Computing in Conceptual Data Systems with Relational Structures

Gerd Stumme[1] and Karl Erich Wolff[2]

[1] Technische Hochschule Darmstadt, Fachbereich Mathematik, Schloßgartenstr. 7, D–64289 Darmstadt; stumme@mathematik.th-darmstadt.de
[2] Fachhochschule Darmstadt, Fachbereich Mathematik und Naturwissenschaften, Schöfferstr. 3, D–64295 Darmstadt; wolff@mathematik.th-darmstadt.de

Abstract. While most data processing tools use numerical aspects of the data, conceptual data systems focus on their conceptual structure. This paper discusses how both approaches can be combined.

1 Introduction

Data tables are usually equipped with different types of structures. While most data processing tools use their numerical structure, conceptual data systems are designed for conceptually structuring data. They enrich data tables with so-called conceptual scales reflecting different conceptual aspects of the data ([10], [7]). Conceptual scales provide graphical representations of the conceptual landscape in the form of line diagrams. These "maps" can be explored with the management system TOSCANA ([4], [11]). This navigation tool allows dynamic browsing through and zooming into the data. Conceptual data systems are based on the mathematical theory of formal concept analysis. For an introduction to formal concept analysis, the reader is referred to [1], [13], [14], and [15].

Many applications indicate the need for combining both computational and conceptual structures for data analysis. This discussion can be seen in a broader framework: the aim is to extend conceptual data systems to conceptual knowledge systems which provide, beside knowledge representation and communication techniques, also techniques for knowledge acquisition and knowledge inference. This requires not only a conceptual and a computational component, but also a logical one. First efforts have been made (cf. [5], [8]).

In this paper, we discuss how to aggregate computational and conceptual aspects in conceptual data systems. A financial controlling system for private households serves as a first example.

2 The Conceptual Aspect of the Controlling System

The basic, controlling example underlying this paper consists of a table of all withdrawals from a private bank account during several months. A small part of this table is shown in Fig. 1. As a reading example, the row numbered 20 contains information about a withdrawal of 641.26 DM for office chairs for the

no.	value	paid for	objective	health	date
3	42.00	Konni	ski-club	s	03.01.1995
20	641.26	Konni, Florian	office chairs	/	23.01.1995
27	68.57	Family	health insurance	hi	02.02.1995
34	688.85	Tobias	office table	/	06.02.1995
37	25.00	Father	gymn. club	s	08.02.1995
52	75.00	Konni	gymn. club	s	24.02.1995
73	578.60	Mother	Dr. Schmidt	d	10.03.1995
77	45.02	Tobias	Dr. Gram	d	17.03.1995
80	77.34	Parents	money due	/	21.03.1995

Fig. 1. Withdrawals from a private bank account

sons Konni and Florian paid on January 23, 1995. In the following, we classify these withdrawals with respect to the question "How much has been paid for health for which family members?" To classify the withdrawals with respect to health, we introduce the attribute values "sport" (s), "doctor" (d), "health insurance" (hi), and "/" for withdrawals not assigned to health. The concept lattice in Fig. 2 is our formalization of the conceptual structure of these values. Concept lattices like this, which are used for scaling attribute values, represent so-called *conceptual scales*.

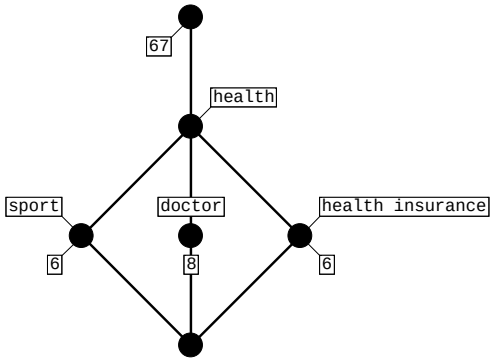

Fig. 2. Frequencies of withdrawals related to "health".

This line diagram represents the classification of the concept "health" into three subconcepts. The set of all withdrawals which are classified by an entry "s" in the column "health" is understood as assigned to "sport", and therefore also to "health". The withdrawals classified by "d" and "hi" are treated analogously.

All withdrawals which are not assigned to "health" are collected just below the highest point.

In the following, we distinguish, for each formal concept $\mathfrak{c}$, between its *extent* (i. e., the set of all objects (here withdrawals) belonging to $\mathfrak{c}$) and its *contingent* (i. e., the set of all objects belonging to $\mathfrak{c}$ but not to any proper subconcept of $\mathfrak{c}$). In the standard line diagram, the contingent of a formal concept $\mathfrak{c}$ is the set of objects which is represented just below the point representing $\mathfrak{c}$. The line diagram in Fig. 2 does not show the contingents explicitly, but only their cardinalities. For instance, the contingent of the highest concept in Fig. 2 is the set of all 67 withdrawals indicated in the data table by "/" in column "health".

The cardinality of a contingent S of a concept $\mathfrak{c}$ is also called the *frequency* of S. In TOSCANA, the user can switch, for each concept, by mouse click between the display of the frequency and the explicit list of objects.

The extent of the highest concept is always the set of all objects. The extent of an arbitrary concept is exactly the union of all contingents of its subconcepts. TOSCANA provides the possibility of displaying either contingents or extents.

As a reading example we mention that there are exactly eight withdrawals in the given data table which are classified under "doctor", formally described in the column "health" by "d". Finally we remark that there are no withdrawals which are assigned to "health" but neither to "sport", "doctor" or "health insurance".

The line diagram in Fig. 2 is unsatisfactory insofar as we would like to see not only the frequencies of withdrawals but the amount of money paid. In the next section we shall discuss how this can be visualized.

3 The Computational Aspect of the Controlling System

For an efficient controlling of the household, the user needs an overview over the distribution of the money, and not of the number of withdrawals. Hence, for each contingent S, we display the sum over the corresponding entries in the column "value" instead of the frequency of S. The result of this computation is the left line diagram in Fig. 3. We can see for example that the withdrawals for "sport" sum up to 383 DM and the withdrawals not concerning "health" sum up to 41538 DM. The right line diagram shows, for each formal concept, the sum over the values of all withdrawals in the extent (instead of the contingent) of this concept, for instance the total amount of 45518 DM for all withdrawals in the given data table and the amount of 3980 DM for "health".

To visualize also the distribution of withdrawals over the family, we use the scale "family" in Fig. 4. This diagram shows for instance that there are withdrawals of 2445 DM for "Mother", that 78 DM are classified under "Parents", but not under "Mother" or "Father" (this is the "money due" in the last row of Fig. 1) and that 38213 DM appear for withdrawals classified under "Family" which are not specified further.

Next we combine the scales "family" and "health". The resulting *nested line diagram* is shown in Fig. 5. Now the withdrawals of each family member (and more generally of each object concept of the family scale) are classified with

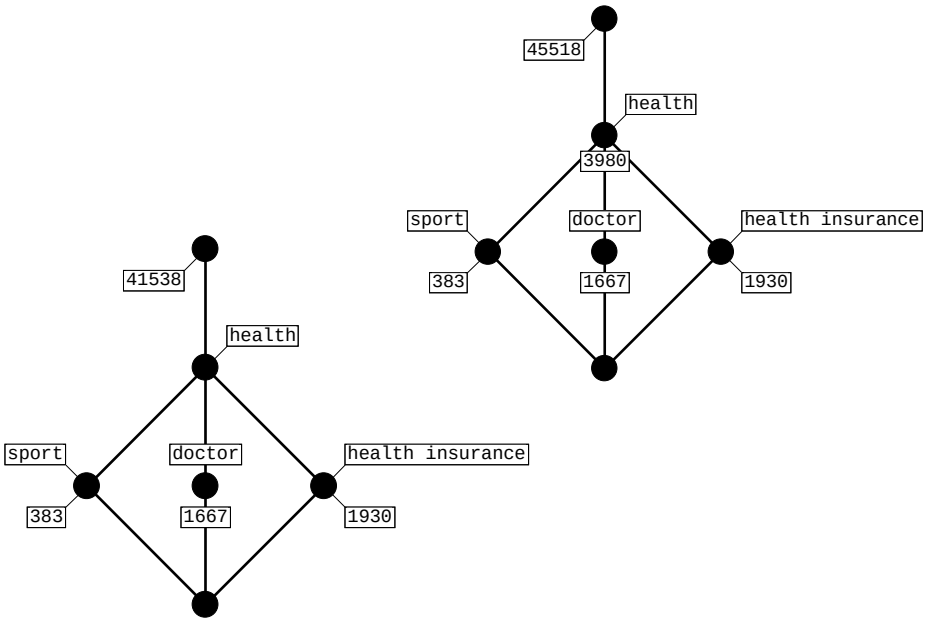

Fig. 3. Summing up book-values over contingents (left) and extents.

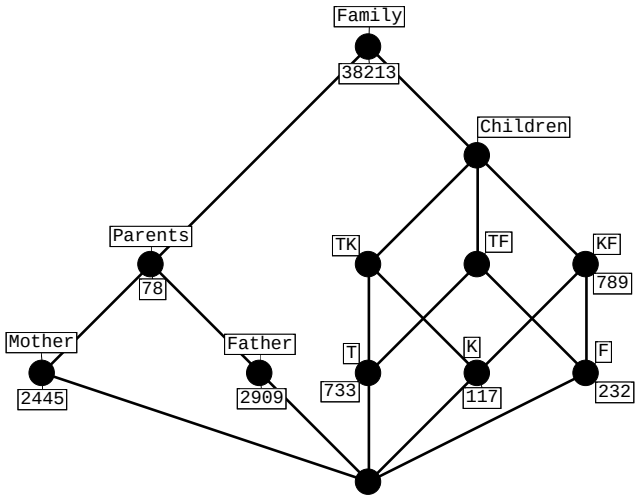

Fig. 4. Summing up book-values over contingents of the scale "family".

respect to the health scale. For instance, 733 DM expended for Tobias split into 45 DM for a doctor and 688 DM not concerning health, i.e., the amount

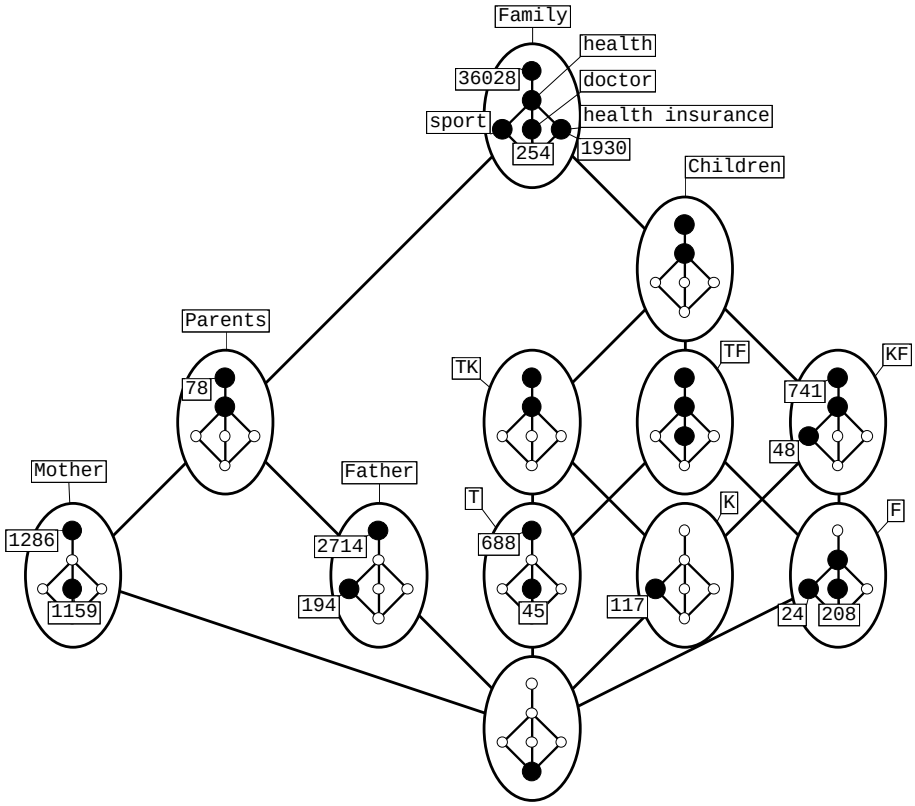

Fig. 5. Summing up over contingents in the nested line diagram of the scales "health" and "family".

spent on his office table (see Fig. 1). This nested line diagram shows also that the withdrawals for health insurance (which amount to 1930 DM) are all summarized under the concept "Family" and are not specified further.

In the next section, we describe the formalization of conceptual and computational structures. This is the basis for generalizing the example in Sect. 5.

4 Conceptual and Computational Structures

In this section, we introduce the mathematical definition of conceptual data systems, and discuss how it can be extended by means of utilizing its algebraic properties.

In formal concept analysis, data tables like the one in Fig. 1 are formalized as a *many-valued context* $\mathbb{K} := (G, M, (W_m)_{m \in M}, I)$. G, M, and W_m, $m \in M$, are

sets, and $I \subseteq \{(g, m, w) \mid g{\in}G, m{\in}M, w{\in}W_m\}$ is a relation where $(g, m, w_1) \in I$ and $(g, m, w_2) \in I$ implies $w_1{=}w_2$. Thus, each $m \in M$ can be seen as a partial function. For $(g, m, w) \in I$ we say that "object g has value w for attribute m" and write $m(g) = w$. The set G of objects can be understood as a primary key in a relational data base. For instance, in Fig. 1, the objects are withdrawal numbers, the attributes are "value", ..., "date", and the attribute values are all possible entries in the corresponding columns.

4.1 Conceptual Scales

The conceptual structure of the data is captured by conceptual scales. Formally, a *conceptual scale* for an attribute $m \in M$ is a formal context $\mathbb{S}_m := (W_m, M_m, I_m)$ where the objects are the possible attribute values in W_m. It serves for "translating" the entries in the many-valued context to aspects the user is interested in. All conceptual scales for a many-valued context are stored in a *conceptual scheme*, together with line diagrams for their concept lattices.

An example is the scale $\mathbb{S}_{\text{health}}$, which was used to generate the diagrams in Figs. 2 and 3. Its concept lattice is shown in Fig. 6. For technical reasons, the set $W_{\text{health}} = \{\text{s}, \text{d}, \text{hi}, /\}$ is replaced by corresponding parts of SQL-queries of the form `health=`"..." . The set M_{health} contains the attributes "health", "sport", "doctor", and "health insurance", and the relation I_{health} indicates their relationships to the SQL-statements.

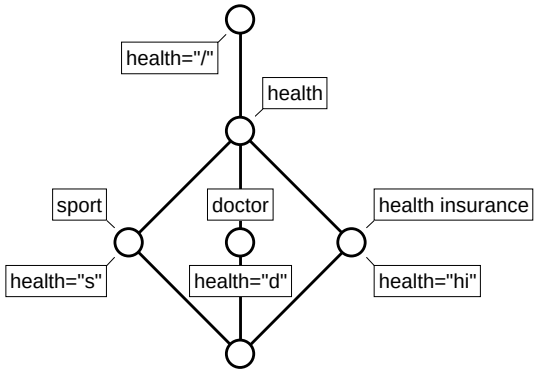

Fig. 6. The scale "health"

In TOSCANA, the user can choose conceptual scales from a menu. Then TOSCANA draws the corresponding (nested) line diagram on the screen. The data base is queried by SQL-statements that are composed out of the statements in the scales. Finally, the results are displayed in the line diagram.

4.2 Relational Structures

In the controlling example, we use the fact that bankbook values are numbers, for which addition is defined. In general, for each $m \in M$, there may be functions and relations on the set W_m.

Definition. A *relational structure* $\mathbb{R} := (W, \mathcal{R}, \mathcal{F})$ consists of a set W, a set $\mathcal{R}$ of relations $R \subseteq W^{ar(R)}$ on W, and a set $\mathcal{F}$ of functions $f: W^{ar(f)} \to W$, where ar assigns to each relation and function its arity.

For instance, the data types implemented in the data base management system (e. g., `Integer`, `Real`, `Boolean`, `Currency`, or `Datetime`) are relational structures.

Hence, for each attribute $m \in M$, we can capture the algebraic structure of its possible attribute values by a relational structure $\mathbb{R}_m := (W_m, \mathcal{R}_m, \mathcal{F}_m)$, just as we captured their hierarchical relationships by a conceptual scale $\mathbb{S}_m := (W_m, M_m, I_m)$. *Conceptional-relational schemes* – as extensions of conceptual schemes – may contain, beside a many-valued context and conceptual scales (together with their line diagrams), relational structures for each attribute.

Here we should mention, that sometimes conceptual and relational aspects overlap. Depending on the purpose, they should be covered by a relational structure or by a conceptual scale, or by both. Time, for instance, can be captured by a linear order in a relational structure, or by some scale (e. g., an inter-ordinal scale, if only certain time intervals are of interest).

Relational structures can be used for creating new scales. This *logical scaling* was developed by S. Prediger (cf. [6]). In this paper, however, we discuss only how relational structures may affect the data analysis process once the conceptual scales are created.

5 Conceptual Scaling Supported by Computational Structures

The controlling example and many other applications show that it is useful not to analyze computational and conceptual aspects of the data independently, but to combine them. In this section, we discuss how conceptual data systems can be extended by a computational component. Since the operability required differs from application to application, the idea is to delegate application-specific computations to an external system (e. g., book-keeping system, CAD system, control system, etc.). TOSCANA already provides an SQL-interface to the relational database management system in which the many-valued context is stored, so that we can use the numerical tools of the relational data base system (as, for instance, in the controlling example).

In the process of going from the request of the user to the diagram shown on the screen, we can distinguish two consecutive, intermediary subprocesses (cf. Fig. 7). First, the chosen scale is imported from the conceptual scheme, and a set of objects is assigned to each of its concepts (by default, its extent or contingent). Second, for each of these sets, some algebraic operations may be performed. Most

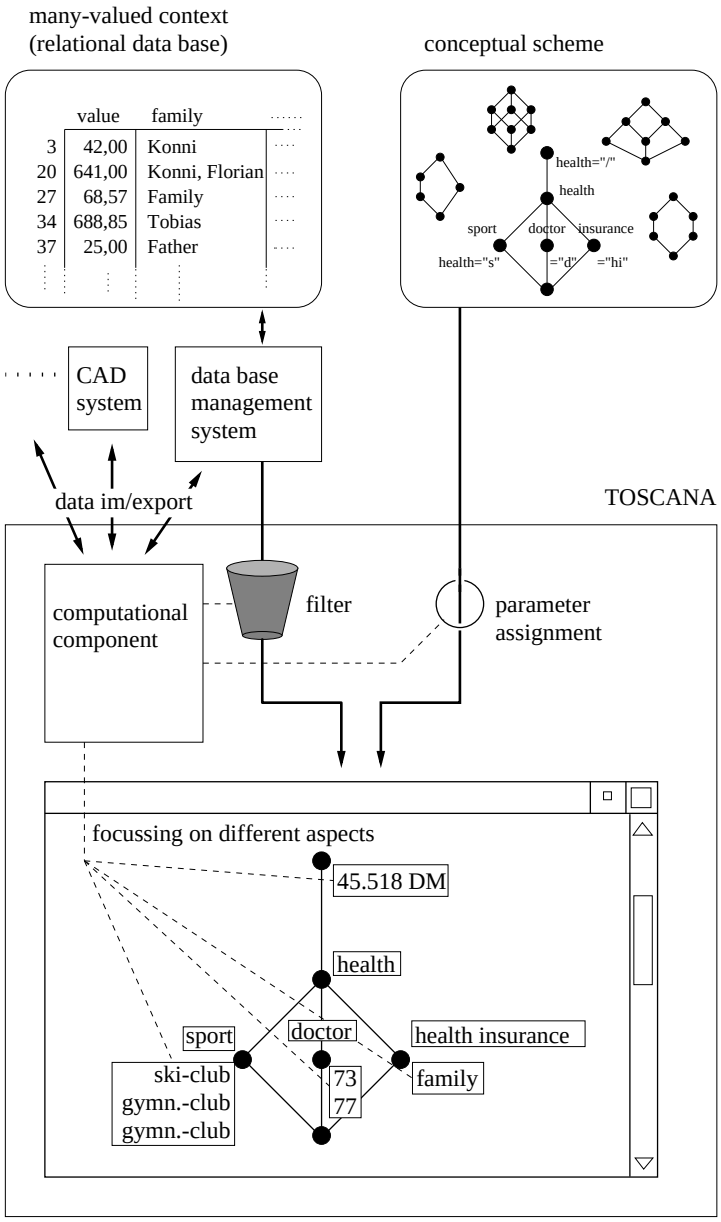

Fig. 7. Interaction of conceptual and computational component in TOSCANA

of the implemented conceptual data systems only activate the first step. Our controlling system is an example where the second step is also activated. In the

first step, we also can identify two actions where a computational component can influence the analysis or retrieval process: the import of scales from the conceptual scheme, where parameters can be assigned to parametrized scales, and the import of objects from the data base, which can be sorted out by filters. Finally, we can imagine a further action, following the display of the line diagram, which results in highlighting interesting concepts.

These four activities which make an interaction between conceptual and computational component possible now shall be discussed in detail.

5.1 Adapting Conceptual Scales to the Data

In general, conceptual scales do not depend on the actual data stored in the data base, as for instance the scales "health" and "family" in the controlling example. Sometimes however, scales should adapt themselves automatically to the data. Therefore we allow parameters in the definition of conceptual scales whose values are determined only at runtime. The way of determining values may vary from application to application. We shall discuss this by three examples.

The first example is implemented in a commercial application where the user wants to analyze data for the last 23 workdays. (This is the maximal number of workdays per month in Germany.) The result is displayed as a nominal scale. When the scale is invoked, then the data base is queried for the last 23 different entries in the "date" column. The answers are stored in variables `#0`, ..., `#22` which are used for labeling all concepts of the nominal scale (except for the top and bottom elements), and for retrieving the corresponding objects from the data base.

The second example is similar to the previous example. Inter-ordinal scales are typically used when a linear order (e. g., a price scale, a time scale) is divided into intervals with respect to their meaning. The boundaries of the intervals are usually fixed by a knowledge engineer. However, the range of possible attribute values is not always known a priori. Hence, for a first glance at the data, it has proved useful to query the data base for the minimal and the maximal value and to split up this interval into intervals of equal length. Depending on the application, it might also be useful to fix the boundaries on certain statistical measures, as for instance average, median, quantiles. These "self-adapting scales" reduce the effort needed to create the conceptual scheme, since they are re-usable.

It is planned to implement a user interface by means of which the user can edit parameters at runtime. For instance, he could first invoke an inter-ordinal scale with equidistant boundaries and then fine-tune it according to his needs.

This user interface leads to the third example, an application in control theory: Process data of the incineration plant of Darmstadt were analyzed in order to make the control system more efficient (cf. [2]). Every ten seconds, different process parameters were measured and stored in the data base, for instance "ram velocity" and "steam". While "ram velocity" describes the feed amount of waste, "steam" describes the steam mass flow produced by the combustion. The ram velocity does not influence the steam volume directly, but only with a time delay of about half an hour, in which the material passes through a drying process.

Hence the engineer had to compute a 'shift' in the data base, assigning to each time stamp not the actual "steam" value, but the value of half an hour later. Instead of computing the shift in the actual process data base, it also can be done by an inner join in the query generated by TOSCANA. When the 'shift time' is kept variable, the user can change it via the interface during the runtime of TOSCANA. That can be used, for instance, for determining the time delay of two variables experimentally: The engineer examines the nested line diagram of the corresponding scales for ordinal dependencies. By varying the shift time, he tries to augment the dependencies, and to determine in this way the time delay.

The possibility of using parameters is also of interest for filters that control the data flow from the data base to TOSCANA. They are discussed in the next subsection.

5.2 Filtering the Objects of the Many-valued Context

In many applications, users are interested in analyzing only a specific subset of objects of the many-valued context. Usually such a subset is determined conceptually, being the extent (more rarely the contingent) of a concept of a suitable combination of conceptual scales. TOSCANA provides for the possibility of "zooming" into that specific concept by mouse click. In the sequel of the analysis only objects belonging to that concept are considered.

Conceptual scales are an answer to the problem that often users are not able to formulate a precise criterion as to how to restrict the set of objects. Conceptual scales provide different concepts to zoom in on, and the user can choose the one that best fits his needs. A typical application is a retrieval system, where users often have only a vague idea about what they are looking for. However, if a precise criterion is at hand, one wants to apply it directly without making the detour of using a conceptual scale. For instance, if one is interested in the withdrawals from the bank account during the past quarter only, no explicit time scale is needed.

For such applications, the conceptual scheme should be extended by *filters*. In addition to conceptual scales, the user can choose filters from a menu. When a filter is activated, then objects are only considered for display if they pass the filter. A filter is realized as an SQL-fragment that is added by AND to the conditions provided by the chosen scales.

The remarks about parameters in the previous subsection apply to filters as well. We give two typical examples for the use of parameters in filters. The first example is a conceptual knowledge system about pipelines (cf. [9]). The system was built in order to support an engineer by choosing suitable pipes, fittings, and valves for the construction of a pipeline. The desired functionality of the pipeline determines – among other parameters – pressure p and working temperature t. These parameters can for instance be imported from a CAD system, or are entered by the engineer. A condition for the pipes used in the construction is $y_{\mathrm{I/II}} \geq p/\sigma_{\mathrm{zul}}$, where $y_{\mathrm{I/II}}$ is a value depending on the measures of the pipe, and σ_{zul} is the maximal strain allowed in the wall of the tube. $y_{\mathrm{I/II}}$ is stored in the data base, and σ_{zul} depends on the temperature t and the material of the tube,

and can be read from a table. Hence the engineer can implement a filter sorting out all components not satisfying the condition.

The second example is again the controlling system of Sects. 2 and 3. As described above, we can construct a filter that only accepts withdrawals effected in a certain period, e. g., the last quarter. The interface for editing parameters introduced in Sect. 5.1 provides the possibility of examining the withdrawals of any period required. When the user activates the filter, he is asked for start and end date.

5.3 Focussing on Specific Aspects of the Objects

The controlling system is an example of focussing on different aspects of the data. There we focus not only on withdrawal numbers, but also on the sum of bankbook values. Now we discuss how this example fits into the formalization described in Sect. 4.

Once the user has chosen one or more scales, TOSCANA determines for each concept of the corresponding concept lattice a set S of objects – in most cases its extent or its contingent. In Sect. 1 we mentioned that the user can choose for each concept whether all names of the objects in S shall be displayed or only the cardinality of S. A third standard aspect in TOSCANA is the display of relative frequencies. The last two aspects are examples of algebraic operations.

The focussing in the example of Sects. 2 and 3 can be understood as being composed of two actions: Firstly, instead of working on the set S, the sequence $(m(g) \in W_m)_{g \in S}$ is chosen. In the controlling example, this *projection* assigns to each withdrawal from S the corresponding book-value. Secondly, the sum $\sum(m(S)) := \sum_{g \in S} m(g)$ is computed (and displayed). The latter is done in the relational structure assigned to the corresponding attribute. In TOSCANA, that is realized by a modification of the way the SQL-queries are generated: the standard COUNT-command used for the computation of the frequency of S is replaced by a SUM-command operating on the column "value".

Another important feature is to sort the list of displayed objects according to some attribute. This means that, for a specified attribute $m \in M$, the objects are ordered by a linear order which is given as binary relation in the relational structure $\mathbb{R}_m$. We define $g_1 \leq g_2$ iff $m(g_1) \leq m(g_2)$. This ordering can be combined with the above described projection. For instance, instead of displaying the sum of book-values, the user might want to see the names of the corresponding shops, but ordered by decreasing book-value.

Depending on the application, there may be other aspects to focus on. In general, this focussing can be realized by manipulating the SQL-queries. The idea is to store them in the conceptual scheme, and to make them available for the user in the same way that he already can switch between the display of the objects and the display of their frequency: the focus can be chosen either globally for the displayed scale(s) by a pull-down menu, or locally for each concept by a pop-up menu which appears by clicking on the concept.

In general, focussing is independent from determining the extents or contingents. In practice, however, these two steps interact in order to increase per-

formance. For instance, if the user is interested only in frequencies, then the concrete object names need not be retrieved from the data base.

5.4 Highlighting Interesting Concepts

Focussing also can be understood in a different setting. It also means drawing the user's attention to those concepts where the frequency of objects (or the sum of book-values, etc.) is extraordinarily high (or low). The determination of these concepts is based on the frequency distribution of the nested line diagram. This distribution can be represented – without its conceptual order – by a contingency table with entry n_{ij} in cell (i, j) where i (j, resp.) is an object concept of the first (second) scale.

As a refinement of Pearson's Chi- Square calculations for contingency tables we recommend calculating for each cell (i, j) the *expected frequency* $e_{ij} := (n_i n_j)/n$ ("expected" means "expected under independence assumption") where n_i (n_j, resp.) is the frequency of object concept i (j, resp.) and n is the total number of all objects. To compare the distribution of the observed frequencies n_{ij} and the expected frequencies e_{ij}, one should study the *dependency double matrix* (n_{ij}, e_{ij}).

Pearson's Chi-Square calculations reduce the dependency double matrix to the famous $\chi^2 := \sum_{ij}((n_{ij} - e_{ij})^2/e_{ij})$. But the matrix also can be used as a whole in order to highlight interesting places in a nested line diagram:

- If the user wants to examine the dependency double matrix in detail, then he may choose to display their entries at the corresponding concepts. Additionally, one of the matrices of differences $n_{ij} - e_{ij}$, quotients $(n_{ij} - e_{ij})/e_{ij}$, or quotients n_{ij}/e_{ij} may be displayed in the same way. The conceptual structure represented by the line diagram helps us to understand the dependency double matrix.
- If a less detailed view is required, then the calculation component can generate graphical marks which indicate those concepts where the matrix entries are above or below a given threshold. A typical condition in applications is "$e_{ij} > k$ and $n_{ij}/e_{ij} > p$" where k and p are parameters which can be chosen on a suitable scale.

The Chi-Square formula is a very rough reduction of the information about dependencies, but, clearly, the degree of reduction depends on the purpose of the investigation. If one is interested not only in having an index showing whether there is a dependency, but in understanding the dependencies between two many-valued attributes with respect to chosen scales in detail, then one should carefully study the distribution of observed and expected frequencies.

6 Outlook

The connections between conceptual scales and relational structures should be studied extensively. Therefore, practical relevant examples containing both parts

should be considered. Some examples are: optimization problems (shortest paths, flow problems, minimal covering, minimizing electrical circuits), similarities and distances connected with conceptual structures, valuations of states of a system, decision problems in conceptually complex situations, applications in logics and fuzzy-logics, hierarchies with valuations, graphs with valuations, orders of substructures (e. g., lattices of subgroups of a group), congruence relations.

It is of particular interest to examine the compatibility of various conceptual scales and relational structures on the same set of attribute values. From a formal point of view both structures are of the same generality in the sense that each conceptual scale can be described as a relational structure and vice versa. But they are used differently: conceptual scales generate overviews for knowledge landscapes, while relational structures serve for computations.

We plan to extend conceptual data systems to *conceptual knowledge systems* which comprise methods of knowledge acquisition and knowledge inference. That indicates the need for extending TOSCANA not only by a computational component, but also by a logical component. Since there are already many tools focussing on logical aspects of the data, we suggest not reinventing the wheel: similarly to the computational component, the logical component should provide an interface so that most of the logical computations can be delegated. One should especially consider combining conceptual data systems with systems for description logics and conceptual graphs.

This paper discussed how computational components can support conceptual data processing. One should also investigate how, vice-versa, results of data analysis and retrieval activities in conceptual data systems can be made accessible to other systems. This discussion may lead to hybrid knowledge systems composed of conceptual, computational and logical subsystems, each focussing on different aspects of the knowledge landscape inherent in the data.

References

1. B. Ganter, R. Wille: Formale Begriffsanalyse: Mathematische Grundlagen. Springer, Heidelberg 1996
2. E. Kalix: Entwicklung von Regelungskonzepten für thermische Abfallbehandlungsanlagen. Diplomarbeit, TH Darmstadt 1997
3. W. Kollewe, C. Sander, R. Schmiede, R. Wille: TOSCANA als Instrument der bibliothekarischen Sacherschließung. In: H. Havekost, H.-J. Wätjen (eds.): *Aufbau und Erschließung begrifflicher Datenbanken.* (BIS)-Verlag, Oldenburg 1995, 95–114
4. W. Kollewe, M. Skorsky, F. Vogt, R. Wille: TOSCANA — ein Werkzeug zur begrifflichen Analyse und Erkundung von Daten. In: R. Wille, M. Zickwolff (eds.): *Begriffliche Wissensverarbeitung — Grundfragen und Aufgaben.* B.I.–Wissenschaftsverlag, Mannheim 1994
5. P. Luksch, R. Wille: A mathematical model for conceptual knowledge systems. In: H.-H. Bock, P. Ihm (eds.): *Classification, data analysis and knowledge organization.* Springer, Berlin 1991, 156–162
6. S. Prediger: Logical scaling in formal concept analysis, *Proceedings of the international conference on conceptual structures.* LNAI, Springer, Heidelberg 1997 (to appear)

7. P. Scheich, M. Skorsky, F. Vogt, C. Wachter, R. Wille: Conceptual data systems. In: O. Opitz, B. Lausen, R. Klar (eds.): *Information and classification.* Springer, Heidelberg 1993, 72–84
8. G. Stumme: Exploration tools in formal concept analysis. In: *Ordinal and symbolic data analysis.* Studies in classification, data analysis, and knowledge organization **8**, Springer, Heidelberg 1996, 31–44
9. N. Vogel: Ein begriffliches Erkundungssystem für Rohrleitungen. Diplomarbeit, TH Darmstadt 1995
10. F. Vogt, C. Wachter, R. Wille: Data analysis based on a conceptual file. In: H.-H. Bock, P. Ihm (eds.): *Classification, data analysis, and knowledge organization.* Springer, Heidelberg 1991, 131–140
11. F. Vogt, R. Wille: TOSCANA — A graphical tool for analyzing and exploring data. In: R. Tamassia, I. G. Tollis (eds.): *Graph Drawing '94*, Lecture Notes in Computer Sciences **894**, Springer, Heidelberg 1995, 226–233
12. R. Wille: Restructuring lattice theory: an approach based on hierarchies of concepts. In: I. Rival (ed.): *Ordered sets.* Reidel, Dordrecht–Boston 1982, 445–470
13. R. Wille: Lattices in data analysis: how to draw them with a computer In: I. Rival (ed.): *Algorithms and order.* Kluwer, Dordrecht–Boston 1989, 33–58
14. R. Wille: Conceptual landscapes of knowledge: A pragmatic paradigm of knowledge processing. This volume
15. K. E. Wolff: A first course in formal concept analysis – How to understand line diagrams. In: F. Faulbaum (ed.): *SoftStat '93, Advances in statistical software* **4**, Gustav Fischer Verlag, Stuttgart 1993, 429–438