An Algebraic Method for Compressing Symbolic Data Tables

Yannis Tzitzikas

Institut d'Informatique
F.U.N.D.P. (University of Namur)
Rue Grandgagnage 21, B-5000, Belgium
Email : ytz@info.fundp.ac.be

Current address and affiliation:
Department of Computer Science, University of Crete
P.O.Box 2208, Heraklion, Crete, GR-714 09 GREECE
and
Institute of Computer Science (ICS)
Foundation for Research and Technology - Hellas (FORTH)
Science and Technology Park of Crete, Vassilika Vouton
P.O.Box 1385, Heraklio, Crete, GR 7110 GREECE
Tel: +30 2810 391 633 Fax: +30 2810 391 638
E-Mail: tzitzik@csi.forth.gr
Web page: http://www.csi.forth.gr/ tzitzik

Abstract. Although symbolic data tables summarize huge sets of data they can still become very large in size. This paper proposes a novel technique for compressing a symbolic data table using the recently emerged *Compound Term Composition Algebra*. One advantage of CTCA is that the closed world hypotheses of its operations can lead to a remarkably high "compression ratio". The compacted form apart from having much lower storage space requirements, it allows designing more efficient algorithms for symbolic data analysis.

Keywords: Symbolic Data Tables, Compression.

1 Introduction

As recent surveys state[1], the world produces between 1 and 2 exabytes (2^{60} bytes) of unique information per year, 90% of which is digital and with a 50% annual growth rate. This plethoric growth rate has stimulated the development of new techniques and automated tools for assisting the transformation of large amounts of data into useful information and knowledge (see data mining and knowledge discovery in databases). *Symbolic data analysis* [3, 4] has been introduced in order to solve the problem of the analysis of data that are given on an aggregated form, i.e. where quantitative variables are given by intervals and where categorical variables are given by histograms. This kind of data are generated when we summarize huge sets of data. Inescapably, even a symbolic data table could become very large in size, making its management problematic in terms of both storage space and computational time.

This paper aims to convey some recent advances from the area of knowledge representation (in particular from the area of faceted taxonomies and faceted classification), that could be exploited for symbolic data analysis. Specifically, this paper gives the theoretical foundation of a novel method that can be used to *compress* (i.e. to reduce the storage space requirements) of large symbolic data tables. The proposed compression is lossless i.e. from the compressed form we can infer exactly what we can from the original symbolic data table.

The contribution of the method is not exhausted to storage space minimization as the resulting compact form could allow the design more efficient symbolic analysis algorithms (e.g. for clustering or classification).

This paper describes in detail (and generalizes) the ideas first sketched in [12]. The rest of this paper is organized as follows. Section 2 sketches the idea and Section 3 recalls the basics of the *Compound Term Composition Algebra* (CTCA), upon which the proposed method is founded. Subsequently, Section 4 describes in more detail the steps of the proposed technique and Section 5 gives some indicative examples of compression using CTCA. Section 6 compares the proposed technique with the existing compression algorithms and discusses in brief the compression ratio that can be achieved. Finally, Section 7 concludes the paper and identifies issues for further research.

[1] http://www.sims.berkeley.edu/research/projects/how-much-info-2003/

2 The Approach in Brief

A *symbolic data table* is a table of data where the columns are the *symbolic variables* which are used in order to describe a set of units called *individuals*. Rows are called *symbolic descriptions* of these individuals because they are not as usual, only tuples of single quantitative or categorical values. For instance, the values of the cells can be intervals (if the variable is quantitative) or frequency distributions (if the variable is categorical). Recall that in classical data analysis a cell can have a single quantitative or categorical value. In general, we could distinguish variables according to their range to (a) single quantitative (e.g. age=18), (b) single categorical (or taxonomic) (e.g. color=red), (c) multi-valued quantitative or categorical (e.g. age=$\{11, 18\}$, color=$\{red, green\}$) (d) interval (e.g. age=[10,20]), and (e) multi-valued with weights (e.g. histograms). Clearly, (a) and (b) are special cases of (c), while (c) is special case of (e) (i.e. when all weights are either 0 or 1) for more see [4].

This paper proposes a method for compacting a symbolic data table by exploiting the *Compound Term Composition Algebra* (*CTCA*). CTCA is a recently emerged algebra that allows specifying the *valid* (meaningful) *compound terms* (conjunction of terms) over a *faceted taxonomy* in a flexible and efficient manner (for more see [16, 15]). A system around CTCA has already been developed (FASTAXON [17]) and there has already been proposed a Web annotation language that allows exchanging faceted taxonomies and expressions of CTCA (for more see XFML+CAMEL [2]). In brief, a faceted taxonomy is a set of taxonomies each one describing the domain of interest from a different (preferably orthogonal) point of view (for more about faceted classification and analysis see [11, 5, 18, 7, 8]). Faceted taxonomies are used in Web Catalogs, Libraries [8], Software Repositories [9, 10], and several others application domains. Current interest in faceted taxonomies is also indicated by several recent or ongoing projects (like FATKS[2], FACET[3], FLAMENGO[4]) and the emergence of XFML [1](Core-eXchangeable Faceted Metadata Language) a markup language for applying the faceted classification paradigm on the Web. Having a faceted taxonomy each domain object (e.g. book or Web page) can be indexed using a *compound term*, i.e., a set of terms containing one or more terms from each facet. We shall use the term *materialized faceted taxonomy* to refer to a faceted taxonomy accompanied by a set of object indices. For example, Figure 1 shows a very small but indicative materialized faceted

[2]http://www.ucl.ac.uk/fatks/database.htm

[3]http://www.glam.ac.uk/soc/research/hypermedia/facet_proj/index.php

[4]http://bailando.sims.berkeley.edu/flamenco.html

taxonomy consisting of three facets and two indexed objects.

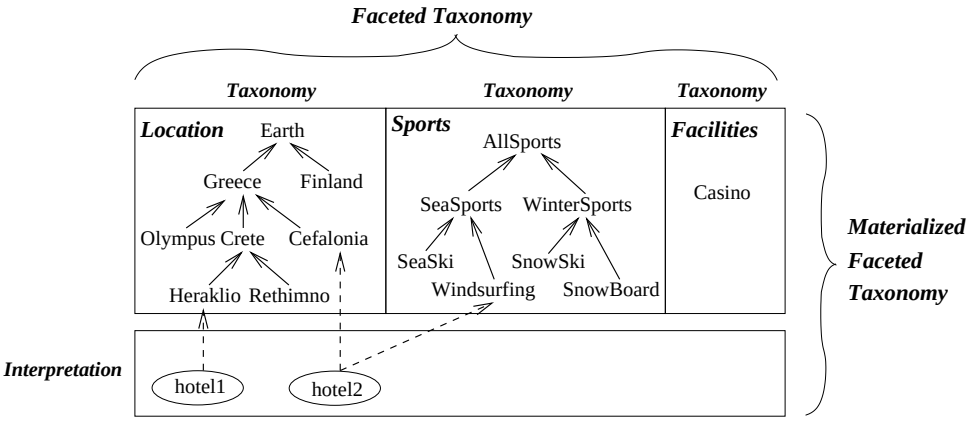

Figure 1: A faceted taxonomy for indexing hotel Web pages

Roughly, and according to the above perspective and phraseology, each symbolic data table can be viewed as a *materialized faceted taxonomy*. This analogy is not hard to grasp. Each symbolic variable can be viewed as a *facet*. Now the range of each symbolic variable can be viewed as a *taxonomy*, i.e. as a partially ordered set of terms (clearly, categories, intervals, and subsets are partially ordered domains). Now each row of the symbolic data table can be viewed as an *object* that has been *indexed* according to a faceted taxonomy, i.e. as an object that has been associated with a *compound term* of the faceted taxonomy, i.e. with a set of values from the range of the symbolic variables.

Several algorithms for finding an expression of CTCA that specifies those compound terms that are *extensionally valid* in a materialized faceted taxonomy were given and analyzed in [13, 14]. In other words, these algorithms compute and return an expression of CTCA that specifies the set of all distinct compound terms that are meaningful, where a compound term is considered meaningful if it is applicable to at least one object of the object base. It follows, that the same algorithms can be exploited for the problem at hand, specifically for finding a short (in storage space) expression of CTCA that specifies the rows of a symbolic data table.

Specifically, this paper focuses on symbolic variables with partially ordered ranges, i.e. taxonomically-ordered categorical, multi-valued quantitative or categorical, and interval-valued variables. The reason is that in this case the employment of CTCA can yield remarkably high compression ratios. However, CTCA can be applied even on unordered ranges, i.e. on sets (we can view a set as a poset with an empty ordering relation), so the proposed method can be also applied on variables whose range is a set of histograms. However, an issue for further research is

to investigate ordering relations over histograms because their availability would allow obtaining higher compression ratios even for this kind of variables (especially when lossy compression is tolerable).

3 Faceted Taxonomies and the *Compound Term Composition Algebra*

Table 1 below recalls in brief the basic notions around taxonomies, faceted taxonomies and materialized faceted taxonomies (for more please refer to [16]).

Name	Notation	Definition
terminology	$\mathcal{T}$	a set of names called terms
subsumption	$\leq$	a preorder relation (reflexive and transitive)
taxonomy	$(\mathcal{T}, \leq)$	$\mathcal{T}$ is a terminology, $\leq$ a subsumption relation over $\mathcal{T}$
faceted taxonomy	$\mathcal{F}$= $\{F_1, ..., F_k\}$	$F_i = (\mathcal{T}_i, \leq_i)$, for $i = 1, ..., k$ and all $\mathcal{T}_i$ are disjoint
compound term over $\mathcal{T}$	s	any subset of $\mathcal{T}$ (i.e. any element of $\mathcal{P}(\mathcal{T})$)
compound terminology	S	a subset of $\mathcal{P}(\mathcal{T})$ that includes $\emptyset$
compound ordering	$\preceq$	$s \preceq s'$ iff $\forall t' \in s' \ \exists t \in s$ such that $t \leq t'$.
broaders of s	$\mathrm{Br}(s)$	$\{s' \in P(\mathcal{T}) \mid s \preceq s'\}$
narrowers of s	$\mathrm{Nr}(s)$	$\{s' \in P(\mathcal{T}) \mid s' \preceq s\}$
broaders of S	$Br(S)$	$\cup\{\mathrm{Br}(s) \mid s \in S\}$
narrowers of S	$Nr(S)$	$\cup\{\mathrm{Nr}(s) \mid s \in S\}$
object domain	Obj	any denumerable set of objects
interpretation of $\mathcal{T}$	I	any function $I : \mathcal{T} \rightarrow 2^{Obj}$
model of $(\mathcal{T}, \leq)$ induced by I	$\bar{I}$	$\bar{I}(t) = \cup\{I(t') \mid t' \leq t\}$
materialized faceted taxonomy	$(\mathcal{F}, I)$	$\mathcal{F}$ is a faceted taxonomy $\{F_1, ..., F_k\}$, I is an interpretation of $\mathcal{T}$= $\bigcup_{i=1,k} \mathcal{T}_i$

Table 1: Notations

CTCA was proposed for defining the meaningful compound terms over a faceted taxonomy in a flexible and efficient manner. The problem of meaningless compound terms and the effort needed to specify the meaningful ones is a practical problem identified even by Ranganatham himself [11] (about 80 years ago) and it is probably the main reason why faceted taxonomies have not dominated every application domain despite their uncontested advantages over the single-hierarchical taxonomies. CTCA is the only well-founded and flexible solution to this

problem.

CTCA has four basic algebraic operations, namely, *plus-product* ($\oplus$), *minus-product* ($\ominus$), *plus-self-product*, ($\overset{*}{\oplus}$), and *minus-self product* ($\overset{*}{\ominus}$). They are all operations over $\mathcal{P}(\mathcal{T})$, the powerset of $\mathcal{T}$, where $\mathcal{T}$ is the union of the terminologies of all facets. The initial operands, thus the building blocks of the algebra, are the basic compound terminologies, which are the facet terminologies with the only difference that each term (for reasons of notational simplicity) is viewed a singleton. Specifically, the *basic compound terminology* of a terminology $\mathcal{T}_i$ is defined as: $T_i = \{\{t\} \mid t \in \mathcal{T}_i\} \cup \{\emptyset\}$. If e is an expression, S_e denotes the outcome of this expression and is called the *compound terminology* of e. An expression e over $\mathcal{F}$ is defined according to the following grammar ($i = 1, ..., k$):

$$e ::= \oplus_P(e, ..., e) \mid \ominus_N (e, ..., e) \mid \overset{*}{\oplus}_P T_i \mid \overset{*}{\ominus}_N T_i \mid T_i,$$

where the parameters P and N denote sets of valid and invalid compound terms over the range of the operation, respectively. Roughly, CTCA allows specifying the valid compound terms over a faceted taxonomy by providing a small set of valid (P) and a small set of invalid (N) compound terms. The self-product operations allow specifying the meaningful compound terms over one facet. Specifically, the definition of each operation of CTCA is summarized in Table 2 where S, S' denote compound terminologies. In addition, $(S_e, \preceq)$ is called the *compound taxonomy* of e.

An expression e is *well formed* iff every facet appears at most once in e, and the parameter sets P and N are always subsets of the corresponding set of *genuine compound terms*. Specifically, each parameter P (resp. N) of an operation $\oplus_P(e_1, ..., e_k)$ (resp. $\ominus_N(e_1, ..., e_k)$) should be subset of the set of genuine compound terms over the compound terminologies $S_{e_1}, ..., S_{e_k}$, i.e. subset of:

$$G_{S_{e_1},...,S_{e_k}} = S_{e_1} \oplus ... \oplus S_{e_k} - \cup_{i=1}^{n} S_{e_i}$$

From an application point of view, another important remark is that there is no need to store the set of valid compound terms that are defined by an expression, as an inference mechanism (given in [16]) can check whether a compound term s belongs to the set of compound terms defined by an expression e (i.e. whether $s \in S_e$) in polynomial time. Specifically the computational complexity of this algorithm is $O(|\mathcal{T}|^3 * |\mathcal{P} \cup \mathcal{N}|)$, where $\mathcal{P}$ denotes the union of all P parameters and $\mathcal{N}$ denotes the union of all N parameters appearing in e. So, only the faceted taxonomy and the expression have to be stored.

For example, recall the faceted taxonomy of Figure 1. One can easily see that several com-

Operation	e	S_e
product	$S_1 \oplus ... \oplus S_n$	$\{ s_1 \cup ... \cup s_n \mid s_i \in S_i\}$
plus-product	$\oplus_P(S_1, ...S_n)$	$S_1 \cup ... \cup S_n \ \cup \ Br(P)$
minus-product	$\ominus_N(S_1, ...S_n)$	$S_1 \oplus ... \oplus S_n - Nr(N)$
self-product	$\overset{*}{\oplus}(T_i)$	$P(\mathcal{T}_i)$
self-plus-product	$\overset{*}{\oplus}_P(T_i)$	$T_i \cup Br(P)$
self-minus-product	$\overset{*}{\ominus}_N(T_i)$	$\overset{*}{\oplus}(T_i) - Nr(N)$

Table 2: The operations of the Compound Term Composition Algebra

pound terms over this faceted taxonomy are meaningless, in the sense they cannot be applied to any object of the domain. For instance, we cannot do any winter sport in the Greek islands (Crete and Cefalonia) as they never have enough snow, and we cannot do any sea sport in Olympus because Olympus is a mountain. For the sake of this example, let us also suppose that only in Cefalonia there exists a hotel that has a casino, and that this hotel also offers sea ski and windsurfing sports. According to this assumption, the partition of compound terms to the set of *valid* (meaningful) compound terms and *invalid* (meaningless) compound terms is shown in Table 3.

Instead of defining this partition explicitly, with CTCA one can define it in a more flexible and quick manner. Specifically, this partition can be specified by the subsequent expression:

$$e = (Location \ominus_N Sports) \oplus_P Facilities$$

with the following P and N parameters:

$$\begin{aligned} N &= \{\{Crete, WinterSports\}, \\ &\quad \{Cefalonia, WinterSports\}\} \\ P &= \{\{Cefalonia, SeaSki, Casino\}, \\ &\quad \{Cefalonia, Windsurfing, Casino\}\} \end{aligned}$$

CTCA can be exploited both forthrightly *and* reversely, i.e. a designer can formulate an expression in order to specify quickly the desired set of compound terms, while from an existing set of compound terms an algorithm can find an expression that describes these compound terms. It is the latter direction that is appropriate for compressing a symbolic data table.

In order to apply CTCA for compacting a symbolic data table we have to generalize our setting

Valid		Invalid	
Earth, AllSports	Greece, AllSports	Crete, WinterSp.	Cefalonia, WinterSp.
Finland, AllSports	Olympus, AllSports	Rethimno, WinterSp.	Heraklio, WinterSp.
Crete, AllSports	Cefalonia, AllSports	Olympus, SeaSki	Olympus, WindSurf.
Rethimno, AllSports	Heraklio, AllSports	Crete, SnowB.	Cefalonia, SnowB.
Earth, SeaSports	Greece, SeaSports	Rethimno, SnowB.	Heraklio, SnowB.oard
Finland, SeaSports	Crete, SeaSports	Crete, SnowSki	Cefalonia, SnowSki
Cefalonia, SeaSports	Rethimno, SeaSports	Rethimno, SnowSki	Heraklio, SnowSki
Heraklio, SeaSports	Earth, WinterSp.	Finland, Cas.	Olympus, Cas.
Greece, WinterSp.	Finland, WinterSp.	Crete, Cas.	Heraklio, Cas.
Olympus, WinterSp.	Earth, SeaSki	Rethimno, Cas.	WinterSp., Cas.
Greece, SeaSki	Finland, SeaSki	SnowBoard, Cas.	SnowSki, Cas.
Crete, SeaSki	Cefalonia, SeaSki	Olympus, SeaSports	Crete, WinterSp., Cas.
Rethimno, SeaSki	Heraklio, SeaSki	Cefalonia, WinterSp., Cas.	Rethimno, WinterSp., Cas.
Earth, WindSurf.	Greece, WindSurf.	Heraklio, WinterSp., Cas.	Olympus, SeaSki, Cas.
Finland, WindSurf.	Crete, WindSurf.	Olympus, WindSurf., Cas.	Crete, SnowB., Cas.
Cefalonia, WindSurf.	Rethimno, WindSurf.	Cefalonia, SnowB., Cas.	Rethimno, SnowB., Cas.
Heraklio, WindSurf.	Earth, SnowB.	Heraklio, SnowB., Cas.	Crete, SnowSki, Cas.
Greece, SnowB.	Finland, SnowB.	Cefalonia, SnowSki, Cas.	Rethimno, SnowSki, Cas.
Olympus, SnowB.	Earth, SnowSki	Heraklio, SnowSki, Cas.	Olympus, AllSports, Cas.
Greece, SnowSki	Finland, SnowSki	Crete, AllSports, Cas.	Rethimno, AllSports, Cas.
Olympus, SnowSki	Earth, AllSports, Cas.	Heraklio, AllSports, Cas.	Crete, SeaSports, Cas.
Greece, AllSports, Cas.	Cefalonia, AllSports, Cas.	Rethimno, SeaSports, Cas.	Heraklio, SeaSports, Cas.
AllSports, Cas.	SeaSports, Cas.	Olympus, WinterSp., Cas.	Crete, SeaSki, Cas.
SeaSki, Cas.	Windsurf., Cas.	Rethimno, SeaSki, Cas.	Heraklio, SeaSki, Cas.
Earth, Cas.	Greece, Cas.	Crete, WindSurf., Cas.	Rethimno, WindSurf., Cas.
Cefalonia, Cas.	Earth, SeaSports, Cas.	Heraklio, WindSurf., Cas.	Olympus, SnowB., Cas.
Greece, SeaSports, Cas.	Earth, SeaSki, Cas.	Olympus, SnowSki, Cas.	Finland, AllSports, Cas.
Greece, SeaSki, Cas.	Cefalonia, SeaSki, Cas.	Finland, SeaSports, Cas.	Finland, WinterSp., Cas.
Earth, WindSurf., Cas.	Greece, WindSurf., Cas.	Finland, SeaSki, Cas.	Finland, WindSurf., Cas.
Cefalonia, WindSurf., Cas.	Cefalonia, SeaSports, Cas.	Finland, SnowSki, Cas.	Finland, SnowB., Cas.
		Earth, WinterSp., Cas.	Greece, WinterSp., Cas.
		Earth, SnowB., Cas.	Greece, SnowB., Cas.
		Earth, SnowSki, Cas.	Greece, SnowSki, Cas.
		Olympus, SeaSports, Cas.	

Table 3: The Valid and Invalid compound terms of the example of Figure 1

and consider facets whose range is a set of *intervals*. Section 3.1 analyzes in detail intervals and shows that CTCA can be applied on interval-ranged symbolic variables as it is.

3.1 Interval-valued Facets

An interval, specifically a closed interval of reals, is any pair of real numbers $[a, b]$ such that $a \leq b$. We can define a partial order ($\leq$) over a set of intervals Φ as follows: $[a, b] \leq [c, d]$ iff $c \leq a$ and $b \leq d$. Clearly, this is the interval inclusion relation.

An interpretation of a set of intervals Φ over a set of objects Obj is any function $I : \Phi \rightarrow 2^{Obj}$. We can distinguish two different meanings to an interpretation I, the *universal* and the *existential* one:

(a) *Universal* meaning

Here, $o \in I(\phi)$ means that object o *appears* in the *whole* ϕ, or equivalently, that the entire interval ϕ applies to o.

For instance, in the context of a variable life we can have $\texttt{Aristotle} \in I_{life}([-384, 322])$ as Aristotle lived from 384 B.C until 322 A.C., or $\texttt{Einstein} \in I_{life}([1879, 1955])$. In the context of a variable shoeSizes we can have $\texttt{Timberland} \in I_{shoeSizes}([36, 44])$.

(b) *Existential* meaning

Here, $o \in I(\phi)$ means that object o *appears somewhere* within ϕ. For instance, in the context of a variable date we can have $\texttt{Easter} \in I_{date}([March, May])$, or in the context of a variable prices we can have $\texttt{HolidayInn} \in I_{prices}([70, 200])$.

For a logical reading, we can view symbolic variables as first order predicates and objects as constants. Under this view we could write the following correspondences:

$$\texttt{Aristotle} \in I_{life}([-384, 322]) \quad \Leftrightarrow \quad (\forall\, t \in [-384, 322])\ \ (life(t, \texttt{Aristotle}))$$

$$\texttt{Easter} \in I_{date}([March, May]) \quad \Leftrightarrow \quad (\exists\, t \in [March, May])\ \ (date(t, \texttt{Easter}))$$

3.1.1 (a) The Universal meaning

If I is an interpretation with universal meaning, it follows that if $o \in I(\phi)$ and $\phi' \leq \phi$ then o appears in the whole ϕ' too. This can be captured by the notion of universal model introduced next.

An interpretation of I of Φ is a *universal model* of $(\Phi, \leq)$ if for any $\phi, \phi' \in \Phi$, if $\phi \leq \phi'$ then $I(\phi) \supseteq I(\phi')$.

We can always extend an interpretation I of Φ to a universal model of $(\Phi, \leq)$, denoted by $\underline{I}$, as follows:

$$\underline{I}(\phi) = \bigcup\{\ I(\phi')\ |\phi \leq \phi'\}$$

Given two interpretations I, I' of Φ, we call I less than or equal to I', and we write $I \sqsubseteq I'$, if $I(\phi) \subseteq I'(\phi)$ for each interval $\phi \in \Phi$. Clearly, $\underline{I}$ is the minimal universal model of $(\Phi, \leq)$ that is greater than I with respect to $\sqsubseteq$. Figure 2.(a) shows an interpretation I and Figure 2.(b) shows its extension to the universal model $\underline{I}$.

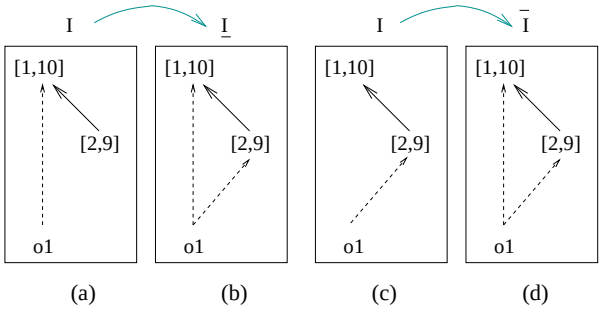

Figure 2: Interpretations and their extensions to universal and existential models

3.1.2 (b) The Existential meaning

If I is an interpretation with existential meaning, it follows that if $o \in I(\phi)$ and $\phi \leq \phi'$ then o also "appears" somewhere within ϕ'. This can be captured by the notion of existential model introduced next.

An interpretation of I of Φ is an *existential model* of $(\Phi, \leq)$ if for any $\phi, \phi' \in \Phi$, if $\phi \leq \phi'$ then $I(\phi) \subseteq I(\phi')$.

We can always extend an interpretation I of Φ to an existential model of $(\Phi, \leq)$, denoted by $\bar{I}$, as follows:

$$\bar{I}(\phi) = \bigcup\{\ I(\phi')\ |\phi' \leq \phi\}$$

Clearly, $\bar{I}$ is the minimal existential model of $(\Phi, \leq)$ that is greater than I with respect to $\sqsubseteq$. Figure 2.(c) shows an interpretation I and Figure 2.(d) shows its extension to the existential model $\bar{I}$.

3.1.3 Queries over Interval-based Sources

We shall call, interval-based source, for short source, any triple $(\Phi, \leq, I)$ where Φ is a set of intervals, $\leq$ is the inclusion relation over Φ and I is an interpretation of Φ.

We can query a source $(\Phi, \leq, I)$ in order to find objects that satisfy certain properties.

Let us now introduce a basic query language. A query is any string derived by the following grammar, $q ::= U[a,b] \mid E[a,b]$, where $[a,b]$ is an interval. We will denote by Q the set of all queries. The queries of the form $U[a,b]$ are called *universal queries*, while the queries of the form $E[a,b]$ are called *existential queries*. Roughly, a universal query $U[a,b]$ seeks for objects that appear in the whole interval $[a,b]$, while an existential query $E[a,b]$ seeks for objects that appear somewhere in the interval $[a,b]$.

Let's now define query answering. Of course the answer of queries over a source $(\Phi, \leq, I)$ depends on the meaning of I. Below we discuss query answering for each possible case.

- universal queries over interpretations with universal meaning

$$ans(U[a,b]) = \underline{I}([a,b]) = \bigcup\{\ I(\phi) \mid [a,b] \leq \phi\}$$

- existential queries over interpretations with existential meaning

$$ans(E[a,b]) = \bar{I}([a,b]) = \bigcup\{\ I(\phi) \mid \phi \leq [a,b]\}$$

- universal queries over interpretations with existential meaning

  The answers of these queries are always empty. This is because from premises of the form $\exists\ X\ P(X)$ we cannot yield conclusions of the form $\forall\ X\ P(X)$, unless our intervals are points but this is a special case of limited practical interest.

- existential queries over interpretations with universal meaning

  Let us first introduce some auxiliary definitions.

  Two intervals $[a,b]$ and $[c,d]$ are *overlapping*, if $c \leq b$. The intersection of two intervals $[a,b]$ and $[c,d]$, denoted by $[a,b] \cap [c,d]$, is defined as: $[a,b] \cap [c,d] = [c,b]$ if they are overlapping, and [] otherwise.

  One can easily see that if q is an existential query, then the answer of q (over a universal interpretation) is given by:

$$ans(E[a,b]) = \bigcup\{I(\phi) \mid \phi \text{ overlaps } [a,b]\}$$

A special case is discussed next.

An interval taxonomy $(\Phi, \leq)$ is *intersection-complete* (or closed on intersection) if for each $\phi, \phi' \in \Phi$, if $\phi \cap \phi' \neq []$ then $\phi \cap \phi' \in \Phi$.

If $(\Phi \cup \{[a,b]\}, \leq)$ is intersection complete, then it holds

$$\{\phi \mid \phi \text{ overlaps } [a,b]\} = \{\phi \mid \phi \leq [a,b]\}$$

for any interval $[a,b]$. It follows that in this case the answer of an existential query q (over a universal interpretation) is given by: $ans(E[a,b]) = \bar{I}([a,b]) = \bigcup\{I(\phi) \mid \phi \leq [a,b]\}$. i.e. it coincides with the query answering method for the case of existential queries over existential interpretations.

Note that we can always extend an interval taxonomy Φ to an intersection-complete taxonomy by adding to it the interval intersections that are missing (and of course adding the appropriate relationships).

3.1.4 Interval Validity and Symbolic Data Tables

Assume a source $(\Phi, \leq, I)$. It is natural to consider that an interval ϕ is *meaningful* (w.r.t. the source) if there are objects that appear somewhere in ϕ. Consequently, validity has an existential meaning. So, an interval $[a,b]$ is meaningful if and only if $ans(E[a,b]) \neq \emptyset$. Summarizing, if I has universal meaning then

$$valid([a,b]) \Leftrightarrow ans(E[a,b]) \neq \emptyset \Leftrightarrow \bigcup\{I(\phi) \mid \phi \text{ overlaps } [a,b]\} \neq \emptyset$$

If, on the other hand, I has an existential meaning, then

$$valid([a,b]) \Leftrightarrow ans(E[a,b]) \neq \emptyset \Leftrightarrow \bigcup\{\ I(\phi) \mid \phi \leq [a,b]\} \neq \emptyset$$

Symbolic Data Tables are mainly derived from summarizing Data Tables. The key observation here is that although the original Data Table may contain intervals with a universal meaning, their summarization results in intervals with existential meaning. It is clear that the result of summarizing a set of intervals with universal meaning cannot have a universal meaning. On the other hand, we can always summarize a set of universal intervals to one existential. For example the intervals $[5,40]$, $[20,50]$ and $[40,100]$ can be summarized by the interval $[5,100]$. Similarly, the intervals $[5,40]$ and $[100,200]$ can be summarized to $[5,200]$.

Thus it is quite natural to assume that all intervals of a Symbolic Data Table have an existential meaning. Consequently, CTCA applies on intervals as it is because the premise of CTCA, namely, if s is valid and $s \leq s'$ then s' is valid too, is true for intervals with existential interpretations.

4 The Technique

Roughly, a symbolic data table with k columns and n rows can be compressed in three steps:

Step 1

At first, we compute the range of each symbolic variable (i.e. the distinct values that appear in the corresponding column), we store it, and we identify what kind of ordering is appropriate for it. For the last, we can distinguish the following cases:

- *single-valued taxonomic variables*

  In this case the range of a variable is a set of categories that are a-priori partially ordered (i.e. a taxonomy) according to available domain knowledge. Here it is enough to store only the transitive reduction of the taxonomic ordering. For example, see Figure 3.(a).

- *multi-valued quantitative or multi-valued categorical variables*

  In this case the range of a variable is a set R of subsets of a set D (i.e. $R \subseteq \mathcal{P}(D)$ where $\mathcal{P}(D)$ denotes the powerset of D). Here we have the partially ordered set $(R, \subseteq)$ and its dual $(R, \supseteq)$. It is the latter that is appropriate for our case, specifically for any $s, s' \in R$ we consider that $s \leq s'$ if and only if $s \supseteq s'$. In this case we only have to store R as here the ordering relation corresponds to the relation $\supseteq$ which can be deduced algorithmically (for any two sets s and s' we can check whether $s \supseteq s'$). For example, see (b) and (c) of Figure 3.

- *interval-valued variables*

  Here the range of a variable is a set of intervals Φ. In this case we only have to store Φ because again the ordering relation, i.e. the interval inclusion relation, can be deduced (recall Section 3.1). For example, see (d) of Figure 3.

- *multi-valued taxonomic variables*

In this case the range of a variable is a set R of subsets of a a-priori partially ordered domain D. Here it is enough to store the elements of R and of course the taxonomic structure of D. For example, Figure 3.(e) shows the ordering (actually the compound ordering) of the set $R = \{\{SeaSports\}, \{SeaSki\}, \{SeaSports, Sauna\}, \{SeaSki, Sauna\}\}$ assuming the partially ordered domain of Figure 3.(a).

Remark: For this kind of symbolic variables the mining algorithms (that are described in Step 2 below) will employ a self-product operation.

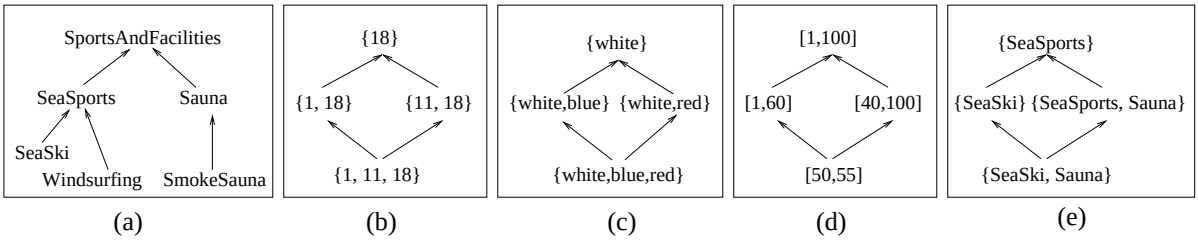

Figure 3: Orderings over the range of symbolic variables

Note that the disjointness of facet terminologies can be implemented in practise by prefixing each value of the range of a variable by the variable name.

Step 2

Subsequently, we can run one of the algorithms described in [13, 14] that *mine* an expression of CTCA that describes exactly the rows of our table.

Using the notations of the previous section, the objective of these algorithms is to find an expression e of CTCA such that

$$S_e = V = \{s \in \mathcal{P}(\mathcal{T}) \mid \bar{I}(s) \neq \emptyset\}$$

where if $s = \{t_1, ..., t_k\}$ then $I(s) = I(t_1) \cap ... \cap I(t_k)$. Paper [13] gives the algorithms for two straightforward methods for extracting a plus-product and a minus-product expression and an exhaustive algorithm for finding the *shortest* (i.e. the most space economical) expression. The latter works as follows. At first it generates the parse trees of all possible well-formed expressions over $T_1, ...T_k$. For example, Figure 4 shows the parse trees over 3 facets A, B, C. In this figure a dotted "X" marks the parse that can be excluded from the candidates, because the size of their corresponding expression can never be smaller than that of the expressions of the rest parse trees. Subsequently, for every operation of each parse tree, the algorithm specifies the P/N parameters so as to hold $S_e = V$, and it measures the storage space of e. The expression with

the least storage space is finally returned. The computational complexity of the algorithm is significantly higher, however the returned expressions can have comparatively very low storage space requirements, thanks to the closed-world assumptions of CTCA.

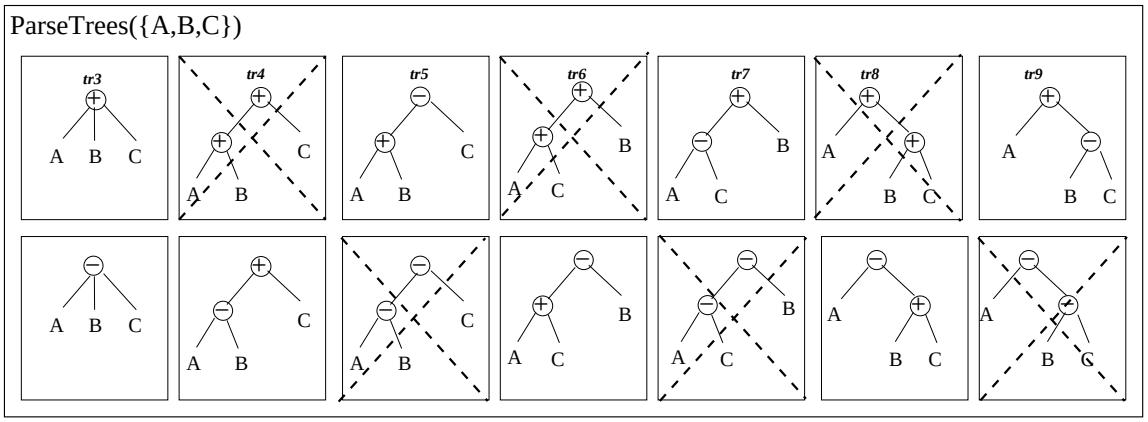

Figure 4: Parse trees

Step 3

Finally, we store the mined expression and its parameters (e.g. in a relational database as it has been done in FASTAXON [17]).

◇

After the above three-step process we can delete the symbolic data table and keep stored only (a) the ranges of the symbolic variables, (b) the ordering over these ranges (as described in Step 1), and (c) the mined expression e. Now suppose that we want to check whether an arbitrary tuple s (over the domain of our variables) exists in the table. We do not have to restore the initial table in order to answer this question. Instead, we run the algorithm described in [16] which takes as input a faceted taxonomy, an expression e and a compound term s and decides in polynomial time whether $s \in S_e$.

Another remark that should be mentioned here is that it is also possible to *browse* the symbolic table without having to reconstruct it. Specifically, by the faceted taxonomy $\mathcal{F}$ and the expression e we can derive dynamically a navigation tree that allows browsing all compound terms in S_e using the algorithm described in [16] that has been implemented in FASTAXON [17].

Of course, at any time we could run a (quite simple) algorithm for reconstructing the symbolic data table at its original form.

5 Illustrative Examples

This section presents a small number of intuitive examples for demonstrating the potential of CTCA for the problem at hand.

Example 1.

Consider that we have two variables A and B. The variable A ranges over the set $\{a_1, a_2, a_3\}$ and assume that this set is ordered according to a taxonomic relation (subsumption) as follows: $a_3 \leq a_2 \leq a_1$. Now consider the following table

A	B
a_1	b_1
a_2	b_1
a_3	b_1

The rows of this table can be described by the expression $e = A \oplus_P B$ where $P = \{\{a_3, b_1\}\}$. One can easily see that $S_e = \{\{a_1, b_1\}, \{a_2, b_1\}, \{a_3, b_1\}\}$.

Alternatively, they can be described by the expression $e' = A \ominus_N B$ where $N = \emptyset$ as $A \ominus_\emptyset B = A \oplus B = \{\{a_1, b_1\}, \{a_2, b_1\}, \{a_3, b_1\}\}$.

Example 2.

Now assume that the range of A is the taxonomy $(\{a_1, a_2, a_3, a_4\}, \{a_2 \leq a_1, a_3 \leq a_2, a_4 \leq a_2\})$, the range of B is the taxonomy $(\{b_1, b_2\}, \{b_2 \leq b_1\})$ and that we have the following table:

A	B
$\{a_3, a_4\}$	b_1
a_2	b_2
a_1	b_1
a_1	b_2
a_2	b_1
a_3	b_1

Here, and in order to describe the set $\{a_3, a_4\}$, we are obliged to use a self-product operation over A. We can describe the rows of this table by any of the above three expressions:

- $e_1 = (\overset{*}{\oplus}_{P1}(A)) \oplus_{P2} (B)$ where $P1 = \{\{a_3, a_4\}\}$ and $P2 = \{\{a_3, a_4, b_1\}, \{a_2, b_2\}\}$
- $e_2 = (\overset{*}{\ominus}_{N1}(A)) \oplus_{P2} (B)$ where $N1 = \emptyset$ and $P2 = \{\{a_3, a_4, b_1\}, \{a_2, b_2\}\}$.

- $e_3 = (\overset{*}{\ominus}_{N1}\ (A))\ \ominus_{N2}\ (B)$ where $N1 = \emptyset$ and $N2 = \{\{a_3, a_4, b_2\}\}$.

Clearly, e_3 is the most space economical expression as it requires us to keep stored only one compound term that consists of three single terms.

Example 3.

Assume that we have the following table with information about hotels:

Id	Location	Prices
H1	Heraklion	[30,50]
H2	Lixouri	[33,40]
H3	Heraklion	[25,300]
H4	Heraklion	[33,40]

For notational simplicity we shall use A for Location and B for Prices. The above table (by ignoring the first column) can be represented by the expression $e = A \ominus_{N_1} B$ where $N_1 = \emptyset$ as all combinations between the domain of these two variables are valid (appear or are semantically inferred from those that appear). Specifically, although Lixouri does not co-appear in the table with neither [30,50] nor with [25, 300], these combination are valid because since there is a hotel at Lixouri with rates [33,40], it is true that we can find a hotel at Lixouri at [30,50] or [25,300] Euros.

Example 4.

Let us now modify one cell of the above table:

Id	Location	Prices
H1	Heraklion	[30,50]
H2	Lixouri	**[30,50]**
H3	Heraklion	[25,300]
H4	Heraklion	[33,40]

This table can be represented by the expression $e_4 = A \ominus_{N_4} B$ where $N_4 = \{\{Lixouri, [33, 40]\}\}$.

Example 5.

Let us now add one more row and one more column to the table of the previous example

Id	Location	Prices	SportsAndFacilities
H1	Heraklion	[30,50]	SeaSki, Sauna
H2	Lixouri	[30,50]	
H3	Heraklion	[25,300]	WindSurfing, SeaSki, Sauna
H4	Heraklion	[33,40]	
H5	Helsinki	[33,40]	SmokeSauna

Let C denote the variable SportsAndFacilities and let the range of C be organized as shown in Figure 5. Let's now try finding the expression that describes this table. The "subtable" over the variables A and B is described by the expression e_4 as we saw earlier in Example 4. Now the range of variable C can be expressed using a self-product operation, specifically by $e_C = \overset{*}{\oplus}_{P_C}(C)$ where $P_C = \{\{WindSurfing, SeaSki, Sauna\}\}$. Note that if SmokeSauna did not belong to the range of C then we would have defined e_C as follows: $e_C = \overset{*}{\ominus}_{N_C}(C)$ with $N_C = \emptyset$.

In order to represent the whole table we have to combine e_4 and e_C. This can be obtained as: $e_3 = e_4 \oplus_{P_3} e_C$ where

$P_3 = \{\{Heraklion, [25, 300], Lixouri, \{WindSurfing, SeaSki, Sauna\}\}$

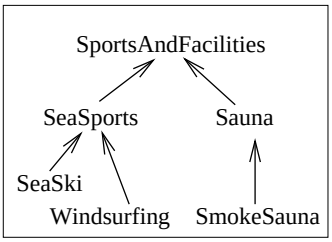

Figure 5: The range of the symbolic variable SportsAndFacilities

Summarizing, CTCA can indeed compact a symbolic data table and can yield to remarkably high compression ratios. One can easily guess that the more symbolic variables we have and the more numerous are the ranges of these variables, the higher compression ratio we can achieve with CTCA.

6 Compression Ratio

The objective of this section is compare the proposed technique with the existing compression algorithms and to discuss the compression ratio that can be achieved with the proposed

technique.

Commonly, lossless compression algorithms are efficient for files that contain lots of repetitive data, i.e. sequences of data that occur more than once. Such algorithms range from the simple RLE (Run Length Encoding) algorithm, to statistical algorithms like Huffman compression [6], and to dictionary based algorithms like LZ77 [19], its variant called Gzip, and LZ78 [20]. Roughly, statistical algorithms assign a smaller number of bits to symbols with higher probability of appearance, while dictionary based algorithms substitute a sequence of symbols by a pointer to a previous occurrence of that sequence.

The key difference of the technique proposed in this paper is that it exploits the semantics of the data (i.e. its mathematic structure), so it can reduce the size of a symbolic data table even if the table has not repetitive data. It is evident that this kind of compression cannot be obtained by the classical (general purpose) compression algorithms mentioned in the previous paragraph. However, the classical compression algorithms could be applied as a post-processing step in order to further reduce the storage space. For instance, all terms of T_i's can be stored in a dictionary form, i.e. a code can be assigned to each term and these codes can then be used (instead of the terms) for storing the P/N parameters.

Let us now discuss the compression ratio. Let $\{(T_1, \leq_1), ...(T_k, \leq_k)\}$ be a faceted taxonomy and $T = T_1 \cup ... \cup T_k$. We shall confine ourselves to the special case where the set of valid compound terms V is not any subset of $\mathcal{P}(T)$ but comprises compound terms that contain exactly one term from each facet, i.e. $V \subseteq C = T_1 \times \ldots \times T_k$. Of course, the results of this study could be generalized analogously in order to capture the general case.

Now suppose an expression e such that $S_e = V$. Our objective is to compare the size of V with the size of e in various cases and under different assumptions. Specifically, we can define as *compression ratio* the size of e expressed as a function of the size of the uncompressed symbolic data table, i.e. as a function of the size of V.

In our study we will consider the *number of terms* as the unit for measuring size. For instance, $size(T_1) = |T_1|$, $size(\leq_1) = 2\ |\leq_1|$ (e.g. $size(\{a \leq b, b \leq c\}) = 4$) and $size(\{a, b\}) = 2$. Consequently we can define the size of a symbolic data table as

$$size(V) = \sum_{s \in V} |s|$$

Let $\mathcal{P}$ and $\mathcal{N}$ denote the set of all P and N parameters of an expression e. We can define the

size of an expression e as follows:

$$\begin{aligned} size(e) &= size(dom(e)) + size(param(e)) \text{ , where} \\ size(dom(e)) &= \sum_{i=1}^{k} (|T_i| + 2\ |\leq_i|) \\ size(param(e)) &= \sum_{P \in \mathcal{P}} size(P) + \sum_{N \in \mathcal{N}} size(N) \\ size(P) &= \sum_{s \in P} |s| \ \text{ and } \ size(N) = \sum_{s \in N} |s| \end{aligned}$$

We can safely assume that all $\leq_1, \ldots, \leq_k$ can be inferred, hence we do not have to store them. Note that if additional domain knowledge is available in the form of a taxonomy, then this taxonomy will be already stored aside the symbolic data table, hence we do not have to include it in our study. So we can assume that $size(dom(e)) = \sum_{i=1}^{k} |T_i|$, reaching to the following definition of size:

$$size(e) = \sum_{i=1}^{k} |T_i| + \sum_{P \in \mathcal{P}} size(P) + \sum_{N \in \mathcal{N}} size(N)$$

Let also define the complement of V, denoted by V^c as $V^c = C - V$.

Let's now elaborate the case where the expression e is derived by the straightforward mining algorithm. In this case e is either one $\oplus$ or a $\ominus$ operation over $T_1, ..., T_k$. Note that this algorithm is efficient but does not return the shortest in size expression.

Prop. 1 $size(param(e)) \leq min(size(V), size(V^c))$. □

The proof is straightforward. If $size(V) > size(V^c)$ then the algorithm returns an expression with one minus-product operation, so $size(param(e)) = size(N) \leq size(V^c)$. In the opposite case, $size(param(e)) = size(P) \leq size(V)$.

Below we shall use $min_{\preceq}(S)$ (resp. $max_{\preceq}(S)$) to denote the minimal (resp. maximal) elements of a set of compound terms S with respect to compound ordering $\preceq$.

Prop. 2 $size(param(e)) \leq min(size(min_{\preceq}(V)), size(max_{\preceq}(V^c)))$. □

This proposition holds because the mining algorithm exploits the property that we can replace the N parameter of a minus-product by $max_{\preceq}(N)$, and we can replace the P parameter of a plus-product by $min_{\preceq}(P)$ and get an equivalent expression in both cases.

Prop. 3 (Best/Worst Compression Ratio)
In the best case we have $size(e) = k\sqrt[k]{size(V)}$, while in the worst case $size(e) \geq size(V)$. □

We reach the best compression ratio when all compound terms are valid, i.e. when $V = T_1 \times \ldots \times T_k$ (i.e. $V = C, V^c = \emptyset$). In this case we have $size(V) = \Pi_{i=1}^{k} |T_i|$. Now $size(e) = size(dom(e)) + size(param(e)) = \sum_{i=1}^{k} |T_i| + 0$. The size of parameters is 0 because e can be a minus product with an empty parameter set N (this also follows from Prop. 1 as here we have $V^c = \emptyset$). Now if we assume that all T_i have the same cardinality, denoted by $|T_o|$, then $size(V) = \Pi_{i=1}^{k} |T_i| = |T_o|^k$ and $size(e) = \sum_{i=1}^{k} |T_i| = k|T_o|$. From the above two it follows that

$$size(e) = k \sqrt[k]{size(V)}$$

and clearly this is the best compression ratio that we can achieve. It is evident that the bigger the number k is (i.e. the more variables we have), the higher compression ratio we obtain.

The worse compression ratio occurs when $\leq_1 = \ldots \leq_k = \emptyset$ and each value at each cell of the symbolic data table appears only once in the symbolic data table. In this case, $size(V) = \sum_{i=1}^{k} |T_i|$. Note that here $size(dom(e)) = \sum_{i=1}^{k} |T_i| = size(V)$. From this it is already clear that in this case we cannot gain anything by compressing. Note that in this case $size(V^c)$ is much bigger than $size(V)$ as $size(V^c) = \Pi_{i=1}^{k} |T_i| - 1$. This means that the straightforward mining algorithm will return a plus-product operation. However the parameter P of this operation has to contain every $s \in V$, so it will be $size(P) = \sum_{i=1}^{k} |T_i|$. So in this case we do not gain anything by compressing. In particular, if we store both $dom(e)$ and $param(e)$, then it will be $size(e) \geq size(V)$. However, we can identify easily such cases so as to store only $param(e)$ (and thus have $size(e) = size(V)$), or to avoid compressing.

Let's now discuss the case where the expression e is derived by the shortest expression mining algorithm. This algorithm returns the shortest possible expression but it's computational complexity is much higher. Here e can have several $\oplus$ or $\ominus$ operations. The range-restricted closed worlds assumptions of these operations make hard the estimation of the compression ratio. This is an issue that is worth further research. Of course, the best/worse compression ratios that were described in the previous section apply to this case as well.

7 Epilogue

Although symbolic data tables summarize huge sets of data they can still become very large in size. This paper proposes a method for compressing a symbolic data table using the recently emerged Compound Term Composition Algebra (CTCA). One advantage of CTCA for the

problem at hand is that the closed world hypotheses of its operations (described analytically at [15]) can lead to a remarkably high compression ratio. Another remark that have to be mentioned here is that the functionality offered by CTCA cannot be obtained by using a classical logic-based formalism, like Description Logics, as it was shown in [15]. At last, but not least, this paper identified the analogies between symbolic data tables and faceted taxonomies (and CTCA) in order to act as a two-way canal between the two communities. An issue for further research is the characterization of the proposed approach according to Kolmogorov's complexity and the extension of this method for frequency-valued symbolic variables.

Acknowledgement

Many thanks to Tonia Dellaporta for the fruitful discussions on this issue, as well as to Monique Noirhomme-Fraiture and to Anne de Baenst-Vandenbroucke for providing me with very useful material for Symbolic Data Analysis.

References

[1] "XFML: eXchangeable Faceted Metadata Language". http://www.xfml.org.

[2] "XFML+CAMEL: Compound term composition Algebraically-Motivated Expression Language". http://www.csi.forth.gr/markup/xfml+camel.

[3] H. H. Bock and E. Diday. *Analysis of Symbolic Data*. Springer-Verlag, 2000. ISBN: 3-540-66619-2.

[4] Edwin Diday. "An Introduction to Symbolic Data Analysis and the Sodas Software". *Journal of Symbolic Data Analysis*, 0(0), 2002. ISSN 1723-5081.

[5] Elizabeth B. Duncan. "A Faceted Approach to Hypertext". In Ray McAleese, editor, *HYPERTEXT: theory into practice, BSP*, pages 157–163, 1989.

[6] D. Huffman. "A Method for the Construction of Minimum-Redundancy Codes". *Proc. of the I.R.E*, 40(9):1090–1101, 1952.

[7] P. H. Lindsay and D. A. Norman. *Human Information Processing*. Academic press, New York, 1977.

[8] Amanda Maple. "Faceted Access: A Review of the Literature", 1995. http://theme.music.indiana.edu/tech_s/mla/facacc.rev.

[9] Ruben Prieto-Diaz. "Classification of Reusable Modules". In *Software Reusability. Volume I*, chapter 4, pages 99–123. acm press, 1989.

[10] Ruben Prieto-Diaz. "Implementing Faceted Classification for Software Reuse". *Communications of the ACM*, 34(5):88–97, 1991.

[11] S. R. Ranganathan. "The Colon Classification". In Susan Artandi, editor, *Vol IV of the Rutgers Series on Systems for the Intellectual Organization of Information*. New Brunswick, NJ: Graduate School of Library Science, Rutgers University, 1965.

[12] Yannis Tzitzikas. "An Algebraic Method for Compressing Very Large Symbolic Data Tables". In *Procs. of the Workshop on Symbolic and Spatial Data Analysis of ECML/PKDD 2004*, Pisa, Italy, September 2004.

[13] Yannis Tzitzikas and Anastasia Analyti. "Mining the Meaningful Compound Terms from Materialized Faceted Taxonomies". In *Procs. of the 3rd Intern. Conference on Ontologies, Databases and Applications of Semantics for Large Scale Information Systems, ODBASE'2004*, pages 873–890, Larnaca, Cyprus, October 2004.

[14] Yannis Tzitzikas and Anastasia Analyti. "Mining the Meaningful Term Conjunctions from Materialized Faceted Taxonomies: Algorithms and Complexity", 2005. Knowledge and Information Systems Journal (accepted for publication).

[15] Yannis Tzitzikas, Anastasia Analyti, and Nicolas Spyratos. "Compound Terms Composition Algebra: The Semantics". *LNCS Journal on Data Semantics*, 2:58–84, 2005.

[16] Yannis Tzitzikas, Anastasia Analyti, Nicolas Spyratos, and Panos Constantopoulos. "An Algebraic Approach for Specifying Compound Terms in Faceted Taxonomies". In *Information Modelling and Knowledge Bases XV, 13th European-Japanese Conference on Information Modelling and Knowledge Bases, EJC'03*, pages 67–87. IOS Press, 2004.

[17] Yannis Tzitzikas, Raimo Launonen, Mika Hakkarainen, Pekka Kohonen, Tero Leppanen, Esko Simpanen, Hannu Tornroos, Pekka Uusitalo, and Pentti Vanska. "FASTAXON: A system for FAST (and Faceted) TAXONomy design.". In *Proceedings of 23th Int. Conf.*

on Conceptual Modeling, ER'2004, Shanghai, China, November 2004. (an on-line demo is available at http://fastaxon.erve.vtt.fi/).

[18] B. C. Vickery. "Knowledge Representation: A Brief Review". *Journal of Documentation*, 42(3):145–159, 1986.

[19] J. Ziv and A. Lempel. "A Universal Algorithm for Sequential Data Compression". *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.

[20] J. Ziv and A. Lempel. "Compression of Individual Sequences via Variable-rate Coding". *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.