange of Data Media, ..., Exchange/Outlook 2000} is the set of all ten data transmission systems listed in Section 7 of [9], K_2 is the set of all threats against at least one of the data transmission systems, and K_3 is set of all safeguards against at least one of these threats for at least one of the data transmission systems.[5] The top part of the concept lattice of the start state is shown in Fig. 2. The names of the threats (which play the role of objects at the moment) are omitted due to representation issues. From this initial state, we will perform a series of navigation steps to illustrate the different features of the model.

First we want to reduce the components to those which are currently used at our research group. We perform thus the operations

op_1: `slice`(1, {7.3 Firewall, 7.4 E-Mail, 7.5 WWW-Server, 7.6 Remote Access, 7.9 Apache Webserver})

op_2: `delG` .

The resulting concept lattice is shown in Fig. 3. The concept in the middle of the diagram indicates for instance that the two threats 'T 3.38 Errors in configuration and

[4] Other assignments have been done in [8, 18, 22]. But as our approach considers all sets equivalently, this assignment influences the start state only. Any initial arrangement can be reached from any other by one `dice` operation.

[5] This restricted scenario can also be reached by three consecutive `slice` operations within the larger system that comprises all components, threads and safeguards discussed in the manual.

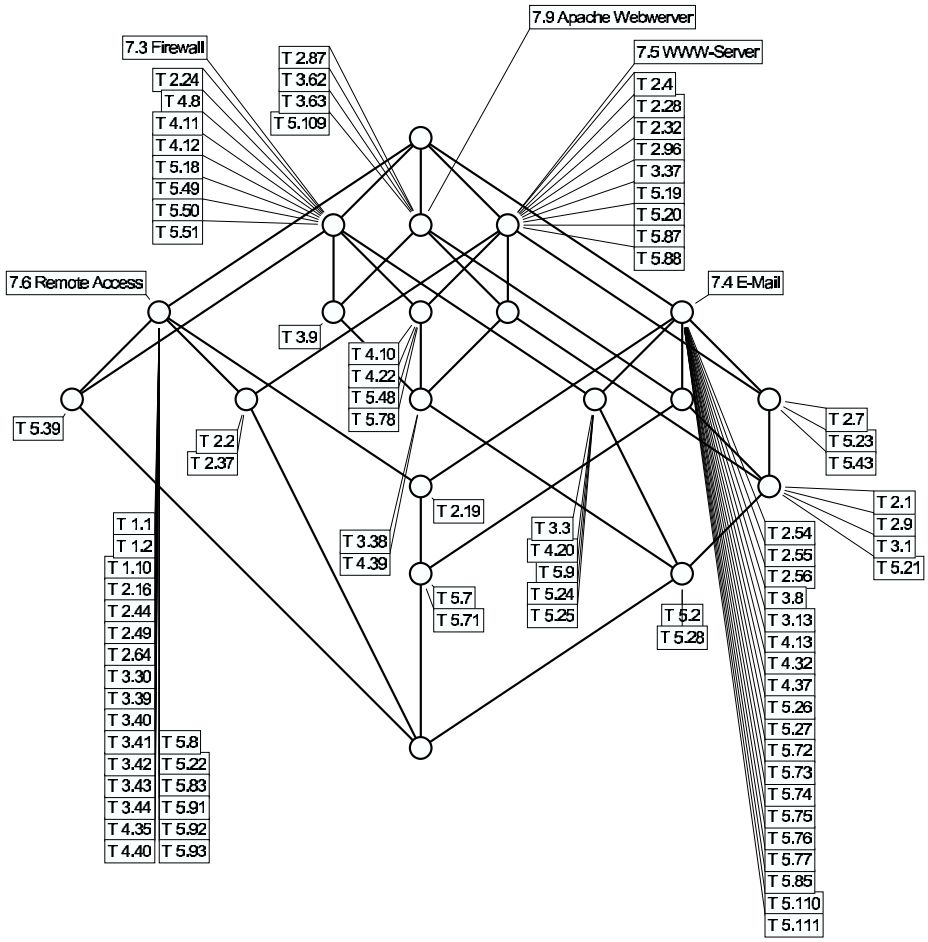

Fig. 3. After slicing the set of components to Firewall, E-Mail, WWW-Server, Remote Access, Apache Webserver

operation' and 'T 4.39 Software conception errors' are the threats which are directed against all the three components Firewall, WWW-Server, and Apache Webserver. The two threats 'T 5.2 Manipulation of data and software' and 'T 5.28 Denial of services' are directed against these components, but they additionally concern the E-Mail system, as the lower right concept indicates. The complete list of threats can be found at the website mentioned above.

A major set of threats are deliberate attacks to the system. We now want to study which of these attacks are potentially dangerous to the data transmission systems of our research group. We perform thus the operation

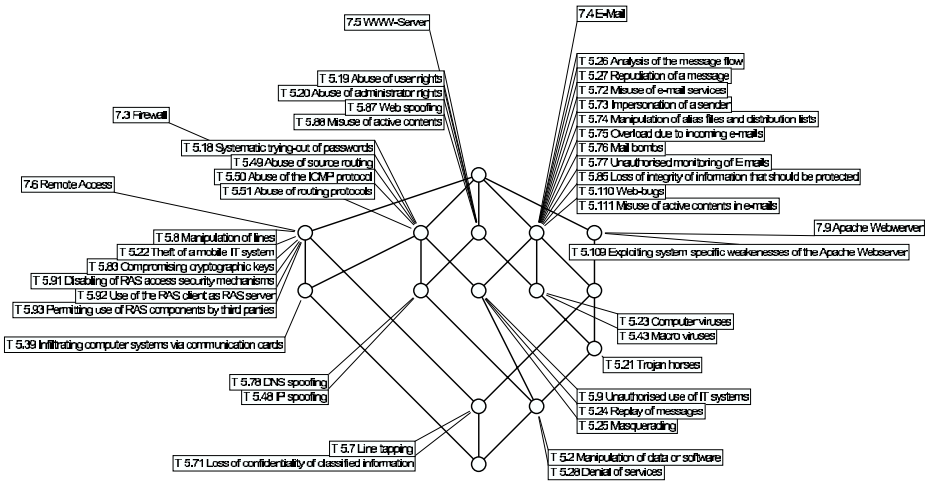

Fig. 4. After slicing the set of threats to the deliberate acts

op_3: `slice`(2, $X_2 \cap \{$'T 5.1 Manipulation or destruction of IT equipment or accessories',..., 'T 5.111 Misuse of active content of E-Mails'$\}$)

(where X_2 is the current set of objects) and obtain the line diagram in Fig. 4. The diagram shows that there are many rather specific threats, as they are related to only one component each.

In order to analyze in more depth deliberate acts against combinations of components, we prune away all deliberate acts against single components. The operation

$$op_4\colon \texttt{slice}(2, \{x \in \tilde{X}_2 | \text{card}(\{x\}') \neq 1\})$$

(where $\tilde{X}_2$ is the current set of objects) yields the concept lattice in Fig. 5. In the lattice, we can for instance discover that there are four threats which endanger at the same time firewalls and WWW servers: IP spoofing, DNS spoofing, manipulation of data or software, and denial of services.

For studying which other components are threatened by denial of service attacks, we could make use of the same diagram. Another option — which supports the more natural way of reading from top to bottom — is to turn the diagram upside down first. As well-known in FCA, this is obtained by interchanging the roles of objects and attributes on the context level:

$$op_5\colon \texttt{dice}(2, 1, 3)$$

The result of the operation is shown in Fig. 6. The components threatened by denial of service attacks are now exactly those which are listed below 'T 5.28 Denial of services' in the diagram. We see that not only firewalls and WWW servers are endangered by this threat, but also the Apache web server and email systems.

What can now be done in order to protect the data transmission systems of our research group against deliberate acts? This question can be approached by focusing on

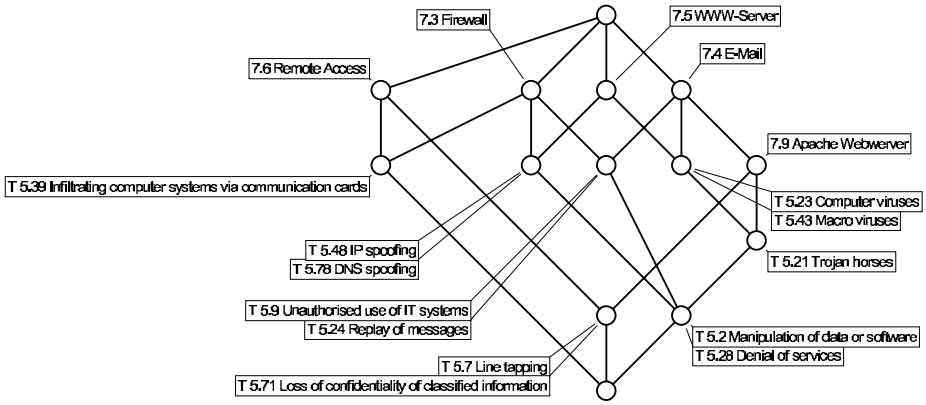

Fig. 5. After slicing away all threats which are against single components

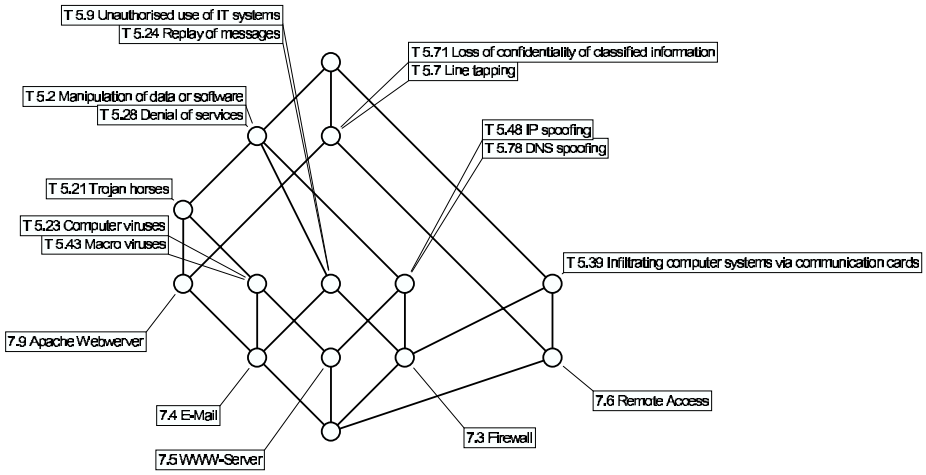

Fig. 6. After dicing (exchange of objects and attributes)

the relation between threats and safeguards rather than between threats and components. In other words, we have to interchange the role of components and safeguards:

$$op_6\colon \texttt{dice}(3, 2, 1)$$

The result is shown in Fig. 6. The encoding of the safeguards can be found online in [9]. The lattice is rather complex — which indicates that there is no easy solution for protecting our IT environment.

For getting a better insight, we focus (first) on those safeguards which are related to hard- and software:

$$op_7\colon \texttt{slice}(3, \{x \in \tilde{X}_3 | x = \text{“S 4. } \ldots\text{”}\})$$

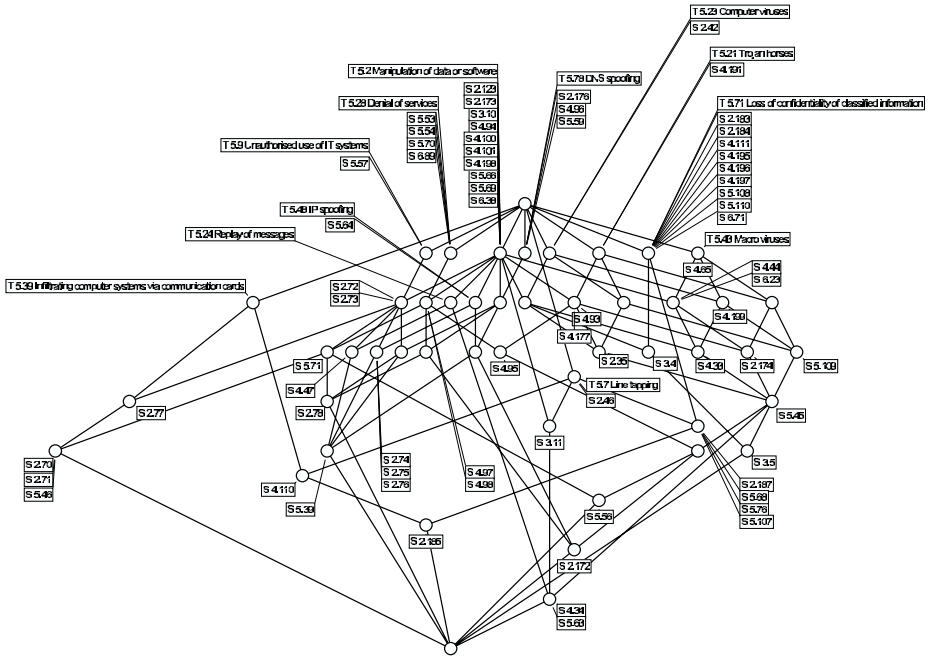

Fig. 7. After dicing (exchange of objects and conditions) again

(where $\tilde{X}_3$ is the current set of objects). This yields the concept lattice in Fig. 8. It shows for instance that there is no hard- or software related safeguard against IP spoofing. This threat has to be countered by other means. On the other side we discover that even with only two safeguards, 'S 4.95 Minimal operating system' and 'S 4.34 Using encryption, checksums or digital signatures', many of the listed threats can be countered.

Figure 8 doesn't show in detail if a safeguard is designed against a threat for all relevant components or just for specific ones. One could expect that the choice of a safeguard is independent of the component. By drill-down, we can analyze this hypothesis. Figure 9 shows the result of

$$op_8 \colon \mathtt{mode}(\Im) \ ,$$

i. e., the concept lattice

$$\underline{\mathfrak{B}}([\![op_8, \ldots, op_1]\!](s_0)) \ .$$

By comparing Figs. 8 and 9, we discover that this hypothesis is indeed almost true, since the concept lattices are quite similar. However, there are some differences. For instance, safeguard 'S 4.33 Use of a virus scanning program while exchanging of data media and data transmission' is adequate against macro viruses and computer viruses for email systems, while it protects web servers against computer viruses and manipulation of data or software. At least, this is what the data provided by the IT baseline manual indicate. One may now discuss if this is adequately modeling the situation. We,

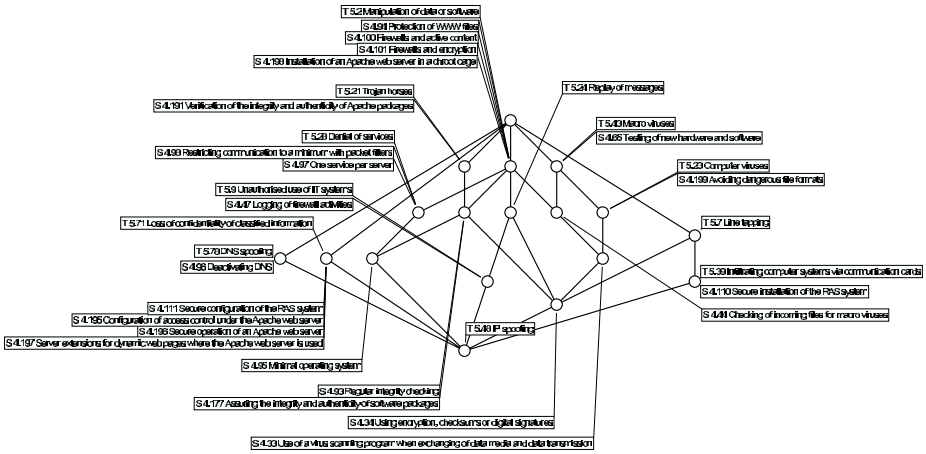

Fig. 8. After slicing the safeguards to the hard- and software related ones

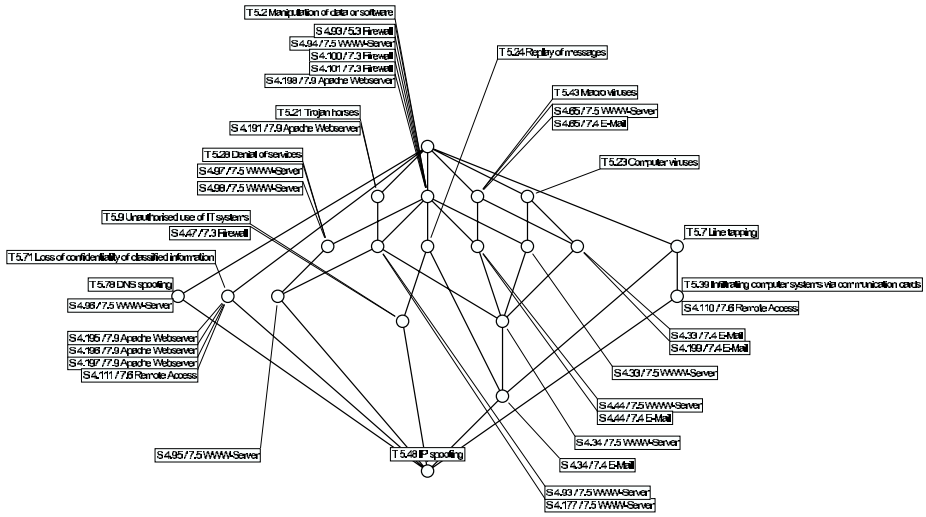

Fig. 9. After drill-down to the components

however, will now leave this example, and return to a more general discussion on future research.

5 Conclusion and Outlook

In this paper, we integrated bits and pieces which were available for analyzing triadic contexts by line diagrams of derived dyadic contexts, and set them into a navigation

framework. As this is only a first step, many interesting research questions remain open. They include:

- Further development of automated drawing routines is important, as there are so many potential concept lattices which can be derived from a triadic context that one cannot create them all beforehand. (In this paper, we have made the layout manually).
- The framework should be implemented and evaluated in more scenarios. This includes a concretization of the means of interaction by which the user can perform the slice & dice operations.
- How can techniques like conceptual scaling or iceberg lattices be exploited for more sophisticated navigation means?
- Are there other aggregation modes? For instance, one might want to integrate power scales [13] or relational scales [12].
- In OLAP, the dimensions carry an additional structure: Each dimension goes along with a hierarchy along which aggregation is performed. On a small scale, such hierarchies also exist in our IT example, by means of clustering each of the sets of components, threats, and safeguards into different sections of the IT Baseline Manual. How can these hierarchies be incorporated into the model?
- Relational databases are an important application domain which provides a lot of interesting applications. From the perspective of Formal Concept Analysis, many-valued contexts and multi-contexts are data structures closely related to relational databases. How can the framework presented here be extended to incorporate them as well?

References

1. K. Biedermann. How triadic diagrams represent conceptual structures. In D. Lukose, H. S. Delugach, M. Keeler, L. Searle, and J. F. Sowa, editors, *Conceptual Structures: Fulfilling Peirce's Dream*, number 1257 in LNAI, pages 304–317, Heidelberg, 1997. Springer.
2. K. Biedermann. Triadic Galois connections. In K. Denecke and O. Lüders, editors, *General algebra and applications in discrete mathematics*, pages 23–33, Aachen, 1997. Shaker Verlag.
3. K. Biedermann. Completion of triordered sets and trilattices. In D. Dorninger, G. Eigenthaler, H.K. Kaiser, H. Kautschitsch, W. More, and W.B. Müller, editors, *Contributions to General Algebra*, number 10, pages 61–78, Klagenfurt, 1998. Johannes Heyn Verlag.
4. K. Biedermann. *A foundation of the theory of trilattices.* Dissertation, TU Darmstadt, Aachen, 1998.
5. K. Biedermann. Powerset trilattices. In M. Mugnier and M. Chein, editors, *Conceptual Structures: Theory, Tools and Applications*, volume 1453 of *Lecture Notes in Computer Science*. Springer, 1998.
6. K. Biedermann. An equational theory for trilattices. *Algebra Universalis*, 42:253–268, 1999.
7. E. F. Codd, S. B. Codd, and C. T. Salley. Providing OLAP (On-Line Analytical Processing) to User-Analysis: An IT Mandate, 1993. White paper.
8. F. Dau and R. Wille. On the modal unterstanding of triadic contexts. In R. Decker and W. Gaul, editors, *Classification and Information Processing at the Turn of the Millenium*, Proc. Gesellschaft für Klassifikation, 2001.

9. German Federal Office for Information Security. IT Baseline Protection Manual. http://www.bsi.de/gshb/, October 2003.
10. B. Ganter and S. A. Obiedkov. Implications in triadic contexts. In *Conceptual Structures at Work: 12th International Conference on Conceptual Structures*, volume 3127 of *Lecture Notes in Computer Science*. Springer, 2004.
11. B. Groh and R. Wille. Lattices of triadic concept graphs. In B. Ganter and G. W. Mineau, editors, *Conceptual Structures: Logical, Linguistic, and Computational Issues*, volume 1867 of *Lecture Notes in Computer Science*. Springer, 2000.
12. J. Hereth. Relational scaling and databases. In U. Priss, D. Corbett, and G. Angelova, editors, *Conceptual Structures: Integration and Interfaces, 10th International Conference on Conceptual Structures, ICCS 2002, Borovets, Bulgaria, July 15-19, 2002, Proceedings*, volume 2393 of *Lecture Notes in Computer Science*, pages 62–76. Springer, 2002.
13. J. Hereth and G. Stumme. Reverse pivoting in conceptual information systems. In H. S. Delugach and G. Stumme, editors, *Conceptual Structures: Broadening the Base*, volume 2120 of *Lecture Notes in Computer Science*, pages 202–215. Springer, 2001.
14. F. Lehmann and R. Wille. A triadic approach to formal concept analysis. In G. Ellis, R. Levinson, W. Rich, and J. F. Sowa, editors, *Conceptual Structures: Applications, Implementation and Theory*, volume 954 of *Lecture Notes in Computer Science*. Springer, 1995.
15. S. Prediger. Nested concept graphs and triadic power context families. In B. Ganter and G. W. Mineau, editors, *Conceptual Structures: Logical, Linguistic, and Computational Issues. Proc. ICCS '00*, number 1867 in LNAI, pages 249–262, Heidelberg, 2000. Springer.
16. U. Priss. A triadic model of information flow. In G.W. Mineau, editor, *Conceptual Structures: Extracting and Representing Semantics*, pages 159–170, Quebec, Canada, 2001. Dept. of Computer Science, University Laval.
17. L. Schoolmann and R. Wille. Concept graphs with subdivision: A semantic approach. In Aldo de Moor, Wilfried Lex, and Bernhard Ganter, editors, *Conceptual Structures for Knowledge Creation and Communication*, volume 2746 of *Lecture Notes in Computer Science*, pages 271–281. Springer, 2003.
18. H. Söll. Begriffliche Analyse triadischer Daten: Das IT-Grundschutzhandbuch des Bundesamts für Sicherheit in der Informationstechnik. Diploma thesis, FB Mathematik, TU Darmstadt, Darmstadt, April 1998.
19. G. Stumme. On-line analytical processing with conceptual information systems. In K. Tanaka and S. Ghandeharizadeh, editors, *Proc. 5th Intl. Conf. on Foundations of Data Organization (FODO'98)*, pages 117–126, November 12-13, 1998.
20. R. Wille. The basic theorem of triadic concept analysis. *Order*, 12:149–158, 1995.
21. R. Wille. Triadic Concept Graphs. In M.-L. Mugnier and M. Chein, editors, *Conceptual structures: theory, tools and application*, number 1453 in LNAI, pages 194–208, Berlin-Heidelberg-New York, 1998. Springer.
22. R. Wille and M. Zickwolff. Grundlagen einer triadischen Begriffsanalyse. In G. Stumme and R. Wille, editors, *Begriffliche Wissensverarbeitung. Methoden und Anwendungen*, pages 125–150, Berlin-Heidelberg, 2000. Springer-Verlag.