Theory-driven Logical Scaling

Conceptual Information Systems meet Description Logics

Susanne Prediger, Gerd Stumme

Technische Universität Darmstadt, Fachbereich Mathematik, Schloßgartenstr. 7,
D–64289 Darmstadt; {prediger, stumme}@mathematik.tu-darmstadt.de

To be published in:
P. Lambrix et al.: Proceedings DL'99. CEUR Workshop Proceedings Vol. 22, 1999
and in E. Franconi et al.: Proceedings KRDB'99. CEUR Workshop Proceedings Vol. 21, 1999
(http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/)

1 Introduction

Conceptual Information Systems ([8]) have been developed for conceptual data analysis and are based on the mathematical theory of Formal Concept Analysis ([4]). A Conceptual Information System provides a front-end for a (relational) database. It uses conceptual hierarchies to unfold the conceptual structure of the data and to support on-line navigation through the data. By so-called *conceptual scales* ([3]), the relevant information can be extracted from the database and stored in a table with an object-attribute-relation (called *formal context*) from which one can derive a conceptual hierarchy (called *concept lattice*) for the actual part of the data. As far as they are needed in this paper, the basics of Conceptual Information Systems and conceptual scales are provided in Section 2.

For a conceptual scale, there is always a trade-off between its size and its soundness with respect to future updates of the database. There are two approaches of designing conceptual scales: data-driven design and theory-driven design.

In *theory-driven design*, knowledge about the application domain is used to exclude impossible combinations of attributes. This keeps the conceptual scales small – and their concept lattices easier to interpret. Theory-driven design is only applicable if there is enough knowledge about which types of objects may occur in the database. If this information is missing, the diagrams may become unnecessarily large.

The second approach is called *data-driven design.* If there is no (or only few) knowledge available, the scales are designed to fit the actual data only, and not to conform to all possible updates of the database. If an update violates the structure of the scale, the user is warned, and the scale has to be redesigned. Hence, data-driven design is not applicable if the database is frequently updated.

A more general approach than conceptual scaling is presented in [5]: (data-driven) *logical scaling.* Instead of using conceptual scales, it uses the terminology of a formal language like Description Logic for extracting information. Logical scaling is shortly recalled in Section 2.

While data-driven logical scaling has the advantage of a more powerful language for defining scales, it has the same drawbacks as data-driven design of ordinary conceptual scales. In Section 3, we introduce *theory-driven logical scaling* which combines both efforts. It determines typical objects and excludes all combinations of attributes which cannot occur because of the semantics of the applied Description Logic.

Theory-driven logical scaling combines three ideas: the use of a terminology for scaling ([5]), the application of Attribute Exploration ([2]) – a knowledge acquisition tool – for bridging the gap between data-driven and theory-driven design of conceptual scales ([7]), and the utilization of a subsumption algorithm of Description Logics as an 'expert' for Attribute Exploration ([1]). Theory-driven logical scaling can be used to set up a Conceptual Information System even when the database is only partially given in the beginning.

2 Conceptual Information Systems and Data-Driven Logical Scaling

Definition. A *(formal) context* is a triple $\mathbb{K} := (G, M, I)$ where G and M are sets and I is a relation between G and M. The elements of G and M are called *objects* and *attributes*, resp., and $(g, m) \in I$ is read *"object g has attribute m".*

A *(formal) concept* is a pair (A, B) with $A \subseteq G$ and $B \subseteq M$ such that A and B are maximal with $A \times B \subseteq I$. The set A is called the *extent* and the set B the *intent* of the concept. The *concept lattice* of $\mathbb{K}$ (denoted by $\underline{\mathfrak{B}}(\mathbb{K})$) is the set of all its concepts together with the hierarchical subconcept–superconcept–relation

$$(A, B) \leq (C, D) :\Longleftrightarrow A \subseteq C \quad (\Longleftrightarrow B \supseteq D) \ .$$

Figure 1 shows a formal context. It has four persons as objects, which are described by six attributes. In the *line diagram* of its concept lattice the name of an object g is always attached to the circle representing the smallest concept with g in its extent; dually, the name of an attribute m is always attached to the circle representing the largest concept with m in its intent. This allows us to read the context relation from the diagram because an object g has an attribute m if and only if there is an ascending path from the circle labeled by g to the circle labeled by m. The extent of a concept consists of all objects whose labels are below in the diagram, and the intent consists of all attributes attached to concepts above in the hierarchy.

	person	wine drinker	red wine drinker	white wine drinker	Bordeaux drinker	red Bordeaux drinker
Mr. Smith	×	×	×			
Mrs. Miller	×	×		×		
Miss Cooper	×	×	×		×	×
Mr. Davis	×	×	×		×	×

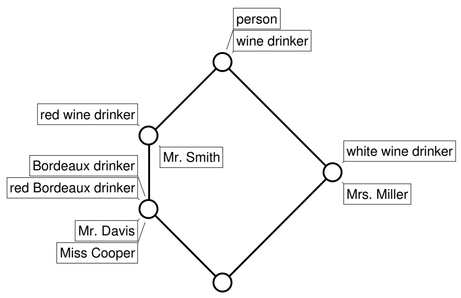

Figure 1: A formal context about wine drinkers and a line diagram of its concept lattice

For example, the concept labeled by Mr. Smith and red wine drinker has {Mr. Smith, Miss Cooper, Mr. Davis} as extent, and {red wine drinker, wine drinker, person} as intent. In the diagram, one can for instance see that the two attributes Bordeaux drinker and red Bordeaux drinker generate the same concept. This indicates that among the four persons there is no-one drinking white Bordeaux. $(*)$

A *Conceptual Information System* consists of a *many-valued context* and a set of *conceptual scales.* A many-valued context may not only have crosses (i. e., yes/no) as entries, but values of attributes pairs. It can be seen as a table of a relational database with the column containing the objects being a primary key.

Definition. A *many-valued context* is a tuple $\mathbb{K} := (\mathbb{G}, \mathbb{M}, (\mathbb{W}_{\gtrdot})_{\gtrdot \in \mathbb{M}}, \mathbb{I})$ where G is a set of objects, M a set of attributes, each W_m a set of possible values for the attribute $m \in M$, and $I \subseteq G \times \{(m,w) \mid m \in M, w \in W_m\}$ a relation with $(g,m,w_1) \in, (g,m,w_2) \in I \Rightarrow w_1 = w_2$. $(g,m,w) \in I$ is read "object g has value w for attribute m".

A *conceptual scale* is a one-valued context which has as objects possible values of the database attributes. It is used to extract the relevant information from the many-valued context such that a concept lattice can be generated. The choice of the attributes of the scale is purpose-oriented and reflects the understanding of an expert of the domain.

Definition. A *conceptual scale* for a subset $B \subseteq M$ of attributes is a (one-valued) formal context $\mathbb{S}_\mathbb{B} := (\mathbb{G}_\mathbb{B}, \mathbb{M}_\mathbb{B}, \mathbb{I}_\mathbb{B})$ with $G_B \subseteq \times_{m \in B} W_m$. The *realized scale* $\mathbb{S}_\mathbb{B}(\mathbb{K})$ is defined by $\mathbb{S}_\mathbb{B}(\mathbb{K}) := (\mathbb{G}, \mathbb{M}_\mathbb{B}, \mathbb{J})$ with $(g,n) \in J$ if and only if there exists $w = (w_m)_{m \in B} \in G_B$ with $(g,m,w) \in I$, for $m \in B$, and $(w,n) \in I_B$.

The idea is to replace the attribute values in W_m which are often too specific

drinks	
Person	Wine
Mr. Smith	Casa Solar
Mrs. Miller	Staehle
Miss Cooper	Figeac
Mr. Davis	Figeac
Mr. Davis	Casa Solar

Wines				
	red wine	white wine	Bordeaux	Price
Figeac	×		×	49,90
Staehle		×		14,90
Casa Solar	×			5,95

Figure 2: Data Base *Wines and Clients*

by binary, more general attributes which are provided in M_B. For an example, see below.

In implemented Conceptual Information Systems, the many-valued context is realized as a table in a relational database. The set G_B of a conceptual scale $\mathbb{S}_\mathbb{B}$ is then replaced by corresponding SQL statements. In the *realized scale*, the objects of the conceptual scale (i. e. the values of the database attributes) are replaced by the corresponding objects of the many-valued context.

Figure 2 shows a small database of a (fictive) wine retailer. For this introductory example, we consider only the table wines as many-valued context $\mathbb{K}$. The diagram in Figure 3 shows the realized scale $\mathbb{S}_{\{\mathsf{Price}\}}(\mathbb{K})$. The chosen attributes reflect the view of the analyst about prices. It divides the price range in four (non-disjoint!) categories: below 5 DM (by the attribute very cheap), below 10 DM (cheap), above 10 DM (expensive), above 20 DM (very expensive). The objects of the corresponding conceptual scale are all possible prices. In the realized scale, each price is replaced by those objects which have this price.

The design of this scale is theory-driven. It reflects the understanding that a wine is either cheap or expensive, that each very cheap wine is also cheap, and that each very expensive wine is also expensive. This understanding excludes ten out of 16 possible combinations of the attributes as concept intents. With the current data, the concept labeled by very cheap is not realized. That means that, at the moment, there are no wines in the database which are very cheap. Data-driven design of the scale would have omitted this concept, but might not be consistent with future updates of the database. Hence data-driven design corresponds to the Closed World Assumption, while theory-driven design corresponds to the Open World Assumption.

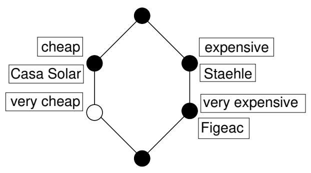

Figure 3: *The realized scale for the price*

wine drinker	:=	person $\sqcap$ $\exists$drinks.(red wine $\sqcup$ white wine)
red wine drinker	:=	person $\sqcap$ $\exists$drinks.red wine
white wine drinker	:=	person $\sqcap$ $\exists$drinks.white wine
Bordeaux drinker	:=	person $\sqcap$ $\exists$drinks.Bordeaux
red Bordeaux drinker	:=	person $\sqcap$ $\exists$drinks.(red wine $\sqcap$ Bordeaux)

Figure 4: Terminology *Wine drinkers*

With 'traditional' conceptual scaling, the context shown in Figure 1 cannot be obtained as a realized scale of the given database. In the sequel, we show how it can be obtained by data-driven logical scaling. Its theory-driven counterpart is introduced in Section 3.

In [5], we presented (data-driven) logical scaling as an alternative method that allows a more explicit and more powerful description of the attributes which are introduced for the scaling process. The basic idea of logical scaling consists of using a formal language like Description Logic to define a terminology with attributes (called concepts (!) in DL) out of the attributes and relations of different tables of the database.

In the *terminology* (TBox), a set of attributes is defined by terms of the Description Logic like it is done in Figure 4. The formal context in Figure 1 is the *realized scale* that we can *derive* from the database (ABox) in Figure 2 with the terminology in Figure 4. Its objects are the persons, its attributes are the attributes defined in the terminology, and the relation I is given by the semantics of the formal language: an object g is in relation with an attribute m if g satisfies the term defining m. (For a formal definition, refer to [5].)

If the conceptual scale is supposed to conform to updates of the database, for example to the introduction of a white Bordeaux drinker (see $(*)$), a larger conceptual scale must be created. This is done by theory-driven scaling.

3 Creating Conceptual Scales by Theory-Driven Logical Scaling

For creating a conceptual scale that is large enough for all possible updates of the database, we use Attribute Exploration ([2]), a knowledge acquisition algorithm. In order to exclude impossible combinations of the attributes, the algorithm generates questions of the form 'Is a wine drinker who is also a Bordeaux drinker and a red wine drinker always a red Bordeaux drinker?'. If the question is denied, then the user has to provide a counter-example. In [7], Attribute Exploration is used for extending data-driven to theory-driven conceptual scales with as few interaction of the domain expert as possible.

In logical scaling, the necessary expert knowledge is already explicitly formalized in the terminology. That is why Attribute Exploration can be combined

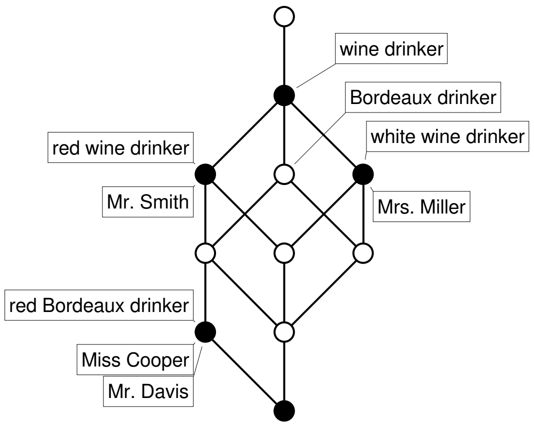

Figure 5: Concept lattice of the theory-driven scale

with a subsumption algorithm of Description Logic as 'expert' ([1]). This can be done with each logic that has a complete subsumption algorithm which generates a counter-example for each non-valid subsumption. In this paper, we use the language $\mathcal{ALC}$ ([6]).

For answering the question mentioned above, the subsumption algorithm solves the equivalent question if

$P :\equiv$ wine drinker $\sqcap$ Bordeaux drinker $\sqcap$ red wine drinker $\sqcap \neg$ red Bordeaux drinker

is inconsistent with respect to the terminology in Figure 4. In order to show that P is inconsistent, the subsumption algorithm tries to generate a counter-example. If this fails, P is consistent (and the question is affirmed). Here the question is denied because the algorithm returns three new (dummy) objects as counter-example: P7, W5, and W6 with W5 having only the attribute red wine, W6 having only the attributes Bordeaux and white wine, an the relations drinks(P7,W5) and drinks(P7,W6). The counter-example is added (temporarily, just for creating the conceptual scale) to the database in Figure 2 and to the context in Figure 1. Then Attribute Exploration generates the next open question and passes it to the subsumption algorithm. In total, Attribute Exploration generates eight questions. The subsumption algorithm denies four of them. For the others, it provides four counter-examples.

The final result is a list of counter-examples which determines the structure of the conceptual scale. (Equivalently, the structure is determined by the list of affirmed questions.)

The (theory-driven) conceptual scale is derived from the concept lattice of the extended context. For each concept, one clause consisting of the attributes of the terminology is introduced which describes the intent of the concept. For instance, P7 is replaced by red wine drinker $\sqcap$ Bordeaux drinker $\sqcap \neg$ red Bordeaux drinker. These objects are used for deriving the realized scale at runtime. The line diagram of the concept lattice of the realized scale is shown in Figure 5. Here one can see which attribute combinations can principally exist according to the terminology, and which of them are realized by the actual data. For instance one can see that the observation made in Section 2 that there is no white Bordeaux drinker (see $(*)$) does not hold in general, but only for the four listed persons.

If one starts the generation of the conceptual scale from an empty database, the same scale will arise, but in its realized scale there will be no realized concepts (beside the bottom concept). As the database grows, more and more concepts become realized. Hence, theory-driven logical scales can also be used for analyzing the degree of completeness of the database with regard to 'typical' objects of the terminology.

In contrast to the terminology in Figure 4, the concept lattice in Figure 5 visualizes the subsumption hierarchy. It combines the intensional part of a Description Logic (the TBox) with its extensional part (the ABox).

We conclude with the observation that, in theory-driven logical scaling, Description Logics and Formal Concept Analysis enrich each other. From the viewpoint of Formal Concept Analysis, the use of a Description Logic allows to extend the scaling process in Conceptual Information Systems to more complicated data structures than just one many-valued context. From the viewpoint of Description Logics, Conceptual Information Systems provide a graphical user interface which supports the navigation through and exploration of the knowledge captured by a Description Logic.

References

[1] F. Baader: Computing a minimal representation of the subsumption lattice of all conjunctions of concept defined in a terminology. In: G. Ellis, R. A. Levinson, A. Fall, V. Dahl (eds.): *Proc. Intl. KRUSE Symposium*, August 11–13, 1995, UCSC, Santa Cruz 1995, 168–178.

[2] B. Ganter: *Two basic algorithms in concept analysis.* FB4-Preprint 831, TH Darmstadt 1984.

[3] B. Ganter, R. Wille: Conceptual scaling. In: F. Roberts (ed.): *Applications of combinatorics and graph theory to the biological and social sciences*, Springer, New York 1989, 139–167.

[4] B. Ganter, R. Wille: *Formal Concept Analysis: Mathematical Foundations.* Springer, Berlin 1999.

[5] S. Prediger: Logical Scaling in Formal Concept Analysis. In: D. Lukose et.al. (eds.): Conceptual Structures: Fulfilling Peirce's Dream, LNAI 1257, Springer, Berlin 1997, 332–341.

[6] M. Schmidt–Schauß, G. Smolka: Attributive concept descriptions with complements. In: *Artificial Intelligence* 48, 1991.

[7] G. Stumme: Acquiring Expert Knowledge for the Design of Conceptual Scales. In: D. Fensel, R. Studer (Hrsg.): *Knowledge Acquisition, Modeling, and Management.* Proc. EKAW '99, LNAI 1621, Springer, Heidelberg 1999, 271–286

[8] F. Vogt, R. Wille: TOSCANA – A graphical tool for analyzing and exploring data. In: R. Tamassia, I. G. Tollis (eds.): *Graph Drawing '94*, In: LNCS 894, Springer, Heidelberg 1995, 226–233.