Updates and Revision in Faceted Taxonomies and CTCA Expressions [1]

Yannis Tzitzikas

Computer Science Department, University of Crete (Greece),
Information Systems Lab, Institute of Computer Science, FORTH-ICS (Greece)
Email : tzitzik@csd.uoc.gr

Abstract

A faceted taxonomy is a forest of taxonomies each describing the application domain from a different (preferably orthogonal) point of view. CTCA is an algebra that allows specifying the set of meaningful compound terms (meaningful conjunctions of terms) over a faceted taxonomy in a flexible and efficient manner. However, taxonomy updates may turn a CTCA expression e not well-formed and may turn the compound terms specified by e to no longer reflect the domain knowledge originally expressed in e. This paper shows how we can revise e after a taxonomy update and reach an expression e' that is both well-formed and whose semantics (compound terms defined) is as close as possible to the semantics of the original expression e before the update. Various cases are analyzed and the revising algorithms are given. The proposed technique can enhance the robustness and usability of systems that are based on CTCA and allows optimizing several other tasks where CTCA can be used (including mining and compressing).

Keywords: faceted taxonomies, updates, knowledge revision.

1 Introduction

Suppose that we want to build a catalog of traditional recipes from all over the world and for this purpose we decide to define facets like *Ingredients*, *LocationOfOrigin* and *CookingStyle* as shown in Figure 1. Notice that several combinations of terms are *invalid*, even in this very small domain. For example, the compound term $\{Truffle$ (from *Ingredients*), *Greece* (from *Location*)$\}$ is invalid as it is impossible to find truffle in Greece, hence there cannot be a traditional Greek recipe that contains truffle. For the same reason the compound term $\{Roquefort$ (from *Ingredients*), *Greece* (from *Location*)$\}$ is invalid as well as the compound term $\{Feta$ (from *Ingredients*), $France$ (from *Location*)$\}$. Moreover, the compound term $\{Wok$ (from *CookingStyle*), *Europe* (from *Location*)$\}$ is invalid because wok is used in Asia and not in Europe. According to these assumptions, the partition of compound terms to the set of *valid* (meaningful) compound terms and *invalid* (meaningless) compound terms is shown in Table 7 found at Appendix A.

CTCA (Compound Term Composition Algebra) [23, 24] is an algebra that allows specifying the set of meaningful compound terms over a faceted taxonomy in a flexible and efficient manner. By using CTCA the designer provides only a small set of valid or invalid compound terms and from these sets other valid and invalid compound terms are inferred. Having partitioned the set of compound terms to the set of valid and invalid is quite important as this can significantly aid the task of indexing

[1] Alternative title: "Revising (or Preserving) the Set of (CTCA-specified) Valid Term Conjunctions after Taxonomy Updates"

objects according to the faceted taxonomy, and the task of browsing a collection of objects that are indexed according to a faceted taxonomy (for more see [25]).

Let F be a faceted taxonomy, i.e. a forest of taxonomies $(\mathcal{T}_1, \leq_1), \ldots, (\mathcal{T}_k, \leq_k)$, and let $\mathcal{T}=\mathcal{T}_1 \cup \ldots \cup \mathcal{T}_k$. Each expression e of CTCA specifies a set S_e^F of valid (i.e. meaningful) compound terms (conjunctions of terms) over $\mathcal{T}$. So an expression e actually defines the partition $(S_e^F, \mathcal{P}(\mathcal{T}) - S_e^F)$ where $\mathcal{P}(\mathcal{T})$ denotes the powerset of $\mathcal{T}$. For example, the partition shown in Table 7, can be specified using the following very short CTCA expression:

$$e = (Ingredients \oplus_P LocationOfOrigin) \ominus_N CookingStyle$$

with the following P and N parameters:

$$\begin{aligned} P &= \{\{Feta, Greece\}, \{Roquefort, France\}, \{Truffle, France\}, \{Truffle, Italy\}, \\ &\quad \{Cheese, Italy\}, \{Cheese, Japan\}\} \\ N &= \{\{Europe, Wok\}\} \end{aligned}$$

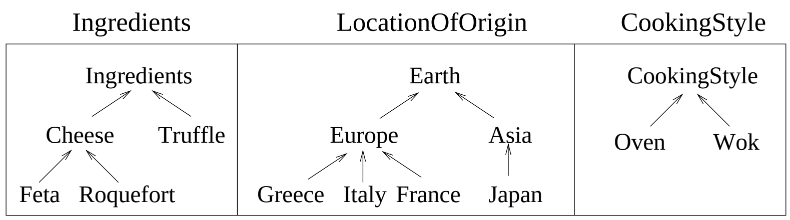

Figure 1: A faceted taxonomy for indexing traditional recipes

Table 1 shows the partition defined by a subexpression of the above expression, specifically by the expression $Ingredients \oplus_P LocationOfOrigin$. This expression partitions the set of compound terms over the first two facets of Figure 1.

Valid		
Feta, Greece	Feta, Europe	Feta, Earth
Cheese, Greece	Cheese, Europe	Cheese, Earth
Ingredients, Greece	Ingredients, Europe	Ingredients, Earth
Roquefort, France	Roquefort, Europe	Roquefort, Earth
Cheese, France	Ingredients, France	Truffle, France
Truffle, Europe	Truffle, Earth	Truffle, Italy
Cheese, Italy	Ingredients, Italy	Cheese, Japan
Cheese, Asia	Ingredients, Japan	Ingredients, Asia

Invalid	
Feta, Italy	Feta, France
Feta, Japan	Feta, Asia
Roquefort, Greece	Roquefort, Italy
Roquefort, Japan	Roquefort, Asia
Truffle, Greece	Truffle, Japan
Truffle, Asia	

Table 1: The partition defined by the expression $Ingredients \oplus_P LocationOfOrigin$

As the facet *Ingredients* has 5 terms and the facet *LocationOfOrigin* has 7 terms, the number of compound terms that contain exactly 1 term from each facet is 5*7 = 35. This table contains 24 valid and 11 invalid compound terms, thus 35 in total.

However, an update operation u_F on F (resulting to a faceted taxonomy F') may turn the expression e obsolete (i.e. not well-formed), or it may make the derived compound terminology $S_e^{F'}$ to no longer reflect the desire of the designer, i.e. it may no longer reflect the domain knowledge that was expressed in e. For example, if a term $t \in \mathcal{T}$ is deleted, and t appears in a compound term in a parameter (P or N, for more see Section 2) of the expression e, then e will no longer be well-formed. In addition, the deletion of t may make several compound terms (that do not even contain t) to no longer belong to $S_e^{F'}$. It would be very useful if we could update automatically e to an expression e' that is (a) well-formed (w.r.t. F'), and (b) $S_{e'}^{F'}$ is as close to S_e^F as possible. We will call this problem *expression revision* after taxonomy update. Figure 2 describes graphically the interrelationships between F, F', e, e', S_e^F and $S_{e'}^{F'}$.

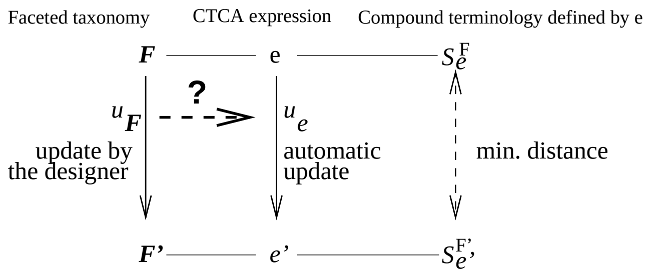

Figure 2: Expression revision after taxonomy update

Solving this problem would be very useful during the process of valid compound term specification, i.e. it can enhance the robustness and usability of systems that are based on CTCA, like FASTAXON [26]. In addition, as a CTCA expression can be also used for exchanging compactly the compound terms that are extensionally valid according to a materialized faceted taxonomy (using the mining algorithms presented in [22, 21]), this automation can be exploited in order to avoid reapplying these (computationally expensive) mining algorithms after an update of the faceted taxonomy. Moreover, as showed in [20, 19], CTCA can be used for compressing a Symbolic Data Table [4]. In this context, this automation can also be exploited in order to avoid recompressing a Symbolic Data Table after a small change on its contents.

The rest of this paper is organized as follows. Section 2 recalls the basics of the *Compound Term Composition Algebra* (CTCA), and Section 3 describes the taxonomy update operations that we consider. Subsequently, Section 4 defines formally the problem of *expression revision* after taxonomy update, and Section 5 gives a solution to this problem for each type of update operations. Finally, Section 7 concludes the paper and identifies issues for further research.

2 The Compound Term Composition Algebra

Faceted classification was suggested quite long ago by Ranganathan in the 1920s [17]. The adoption of faceted taxonomies is beneficial for Libraries [12], Software Repositories [13, 14], Web Catalogs or Web Sites [15], and other application domains, like biology[2]. Current interest in faceted taxonomies is also indicated by several recent or ongoing projects and the emergence of XFML [1](Core-eXchangeable Faceted Metadata Language), a markup language for applying the faceted classification paradigm on the Web. Other work on faceted classification includes [6, 27, 11] and the more recent [16, 18].

The *Compound Term Composition Algebra* (CTCA) is an algebra consisting of four basic algebraic operators (plus-product, minus-product, plus-self-product, and minus-self-product) which can be used for specifying the set of compound terms over a given faceted taxonomy that are *valid* (i.e. *meaningful*) in the application domain. From a "logical" point of view, we could say that CTCA is an algebra for specifying the "satisfiable" conjunctions of terms. The initial motivation for CTCA was to provide a well-founded method that is both flexible and economical (in terms of required input) and computationally efficient. One system based on CTCA has already been built [26], while other applications of CTCA are described in [20, 22, 21]. The semantics of CTCA differ from that of Description Logics (DL)[5] mainly because each operation of CTCA makes either a positive or a negative closed world assumption at its range, as it is shown in detail in [24]. Specifically, for a DL-based representation of an expression e, we have to convert either all plus-product operations to minus-products, or all minus-

[2] The *Gene Ontology* (http://www.geneontology.org/) is a faceted taxonomy for indexing gene products.

product operations to plus-products. Firstly, this does not allow a natural representation in DL, and secondly, in many cases the resulting DL representation of e has much more sentences (concept axioms or concept assertions) than the parameters of the expression e.

Table 2 below recalls in brief the basic notions and notations around taxonomies and faceted taxonomies that are used in this paper.

Name	Notation	Definition
terminology	$\mathcal{T}$	a finite 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	$F = \{F_1, ..., F_k\}$	$F_i = (\mathcal{T}_i, \leq_i)$, 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 over S	$\preceq$	Given $s, s' \in S$, $s \preceq s'$ iff $\forall t' \in s'\ \exists t \in s$ such that $t \leq t'$.
immediate broaders of t	$Br_{(1)}(t)$	the smaller terms that are greater than t (w.r.t $\leq$), i.e. $minimal_{\leq}(\{t' \in \mathcal{T} \mid t \leq t', t \neq t'\})$
immediate narrowers of t	$Nr_{(1)}(t)$	the bigger terms that are smaller than t (w.r.t $\leq$), i.e. $maximal_{\leq}(\{t' \in \mathcal{T} \mid t' \leq t, t \neq t'\})$
broaders of t	$Br(t)$	$\{t' \in \mathcal{T} \mid t \leq t'\}$
narrowers of t	$Nr(t)$	$\{t' \in \mathcal{T} \mid t' \leq t\}$
broaders of s	$Br(s)$	$\{s' \in P(\mathcal{T}) \mid s \preceq s'\}$
narrowers of s	$Nr(s)$	$\{s' \in P(\mathcal{T}) \mid s' \preceq s\}$
broaders of S	$Br(S)$	$\cup\{Br(s) \mid s \in S\}$
narrowers of S	$Nr(S)$	$\cup\{Nr(s) \mid s \in S\}$

Table 2: Notations

Some remarks about the taxonomies that we consider are in order.

Each cycle (formed by subsumption relationships) that may exist in a taxonomy, defines a class of equivalent terms (e.g. we can write $t \sim t'$ iff $t \leq t'$ and $t' \leq t$). However, and without loss of generality, we can hereafter consider that each $\leq$ is acyclic. In case the initial subsumption relation is cyclic, we can consider that $\leq$ denotes the subsumption relation over the classes of equivalence that are induced by the initial subsumption relation. So, we can safely assume that $\leq$ has the form of a directed acyclic graph (or a tree). Note that under this perspective, $\leq$ is also antisymmetric, so it is actually a partial order (and not just a preorder).

2.1 CTCA: Syntax and Semantics

Let $F = \{(\mathcal{T}_1, \leq_1), \ldots, (\mathcal{T}_k, \leq_k)\}$ be a faceted taxonomy and let $\mathcal{T}$=$\mathcal{T}_1 \cup \ldots \cup \mathcal{T}_k$. Each CTCA expression e specifies a compound terminology, i.e. a set of compound terms which we denote by S_e^F, or S_e for short (clearly, $S_e \subseteq \mathcal{P}(\mathcal{T})$). Syntactically, an expression e over F is defined according to the following grammar ($i = 1, ..., k$):

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

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 is viewed a singleton. In most of the cases, taxonomies are trees. The basic compound terminology of a tree-structured taxonomy $(\mathcal{T}_i, \leq_i)$ is defined as:

$$T_i = \{\{t\} \mid t \in \mathcal{T}_i\} \cup \{\emptyset\}$$

A definition that captures the general case (i.e. taxonomies that are not trees) follows:

$$T_i = \cup\{\ Br(\{t\}) \mid t \in \mathcal{T}_i\}$$

The motivation for this difference is that every individual term of a taxonomy is by default assumed that it is valid (meaningful), i.e. there are real-world objects (at least one) to which this term applies. It follows, that in the taxonomy C of Figure 3, the compound term $\{c2, c3\}$ should be considered as valid as it subsumes $\{c4\}$. This is captured by the above formula as $\{c2, c3\} \in Br(\{c4\})$.

Plus-products and *minus-products*, denoted by $\oplus_P$ and $\ominus_N$ respectively, have a parameter that is denoted by P (resp. N) which is a set of compound terms over $\mathcal{T}$. In a P parameter the designer puts valid compound terms, while in a N parameter the designer puts invalid compound terms. The exact definition of each operation of CTCA (also including two auxiliary operations, called *product* and *self-product*) is summarized in Table 3.

An expression e is *well formed* iff every facet T_i appears at most once, and every parameter set P or N of e is always subset of the corresponding set of *genuine compound terms*. Specifically, the genuine compound terms in the context of an operation $\oplus_P(e_1, ..., e_k)$ (or $\ominus_N(e_1, ..., e_k)$) is denoted by $G_{e_1,...,e_k}$ and it is defined as:

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

For example, the compound term $\{Truffle, Greece\}$ is a genuine compound term in the context of an operation $Ingredients \ominus_N LocationOfOrigin$, but not genuine in the context of the operation $(Ingredients \ominus_N LocationOfOrigin) \oplus CookingStyle$.

Now the set of genuine compound terms in the context of a self-product operation, is denoted by G_{T_i} and is defined as: $G_{T_i} = \overset{*}{\oplus} (T_i) - T_i$.

From Table 3, one can easily see that if e is a plus-product then S_e increases as P gets larger, while if e is a minus-product then S_e decreases as N gets larger. In that sense, minus-products are non-monotonic. However, as we have shown in [24], well-formed expressions have a monotonic behavior with respect to number of facets, meaning that the valid compound terms of a subexpression cannot be invalidated by an expression that contains it.

The algorithm $IsValid(e, s)$, given in [25], takes as input a (well-formed) expression e and a compound term s, and checks whether $s \in S_e$. This algorithm has polynomial time complexity, specifically $O(|\mathcal{T}|^3 * |\mathcal{P} \cup \mathcal{N}|)$, where $\mathcal{P}$ denotes the union of all P parameters and $\mathcal{N}$ denotes the union of all N parameters appearing in e. The pair $(S_e, \preceq)$ is called the *compound taxonomy* of e.

Operation	e	S_e
	T_i	$\{ \{t\} \mid t \in \mathcal{T}_i\} \cup \{\emptyset\}$
product	$e_1 \oplus ... \oplus e_n$	$\{ s_1 \cup ... \cup s_n \mid s_i \in S_{e_i}\}$
plus-product	$\oplus_P(e_1, ...e_n)$	$S_{e_1} \cup ... \cup S_{e_n} \ \cup \ Br(P)$
minus-product	$\ominus_N(e_1, ...e_n)$	$S_{e_1} \oplus ... \oplus S_{e_n} - Nr(N)$
self-product	$\overset{*}{\oplus} (T_i)$	$P(\mathcal{T}_i)$
plus-self-product	$\overset{*}{\oplus}_P \ (T_i)$	$T_i \cup Br(P)$
minus-self-product	$\overset{*}{\ominus}_N \ (T_i)$	$\overset{*}{\oplus} (T_i) - Nr(N)$

Table 3: The operations of the Compound Term Composition Algebra

For instance, Figure 3 shows a faceted taxonomy consisting of three facets, A, B and C. Some examples of compound terminologies that are defined by expressions of CTCA are given in Table 4 (the empty compound term $\{\emptyset\}$ is not shown, and we adopt the basic compound terminologies for trees, i.e. we do not show the $\{c2, c3\}$).

We also assume that each facet $(\mathcal{T}_i, \leq_i)$ is assigned a unique name, which we will denote by $nm(\mathcal{T}_i)$. Some extra notations that we shall use in the sequel follow:

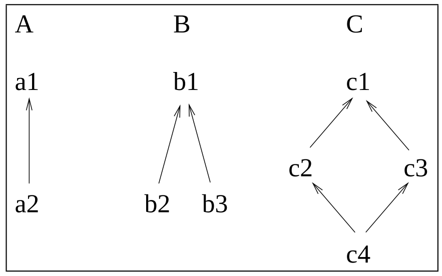

Figure 3: A faceted taxonomy consisting of three facets

e	S_e
$A \oplus_P B$, $P = \emptyset$	$\{\{a1\}, \{a2\}, \{b1\}, \{b2\}, \{b3\}\}$
$A \ominus_N B$, $N = \emptyset$	$\{\{a1\}, \{a2\}, \{b1\}, \{b2\}, \{b3\},$ $\{a1, b1\}, \{a1, b2\}, \{a1, b3\}, \{a2, b1\}, \{a2, b2\}, \{a2, b3\}\}$
$A \oplus_P B$, $P = \{\{a2, b1\}\}$	$\{\{a1\}, \{a2\}, \{b1\}, \{b2\}, \{b3\}, \{a1, b1\}, \{a2, b1\}\}$
$A \ominus_N B$, $N = \{\{a1, b2\}, \{a1, b3\}\}$	$\{\{a1\}, \{a2\}, \{b1\}, \{b2\}, \{b3\}, \{a1, b1\}, \{a2, b1\}\}$
$(A \ominus_N B) \oplus_P C$, $N = \{\{a2, b2\}\}$, $P = \{\{a1, b3, c1\}\}$	$\{\{a1\}, \{a2\}, \{b1\}, \{b2\}, \{b3\},$ $\{a1, b1\}, \{a1, b2\}, \{a1, b3\}, \{a2, b1\}, \{a2, b3\},$ $\{a1, b3, c1\}, \{a1, b1, c1\}, \{a1, c1\}, \{b3, c1\}, \{b1, c1\}\}$
$(A \oplus_P B) \ominus_N C$, $P = \{\{a1, b1\}\}$ $N = \{\{b3, c4\}\}$,	$\{\{a1\}, \{a2\}, \{b1\}, \{b2\}, \{b3\}, \{a1, b1\},$ $\{a1, c1\}, \{a1, c2\}, \{a1, c3\}, \{a1, c4\},$ $\{a2, c1\}, \{a2, c2\}, \{a2, c3\}, \{a2, c4\},$ $\{b1, c1\}, \{b1, c2\}, \{b1, c3\}, \{b1, c4\},$ $\{b2, c1\}, \{b2, c2\}, \{b2, c3\}, \{b2, c4\},$ $\{b3, c1\}, \{b3, c2\}, \{b3, c3\},$ $\{a1, b1, c1\}, \{a1, b1, c2\}, \{a1, b1, c3\}, \{a1, b1, c4\}\}$

Table 4: Some examples of CTCA-defined compound terminologies

- $f(t)$: the name of the facet of term t, i.e. if $t \in \mathcal{T}_i$ then $f(t) = nm(\mathcal{T}_i)$, e.g. $f(Feta) = Ingredients$.
- $f(e)$: the names of the facets that appear in e, e.g. $f((T_1 \oplus_P T_2) \ominus_N T_3) = \{nm(\mathcal{T}_1), nm(\mathcal{T}_2), nm(\mathcal{T}_3)\}$.
- $\pi_{f(e)}(s) = \{\ t \in s \mid f(t) \in f(e)\}$, this is the "projection" of s to the facets of e, e.g. $\pi_{f(A \oplus_P B)}(\{a1, b1, c3\}) = \{a1, b1\}$.

3 Taxonomy Updates

Here we discuss the taxonomy update operations that we consider. We consider two primitive update operations on subsumption relationships, namely:

- subsumption relationship deletion, denoted by $\texttt{delete}(t \leq t')$, and
- subsumption relationship addition, denoted by $\texttt{add}(t \leq t')$.

Before an operation $\texttt{delete}(t \leq t')$ we assume that the relationship $t \leq t'$ belongs to the transitive reduction (Hasse Diagram) of $\leq$. Now before an operation $\texttt{add}(t \leq t')$ we assume that the relationship $t \leq t'$ does not already exist in $\leq$. For instance, Figure 4 shows an example of a deletion and an addition of a subsumption relationship. We also assume that both t and t' belong to the same facet.

We also consider three update operations on terms:

- term renaming, denoted by $\texttt{rename}(t, t')$,

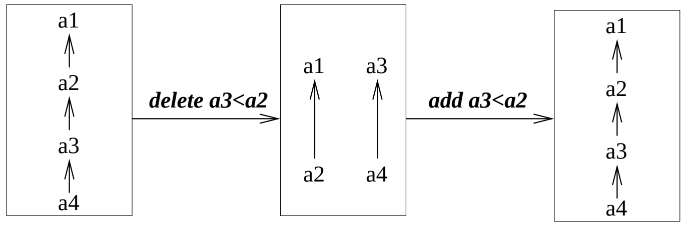

Figure 4: Deletion and addition of subsumption relationships

- term deletion, denoted by `delete`(t), and
- term addition, denoted by `add`(t).

Concerning the deletion of terms we consider that whenever a term t is deleted, all subsumption relationships in which t participates are deleted too. For example, the deletion of a_3 in Figure 5(a) will trigger the deletion of the following relationships $\{a_3 \leq a_1, a_3 \leq a_2, a_4 \leq a_3, a_5 \leq a_3\}$. However, as the relation $\leq$ is transitive, after the deletion of a_3 the taxonomy will be as shown in Figure 5(b). If however the transitive links of $\leq$ are not stored in the base, i.e. if only the transitive reduction of $\leq$ is stored, then whenever a term t is deleted, the immediate parent(s) of t should become parent(s) of all immediate children of t. Recall that $Br_{(1)}(t)$ denotes the set of all terms which immediately subsume t, and $Nr_{(1)}(t)$ denotes the set of all terms which are immediately subsumed by t. After deleting term t, for all $t' \in Nr_{(1)}(t)$ is holds $Br_{(1)}(t') \supseteq Br_{(1)}(t)$.

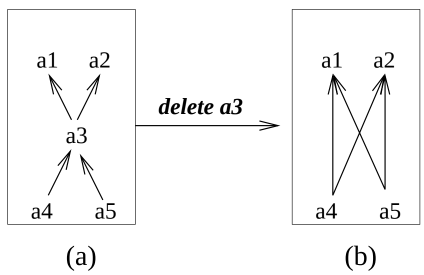

Figure 5: Term Deletion

Now we will introduce two auxiliary (composite) operations. Although they can be expressed in terms of the primitive update operations, we study them separately because they occur very frequently in practice, and because it is interesting to study what the designer wishes (concerning expression revision) after each of these operations. In particular, we consider the following two:

- term addition as leaf node, denoted by `addLeaf`(t, Par), where $Par \subseteq \mathcal{T}$

  This operation adds a new term t as leaf of the taxonomy, specifically t becomes child of every term in Par.

- term addition as intermediate node, denoted by `addIntermediate`(t, Chi, Par).

  This operation adds a new term t as intermediate term, specifically t becomes child of every term in Par and parent of every term in Chi.

Clearly, the first operation is a special case of the second, i.e. `addLeaf`$(t, Par) =$ `addIntermediate` $(t, \emptyset, Par)$. Now `addIntermediate`(t, Chi, Par) can be analyzed in the following sequence of updates: `add`(t), `add`$(t \leq p)$ for every $p \in Par$, and `add`$(c \leq t)$ for every $c \in Chi$.

If we assume that these update operations take place in a taxonomy $\mathcal{T}$, then their preconditions can be expressed as shown in Table 5.

Operation	Pre-condition
`add`(a)	$a \notin \mathcal{T}$
`delete`(a)	$a \in \mathcal{T}$
`delete`$(b \leq a)$	$b \in Nr^F_{(1)}(a)$
`add`$(b \leq a)$	$b \notin Nr^F(a),\ f(b) = f(a)$
`addLeaf`(a, Par)	$a \notin \mathcal{T}$, $Par \subseteq \mathcal{T}$
`addIntermediate`(a, Chi, Par)	$a \notin T$, $Chi \subseteq \mathcal{T}$, $Par \subseteq \mathcal{T}$

Table 5: Preconditions of Taxonomy Update Operations

Let F' denote the faceted taxonomy F after one update operation. Below we describe the effects of each update operation on the broader and narrower terms of each term.

- `add`(a)

  $Nr^{F'}(a) = Br^{F'}(a) = \emptyset$

  If $t \neq a$ then $Br^{F'}(t) = Br^F(t)$ and $Nr^{F'}(t) = Nr^F(t)$.

- `delete`(a)

  Roughly, we could say that for every $t \neq a$ it holds $Br^F(t) - Br^{F'}(t) \subseteq \{a\}$ and $Nr^F(t) - Nr^{F'}(t) \subseteq \{a\}$. More specifically:

  If $t \leq a$, then $Br^{F'}(t) = Br^F(t) - \{a\}$, otherwise $Br^{F'}(t) = Br^F(t)$.

  If $a \leq t$, then $Nr^{F'}(t) = Nr^F(t) - \{a\}$, otherwise $Nr^{F'}(t) = Nr^F(t)$.

- `delete`$(b \leq a)$

  If $a \leq t$, then $Nr^{F'}(t) = Nr^F(t) - Nr^F(b)$, otherwise $Nr^{F'}(t) = Nr^F(t)$.

  If $t \leq b$, then $Br^{F'}(t) = Br^F(t) - Nr^F(a)$, otherwise $Br^{F'}(t) = Br^F(t)$.

- `add`$(b \leq a)$

  If $a \leq t$ then $Nr^{F'}(t) = Nr^F(t) \cup Nr^F(b)$, otherwise $Nr^{F'}(t) = Nr^F(t)$.

  If $t \leq b$ then $Br^{F'}(t) = Br^F(t) \cup Br^F(a)$, otherwise $Br^{F'}(t) = Br^F(t)$.

- `addLeaf`(a, Par)

  $Nr^{F'}(a) = \emptyset$

  $Br^{F'}(a) = \bigcup\{Br^F(p) \mid p \in Par\}$

  If $p \in Par$ and $p \leq t$, then $Nr^{F'}(t) = Nr^F(t) \cup \{a\}$, otherwise $Nr^{F'}(t) = Nr^F(t)$.

  If $t \neq a$ then $Br^{F'}(t) = Br^F(t)$.

- `addIntermediate`(a, Chi, Par)

  $Nr^{F'}(a) = \bigcup\{Nr^F(c) \mid c \in Chi\}$

  $Br^{F'}(a) = \bigcup\{Br^F(p) \mid p \in Par\}$

  If $p \in Par$ and $p \leq t$, then $Nr^{F'}(t) = Nr^F(t) \cup \{a\} \cup (\bigcup\{Nr^F(c) \mid c \in Chi\})$, otherwise $Nr^{F'}(t) = Nr^F(t)$.

  If $c \in Chi$ and $t \leq c$, then $Br^{F'}(t) = Br^F(t) \cup \{a\} \cup (\bigcup\{Br^F(p) \mid p \in Par\})$, otherwise $Br^{F'}(t) = Br^F(t)$.

4 Problem Statement

Let F be a faceted taxonomy and let e be an expression of CTCA that defines the desired compound terminology S_e^F. Now assume an update operation u_F on F and let F' be the resulting faceted taxonomy. Clearly, this update may turn the expression e obsolete, specifically:

- e may no longer be well-formed (and thus $S_e^{F'}$ may be undefinable),
- $S_e^{F'}$ may be well-formed but may no longer reflect the desire of the designer.

Roughly, and in the ideal case, we would like to find an expression e' such as:

(α) e' is well-formed, and

($\beta^=$) $S_{e'}^{F'} = S_e^F$.

Although condition (α) can be satisfied quite easily, condition ($\beta^=$) may be impossible to satisfy in some cases, e.g. in the obvious case when F' is derived by deleting terms from F. We can thus relax condition ($\beta^=$) and consider that our objective is to find an expression e' such that $S_e^{F'}$ is as *close* to S_e^F as possible. Of course, closeness or distance has to be defined formally. It is quite natural to define the distance between two compound terminologies S, S' as the cardinality of their symmetric difference (in the classical set-theoretic sense), i.e. we can write:

$$dist(S, S') = |(S - S') \cup (S' - S)| = |S - S'| + |S' - S| \tag{1}$$

Now let $\mathcal{S}^{F'}$ be the set of *all* compound terminologies over F' that can be defined by expressions of CTCA. We can now express condition (β) formally as follows:

(β) $S_{e'}^{F'} = \mathtt{arg}_S\ \mathtt{min}\{dist(S, S_e^F) \mid S \in \mathcal{S}^{F'}\}$

The righthand side of this equation returns the $S \in \mathcal{S}^{F'}$ that has the minimum distance from S_e^F.

However, in some application scenarios, we may prefer $S_{e'}^{F'}$ to be a subset of S_e^F than being a superset, or the reverse. Consequently, we may state state two, different than (β), conditions:

(γ) $S_{e'}^{F'} \subseteq S_e^F$ and $S_{e'}^{F'}$ is the biggest possible in $\mathcal{S}^{F'}$.
In other words, $S_{e'}^{F'} = \mathtt{arg}_S\ \mathtt{min}\{|S_e^F - S| \mid S \in \mathcal{S}^{F'}, S \subseteq S_e^F\}$

(δ) $S_{e'}^{F'} \supseteq S_e^F$ and $S_{e'}^{F'}$ is the smallest possible in $\mathcal{S}^{F'}$.
In other words, $S_{e'}^{F'} = \mathtt{arg}_S\ \mathtt{min}\{|S - S_e^F| \mid S \in \mathcal{S}^{F'}, S \supseteq S_e^F\}$

Of course, to find the sought expression e' we would not like to investigate all expressions in $\mathcal{S}^{F'}$ (as this would be computationally inadmissible), but we rather want to find a method for *modifying* e to an e' that satisfies (α) and (β or γ or δ).

5 CTCA Expression Revision

In the following we will assume that the basic compound terminologies are defined as in tree-structured taxonomies (i.e. $T_i = \{\{t\} \mid t \in \mathcal{T}_i\} \cup \{\emptyset\}$). The reason is that an operation $\mathtt{delete}(b \leq a)$ may turn a DAG-structured taxonomy into a tree-structured taxonomy, while an operation $\mathtt{add}(b \leq a)$ may turn a tree-structured taxonomy into a DAG-structured taxonomy or into a cyclic taxonomy. So these operations may change the basic compound terminologies. By adopting basic compound terminologies for tree-structured taxonomies we can overcome this issue and focus on the essential part of the problem of expression revision that concerns the combinations of elements from the basic compound terminologies (specifically, of those compound terms that contain at most one term from each facet).

Let's now introduce some additional notations. Given a compound term s and a term t, we shall use the notation $s\#t$ to denote the compound term $s - \{t\}$. Now given a compound term s and two terms t and t', we shall use the notation $s\#t\#t'$ to denote the compound term s if $t \notin s$, otherwise the compound term derived from s by replacing t by t', i.e.:

$$s\#t\#t' = \begin{cases} (s - \{t\}) \cup \{t'\}, & \text{if } t \in s \\ s & \text{otherwise} \end{cases}$$

For example, $\{a, b, c\}\#b\#e = \{a, e, c\}$, while $\{a, b, c\}\#e\#f = \{a, b, c\}$.

We can generalize and for every compound terms s, $s1$, $s2$, define:

$$s\#s1\#s2 = \begin{cases} (s - s1) \cup s2, & \text{if } s \cap s1 \neq \emptyset \\ s & \text{otherwise} \end{cases}$$

For example, $\{a, b, c\}\#\{b, c, d\}\#\{e, f, g\} = \{a, e, f, g\}$.

Below we study expression revision for each update operation u_F that can be applied on F.

5.1 term renaming, `rename`(t, t')

This is rather a trivial case. It is evident that the "best" compound terminology in $\mathcal{S}^{F'}$, is the one obtained by replacing t by t', i.e.: $S_{sol} = \{s\#t\#t' \mid s \in S_e\}$ (and clearly $S_{sol} \in \mathcal{S}^{F'}$).

In order to reach to an expression e' (that defines S_{sol}) we just have to replace the term t by the term t' in all compound terms of the parameters P and N of e (in case they contain the term t). Thus, from each parameter set P (or N) of e, we can derive the corresponding parameter P' (or N') of e', as follows:

$$P' = \{s\#t\#t' \mid s \in P\} \text{ and } N' = \{s\#t\#t' \mid s \in N\}.$$

5.2 term deletion, `delete`(a)

It is quite clear that here the "best" compound terminology in $\mathcal{S}^{F'}$, is the following: $S_{sol} = \{s\#a \mid s \in S_e^F\}$. It is also clear that for every P or N parameter of e, the corresponding P' or N' parameter of the sought expression e', should satisfy the following equations:

$$\begin{aligned} Br^{F'}(P') &= \{ s\#a \mid s \in Br^F(P)\} \\ Nr^{F'}(N') &= \{ s\#a \mid s \in Nr^F(N)\} \end{aligned}$$

Note that if a compound term s does not contain a then

$$Br^{F'}(s) = \{ s'\#a \mid s' \in Br^F(s)\} \text{ , and } \quad Nr^{F'}(s) = \{ s'\#a \mid s' \in Nr^F(s)\}$$

This holds due to the postconditions of `delete`(a) as mentioned in Section 3[3]. This means that we do not have to care about the parameters of e that do not contain the term a. On the other hand, if a appears in one parameter of e, then $S_e^{F'}$ is no longer well-formed. We should therefore modify all parameters of e that contain a. Consider a compound term s that contains a and s appears in a P

[3]More specifically:
If $t \leq a$, then $Br^{F'}(t) = Br^F(t) - \{a\}$, otherwise $Br^{F'}(t) = Br^F(t)$.
If $a \leq t$, then $Nr^{F'}(t) = Nr^F(t) - \{a\}$, otherwise $Nr^{F'}(t) = Nr^F(t)$.

parameter. In this case we should replace s by all s' that are obtained by replacing a by an immediately broader term of a. Let $sp = \{\ s\#a\#t \mid t \in Br^F_{(1)}(a)\}$. It is clear that $Br^{F'}(sp) = \{\ s\#a \mid s \in Br^F(s)\}$. If instead s appears in a N parameter, then we should replace s by all s' that are obtained by replacing a by an immediately narrower term of a. Let $sn = \{\ s\#a\#t \mid t \in Nr^F_{(1)}(a)\}$. It is clear that $Nr^{F'}(sn) = \{\ s\#a \mid s \in Nr^F(s)\}$.

Summarizing, we can define the sets P' and N' of e' as follows:

$$P' = \bigcup_{s \in P} \{s\#a\#t \mid t \in Br^F_{(1)}(a)\}$$
$$N' = \bigcup_{s \in N} \{s\#a\#t \mid t \in Nr^F_{(1)}(a)\}$$

Consequently, it is not hard to see that it holds: $S^{F'}_{e'} = \{s\#a \mid s \in S^F_e\}$. Clearly, this is the closest to S^F_e compound terminology over F'. Specifically, it satisfies the condition (β). Figure 6 shows two examples of such an updating. In the first one, e is a plus-product operation, while in the second, e is a minus-product operation.

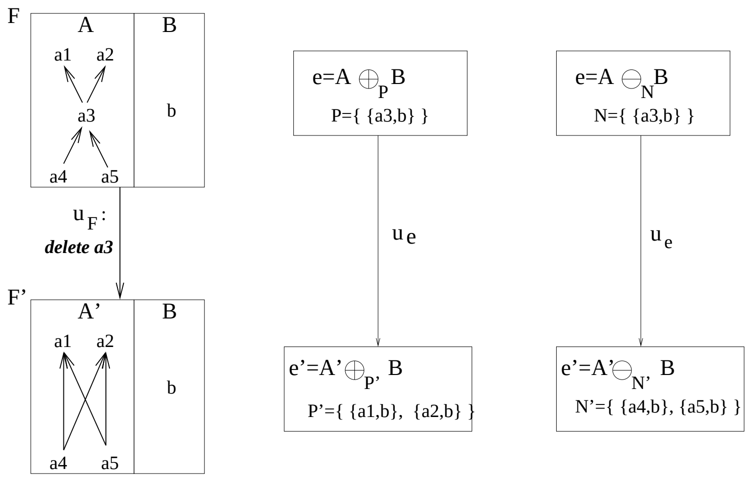

Figure 6: Expression revision after term deletion

5.3 term addition, `add`(a)

Clearly, this addition does not make e obsolete, i.e. $S^{F'}_e$ is certainly a well-formed expression. The question here is whether the compound terms that contain the newly inserted term a should be valid or not. According to the minimum distance criterion (of Section 4), they should be invalid, in other words, it should hold $S^F_e = S^{F'}_{e'}$.

Suppose that a has been assigned to a facet $\mathcal{T}_i$ which is operand of a plus-product operation. In this case, we don't have to update the parameter P of this operation because all compound terms that contain a do not belong to $Br^{F'}(P)$ (because a is not connected to any other element of $\mathcal{T}_i$). For the same reason we do not need to update any other P parameter of e.

On the other hand, if facet $\mathcal{T}_i$ is operand of a minus-product operation, then we have to modify the parameter N. The reason is that since a is not connected to any other term of $\mathcal{T}_i$, all compound terms that contain a cannot belong to $Nr^{F'}(N)$, hence they are considered as valid (according to the semantics of $\ominus_N$). Below we explain how we can modify N so as to turn these compound terms invalid. Let *tops* denote the maximal elements (w.r.t. $\preceq$) of the compound terminologies that are operands of the minus-product operation, excluding the facet $\mathcal{T}_i$. For example, if $e = \ominus_N(T_1, \ldots, T_k)$

then $tops = \cup_{j=1...k, j \neq i} maximal_{\leq}(\mathcal{T}_j)$. We have to add to N all compound terms $\{ \{a, u\} \mid u \in tops\}$, thus we can define N' as follows:

$$N' = N \cup \left(\bigcup_{j=1..k, j \neq i} \{ \{a, u_j\} \mid u_j = maximal_{\leq}(T_j)\} \right)$$

We have to update analogously the N parameter of every minus-product operation. Specifically, for every minus-product operation $\ominus_N(e_1, ..., e_k)$ and for every e_i $(1 \leq i \leq k)$ such that $f(a) \notin f(e_i)$, we have to add to N the parameter $\{a, u_i\}$ for each $u_i \in maximal_{\preceq}(S_{e_i})$.

It is not hard to see that in this way we will get $S_{e'}^{F'} = S_e^F \cup \{\{a\}\}$.

5.4 subsumption relationship deletion, `delete`$(b \leq a)$

This deletion does not necessarily make e obsolete. However, this deletion can change the sets $Nr(N)$ or $Br(P)$ of the operation that contains the facet of the terms a and b, and thus change the set of genuine compound terms of a subsuming operation, turning the expression e not well-formed.

Let's now suppose that we seek for an e' such as $S_e^F = S_{e'}^{F'}$. Ideally, for every P or N of e we want to find a P' or N' such that: $Br^{F'}(P') = Br^F(P)$ and $Nr^{F'}(N') = Nr^F(N)$. Recall from Sec. 3 that after the operation `delete`$(b \leq a)$ it holds:

If $a \leq t$, then $Nr^{F'}(t) = Nr^F(t) - Nr^F(b)$, otherwise $Nr^{F'}(t) = Nr^F(t)$.

If $t \leq b$, then $Br^{F'}(t) = Br^F(t) - Br^F(a)$, otherwise $Br^{F'}(t) = Br^F(t)$.

It follows that we have to care only about those parameters of e that contain either a term broader than a, or a term narrower than b. For these parameters we should add extra parameters so that to recoup the "missing compound terms", i.e. those missed due to the reduction of $Nr(t)$ and $Br(t)$.

For achieving this, for each $s \in P$ which contains a term t'' that is narrower than b, we add to P' a compound term s' which is derived from s by replacing t'' by a. One can easily see that in this way we have $Br^{F'}(P') = Br^F(P)$. Specifically the set P' is defined as follows:

$$P' \quad = \quad P \cup \{ \ s \# Nr^F(b) \# \{a\} \mid s \in P\}$$

Analogously, for each $s \in N$ which contains a term t'' that is broader than a, we add to N' a compound term s' which is derived from s by replacing t'' by b. One can easily see than in this way we have $Nr^{F'}(N') = Nr^F(N)$. Specifically, the set N' is defined as follows:

$$N' \quad = \quad N \cup \{ \ s \# Br^F(a) \# \{b\} \mid s \in N\}$$

We can easily see that in this way the result of the operation that involves the facet $\mathcal{T}_i$ remains the same. Consequently, we don't have to make any other update on the expression. We have achieved $S_e^F = S_{e'}^{F'}$. Figure 7 shows two examples of such an updating: one for a plus-product and one for a minus-product operation.

5.5 subsumption link addition, `add`$(b \leq a)$

Although this addition does not necessarily make e obsolete, it may however change the genuine compound terms of a subsuming operation, and thus turn the entire e not well-formed. Our main objective is to revise the expression to a well-formed one. Secondly, we would like to find an e' such as $S_e^F = S_{e'}^{F'}$. As we shall will see below there are cases where there is no expression e' such that $S_e^F = S_{e'}^{F'}$. Two such cases are shown in Figure 8. Below we shall identify when this happens.

In the following we assume that terms a and b belong to a facet $\mathcal{T}_i$ of a faceted taxonomy F, and that F' denotes the faceted taxonomy after the update operation `Add`$(b \leq a)$.

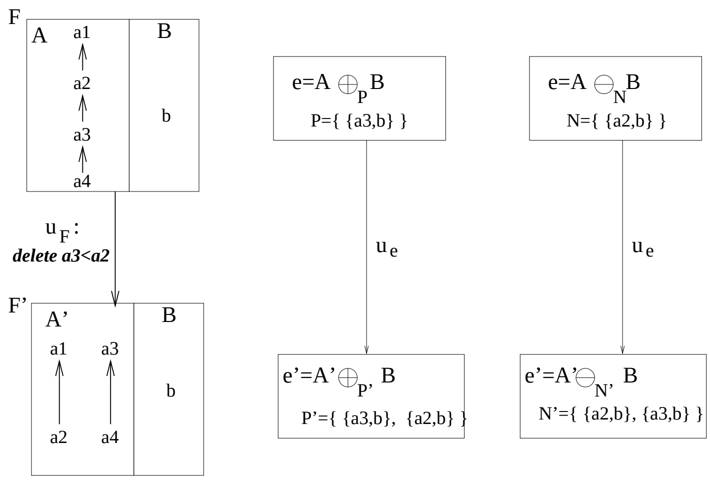

Figure 7: Expression revision after subsumption relationship deletion

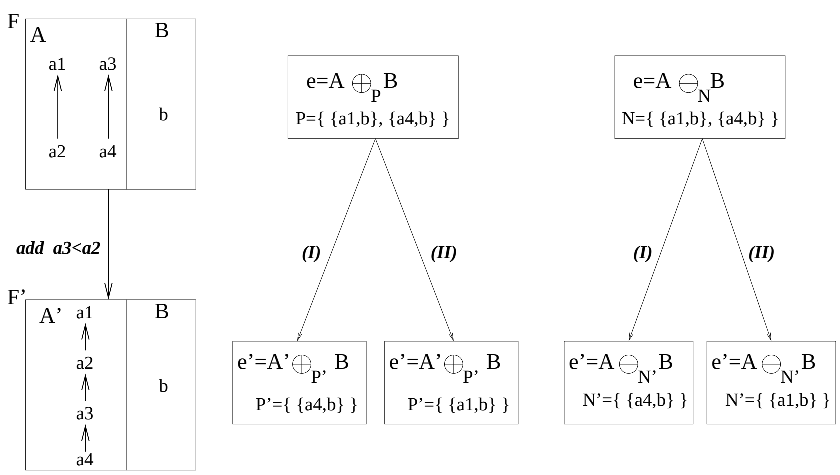

Figure 8: Expression revision after subsumption relationship addition

Prop. 1 We can find an expression e' such that $S_{e'}^{F'} = S_e^F$ if and only if:

(a) for every P parameter of e, it holds:

$$\nexists x \neq \emptyset \text{ s.t. } \{b\} \cup x \in Br^F(P) \text{ while } \{a\} \cup x \notin Br^F(P)$$

(b) for every N parameter of e, it holds:

$$\nexists x \neq \emptyset \text{ s.t. } \{a\} \cup x \in Nr^F(N) \text{ while } \{b\} \cup x \notin Nr^F(N)$$

Proof (sketch):

For every P of e, our objective is to find a P' such that $Br^F(P) = Br^{F'}(P')$. This is impossible if there is a nonempty compound term x (i.e. $x \neq \emptyset$) such that:

$$\{b\} \cup x \in Br^F(P) \text{ while } \{a\} \cup x \notin Br^F(P) \quad (2)$$

This is because in F' it always holds: $\{b\} \cup x \preceq \{a\} \cup x$.

Now for every N parameter of e, our objective is to find a N' such that $Nr^F(N) = Nr^{F'}(N')$. One can easily see that this is impossible if there is a compound term $x \neq \emptyset$ such that:

$$\{a\} \cup x \in Nr^F(N) \text{ while } \{b\} \cup x \notin Nr^F(N) \quad (3)$$

This is because in F' it always holds: $\{b\} \cup x \preceq \{a\} \cup x$.

$\diamond$

In other words, we can find an expression e' such that $S_{e'}^{F'} = S_e^F$ if and only if for every $x \neq \emptyset$ it holds: if $\{b\} \cup x \in Br^F(P)$ then $\{a\} \cup x \in Br^F(P)$ and if $\{a\} \cup x \in Nr^F(N)$ then $\{b\} \cup x \in Nr^F(N)$. In this case, we can set $e' = e$ and get $S_e^{F'} = S_e^F$.

If conditions (a) and (b) of Prop. 1 do not hold, then it is not possible to satisfy the condition ($\beta^=$). Moreover, and as we shall see below, there are cases where it is impossible to reach an $S_{e'}^{F'}$ that is either a subset or a superset of S_e^F. For instance, consider the case shown in Figure 9 and the following expression:

$$e = (A \oplus_P B) \ominus_N C \text{ where } P = \{\{a2, b\}\} \text{ and } N = \{\{a1, c\}\}$$

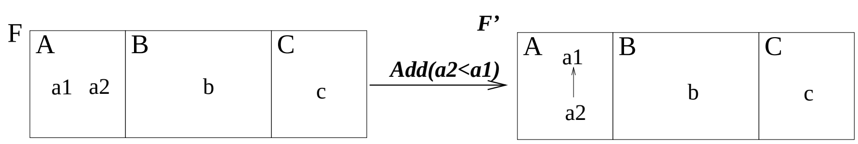

Figure 9: Addition of a subsumption relationship

In this example we have:

$$\begin{aligned}
S_{A\oplus_P B}^F &= \{\{a2, b\}, \{a1\}, \{a2\}, \{b\}\} \\
S_{A\oplus_P B}^{F'} &= \{\{a2, b\}, \{a1, b\}, \{a1\}, \{a2\}, \{b\}\} \\
S_{(A\oplus_P B)\ominus_N C}^F &= \{\{a2, b, c\}, \{a2, c\}, \{b, c\}, \{a2, b\}, \{a1\}, \{a2\}, \{b\}\} \\
S_{(A\oplus_P B)\ominus_N C}^{F'} &= \{\{a2, c\}, \{b, c\}, \{a2, b\}, \{a1, b\}, \{a1\}, \{a2\}, \{b\}\}
\end{aligned}$$

Although $S_{A\oplus B}^{F'} \supset S_{A\oplus B}^F$, $S_e^{F'}$ misses elements that are contained in S_e^F. Notice that $S_{e'}^{F'}$ and S_e^F are not related with subset relation (only S_e^F contains the compound term $\{a2, b, c\}$ and only $S_{e'}^{F'}$ contains the compound term $\{a1, b\}$).

Three questions arise now:

(i) How can we check efficiently whether the conditions (a) and (b) of Prop. 1 hold?

(ii) If the conditions of Prop. 1 do not hold, is $S_e^{F'}$ well-formed or not?

(iii) If the conditions of Prop. 1 do not hold, how we should revise e to a well-formed (w.r.t. F') e' so that the distance between S_e^F and $S_{e'}^{F'}$ to be minimal ?

The subsequent two propositions gives us an answer to question (i).

Prop. 2
If $p \cap Nr^F(b) = \emptyset$ for all $p \in P$ of every P of e, and
if $n \cap Br^F(a) = \emptyset$ for all $n \in N$ of every N of e,
then conditions (a) and (b) of Prop. 1 hold, and thus, $S_e^{F'} = S_e^F$.

Proof:

Recall from Sec. 3 that:
if $t \in Br^F(a)$ then $Nr^{F'}(t) = Nr^F(t) \cup Nr^F(b)$, otherwise $Nr^{F'}(t) = Nr^F(t)$.
if $t \in Nr^F(b)$ then $Br^{F'}(t) = Br^F(t) \cup Br^F(a)$, otherwise $Br^{F'}(t) = Br^F(t)$.

This means that:
if $p \cap Nr^F(b) = \emptyset$ then $Br^{F'}(t) = Br^F(t)$ for each $t \in p$, which implies that $Br^{F'}(p) = Br^F(p)$.
if $n \cap Br^F(a) = \emptyset$ then $Nr^{F'}(t) = Nr^F(t)$ for each $t \in n$, which implies that $Nr^{F'}(n) = Nr^F(n)$.

$\diamond$

This means that if the above conditions (which can be evaluated by a simple and efficient algorithm) are satisfied, then we are sure that $S_e^{F'}$ is well-formed and that $S_e^{F'} = S_e^F$. If on the other hand, they do not hold, then we cannot decide whether the conditions (a) and (b) of Prop. 1 hold or not. The following proposition gives us sufficient and necessary conditions.

Prop. 3 We can find an expression e' such that $S_{e'}^{F'} = S_e^F$ if and only if:

(i) for each $p \in P$ of every parameter P of e it holds:
If $p \cap Nr^F(b) \neq \emptyset$ then $\exists p' \in P$ such that $p' \preceq_F (p - Nr^F(b)) \cup \{a\}$

(ii) for each $n \in N$ of every parameter N of e it holds:
If $n \cap Br^F(a) \neq \emptyset$ then $\exists n' \in N$ such that $n' \succeq_F (n - Br^F(a)) \cup \{b\}$.

If (i) and (ii) hold then $S_e^{F'} = S_e^F$.

Proof:

Firstly, note that if $p \cap Nr^F(b) = n \cap Br^F(a) = \emptyset$, then the conditions (a) and (b) of Prop. 1 hold due to Prop. 2.

According to Prop. 1, we can find an e' such that $S_e^F = S_{e'}^{F'}$ if whenever $\{b\} \cup x \in Br^F(P)$, it also holds $\{a\} \cup x \in Br^F(P)$, for every $x \neq \emptyset$. We can easily see that there exists a $x \neq \emptyset$ such that $\{b\} \cup x \in Br^F(P)$, if and only if there is a $p \in P$ such that $p \cap Nr^F(b) \neq \emptyset$. Let's now investigate what x can be. From the above we can infer that $x \in Br^F(p - Nr^F(b))$. Thus we want $\{a\} \cup x \in Br^F(P)$, for each $x \in Br^F(p - Nr^F(b))$. This is true if and only if $\exists p' \in P$ such that $p' \preceq_F (p - Nr^F(b)) \cup \{a\}$. So we proved (i).

(ii) is proved analogously to (i).

$\diamond$

Question (ii)

If the conditions of Prop. 1 do not hold, then it is impossible to find an e' such that $S_e^F = S_{e'}^{F'}$. However, $S_e^{F'}$ is not necessarily badly-formed. For instance, in the example of Figure 9, $S_e^{F'}$ is well-formed. One method to check whether $S_e^{F'}$ is well-formed is to check whether every individual element of the P/N parameters of e belongs to the associated set of genuine compound terms. Notice that this involves running $|\mathcal{P} \cup \mathcal{N}|$ times the algorithm $IsValid(e, s)$ [25].

Question(iii)

Suppose we follow the approach described above, i.e. we check every individual element of the P/N parameters of e. What should we do in case we encounter an element of a parameter that does not belong to the genuine compound terms of the associated operation? Should we delete it or modify it and how?

Let's first recall the effects of adding a subsumption relationship. For every t it holds $Br^F(t) \subseteq Br^{F'}(t)$ and $Nr^F(t) \subseteq Nr^{F'}(t)$. It follows that for every P or N parameter of an expression e it holds:

- $Br^F(P) \subseteq Br^{F'}(P)$, hence $S_e^F \subseteq S_e^{F'}$ (the compound terminology *grows*)
- $Nr^F(N) \subseteq Nr^{F'}(N)$, hence $S_e^F \supseteq S_e^{F'}$ (the compound terminology *shrinks*)

As only in minus-products the compound terminology becomes smaller (in plus-products it becomes bigger), we may encounter a problematic compound term in a parameter of an operation that has as operand (direct or indirect) a minus-product operation. This means that if e has only plus-products then $S_e^{F'}$ is certainly well-formed.

Below we introduce some notation that we shall use in the sequel.

- $exprs(e)$: the subexpressions of e. Each non-leaf node of the parse tree of e correspond to a subexpression of e. Note that $e \in exprs(e)$.
- $exprs^-(e)$: the subexpressions of e that contain at least one $\ominus$ operator that is not their top-most operation. For example, $exprs^-((((T_1 \ominus T_2) \oplus T_3) \ominus T_4) \oplus T_5) =$ $\{\ (T_1 \ominus T_2) \oplus T_3,\ \ ((T_1 \ominus T_2) \oplus T_3) \ominus T_4,\ \ (((T_1 \ominus T_2) \oplus T_3) \ominus T_4) \oplus T_5\}$.

Returning to our problem. If a parameter element s of an expression e does not belong to the corresponding set of genuine compound terms (w.r.t. F'), then this is due to a contained minus-product operation. This means that only expressions in $exprs^-(e)$ can have parameter elements that are not genuine.

Suppose that in the context of an expression e we encounter a parameter element s of e that is not genuine. Now for each $e_i \in exprs^-(e)$ we can define $s_i = \pi_{f(e_i)}(s)$. Since $s \notin G_e^{F'}$ (where $G_e^{F'}$ denotes the set of genuine compound terms of the operands of e) we are sure that for at least one i it holds: $s_i \notin S_{e_i}^{F'}$. We can also be sure that e_i is a minus-product operation. Our objective is to fix this problem. This can be achieved in two different ways:

(a) revise s_i to an s'_i such that $s'_i \in S_{e_i}^{F'}$

If we do this kind of revision to each "problematic" s_i then we will reach a s' that belongs to $S_{e_j}^{F'}$ (if there is only one problematic s_i, then $s' = (s - s_i) \cup s'_i$). So by following this approach we will finally reach to an expression e' that is well-formed. What is left to describe is how we should revise each problematic s_i (this will be explained below).

(b) revise e_i to an e'_i such that $s_i \in S_{e'_i}^{F'}$

Recall that e_i is certainly a minus-product. Solving the problem requires enlarging the compound terminology of e_i, i.e. deleting or relaxing (narrowing) one or more parameter elements of e_i.

One remark here is that the revision of e_i will not cause any extra non genuine compound terms (in the subsuming operations) because it will hold $S^{F'}_{e_i} \subset S^{F'}_{e'_i}$.

It is evident that e_i has at least one parameter element n_i such that $s_i \in Nr^{F'}(n_i)$. Specifically, the previous propositions imply that:

- s_i certainly has a term $t'' \leq b$, and
- n_i certainly has a term $t' \geq a$.

Policy (a) means revising s_i to an s'_i such that $s'_i \notin Nr^{F'}(n_i)$.
Policy (b) means revising n_i to an n'_i such that $s_i \notin Nr^{F'}(n'_i)$.
We can implement policy (a) by replacing in s_i the term t'' by the term $Br^F_{(1)}(t')$, i.e.

$$s'_i = s_i \# Nr^F(b) \# Br^F_{(1)}(t')$$

We can implement policy (b) by replacing in n_i the term t' by the term $Nr^F_{(1)}(t'')$, i.e.

$$n'_i = n_i \# Br^F(a) \# Nr^F_{(1)}(t'')$$

It is evident, that $s'_i \notin Nr^{F'}(n_i)$ (in policy (a)) and that $s_i \notin Nr^{F'}(n'_i)$ (in policy (b)).
For simplicity, above we have assumed that there is only one n_i and that $|Br^F_{(1)}(t')| = |Nr^F_{(1)}(t'')| = 1$. Below we describe the algorithms for the general case.

Policy (a)

(1). $\mathbf{n} := \{n \in N \mid s_i \preceq_{F'} n\}$
(2). $X := \bigcup_{n_i \in \mathbf{n}} (n_i \cap Br^F(a))$
(3). $Y := \bigcup_{x \in maximal_{\leq}(X)} Br^F_{(1)}(x)$
(4). $Z := \{s_i \# Nr^F(b) \# \{y\} \mid y \in Y\}$
(5). Replace s by the set of compound terms $\{s \# s_i \# z \mid z \in Z\}$

Step (1) defines the set $\mathbf{n}$ comprising all parameter elements of N that are broader than s_i. Step (2) computes the set X consisting of those terms of the elements in $\mathbf{n}$ that are broader than a (i.e. all "t'" in our previous discussion). Let's now discuss step (3). It is clear that the revised version of s_i should not contain any term of $\bigcup_{x \in X} Nr^{F'}(x)$. So the terms of s_i that belong to $Nr^{F'}(b)$ should be replaced by terms that belong to $\bigcup_{x \in X} Br^{F'}(x) - X$ (note that $\bigcup_{x \in X} Br^{F'}(x) - X \subseteq \mathcal{T} - \bigcup_{x \in X} Nr^{F'}(x)$). Now according to the minimum distance criterion, the most preferable terms are those in the set Y as defined in step (3). Step (4) computes the set Z of all revised versions of s_i, and finally, step (5) replaces the original "problematic" parameter element s by one or more compound terms, specifically by those derived after substituting the s_i part of s (recall that $s_i \subseteq s$) by the revised version(s) of s_i.

Policy (b)

(1). $\mathbf{n} := \{n \in N \mid s_i \preceq_{F'} n\}$
(2). For each $n_i \in \mathbf{n}$ do
(3). $\quad X := s_i \cap Nr^F(b)$
(4). $\quad Y := \bigcup_{x \in minimal_{\leq}(X)} Nr^F_{(1)}(x)$
(5). $\quad Z := \{n_i \# Br^F(a) \# \{y\} \mid y \in Y\}$
(6). $\quad$ Replace n_i by the set of compound terms Z

Again, step (1) defines the set $\mathbf{n}$ comprising all parameter elements of N that are broader than s_i. It is this set of parameter elements that we should revise in order to reach a N' such that $s_i \notin Nr^{F'}(N')$. For each n_i in $\mathbf{n}$, step (3) computes the set X comprising the terms of n_i that are narrower than b (i.e. all "t''" in our previous discussion). All these terms have to be replaced. Furthermore, the revised version of n_i should not contain any term in $\bigcup_{x \in X} Br^{F'}(x)$. In the place of these terms, n_i should contain terms from the set $\bigcup_{x \in X} Nr^{F'}(x) - X$ (note that $\bigcup_{x \in X} Nr^{F'}(x) - X \subseteq \mathcal{T} - \bigcup_{x \in X} Br^{F'}(x)$). According to the minimum distance criterion, the most preferable terms are those in the set Y as defined in step (4). At the end of this algorithm we are sure that $s_i \notin Nr^{F'}(N')$[4].

Concerning the dilemma policy (a) versus policy(b), note that both of them result in revised parameters. Policy (a) favors revising the parameters of operations that are high in the parse tree, while policy (b) prefers those that are low in the parse tree. From the perspective of the minimum distance criterion, it can be shown that the "distance" of the resulting compound terminology is the closest possible but this is true only for the operation whose parameters we decided to update. Concerning the compound terminology of the entire expression we cannot say for sure which policy prevails, as this depends on the size of all S_{e_i}. So the choice is left to the designer, or the system may adopt by default one policy.

The reader could now see again the example of Figure 8. As another example consider the case shown in Figure 10. The addition of the subsumption link $b_2 \leq b_1$ makes the expression e not well-formed as the compound term $\{a, b_2, c\}$ no longer belongs to the set of genuine compound terms of the plus-product operation $(A \ominus_N B) \oplus_P C$. According to policy (a) and since $Br^F_{(1)}(b_1) = \{\emptyset\}$, we actually have to delete the term b_2 from the compound term $\{a, b_2, c\}$. Policy (b) is left to the reader.

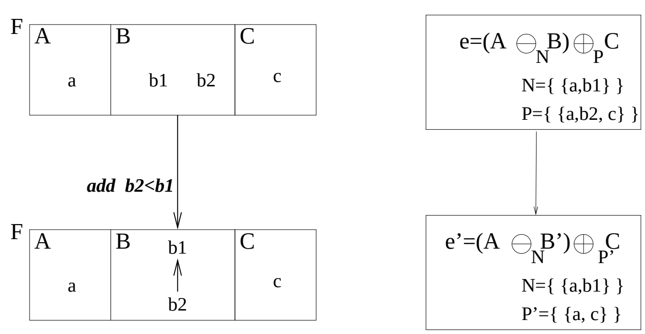

Figure 10: Addition of a subsumption relationship and well-formedness

In conclusion, Prop. 3 gave us an efficient method for checking whether condition $(\beta^{=})$ can be

[4]This however does not guarantee that the elements of the revised parameter N' will not be revised again while checking the genuineness of the subsequent parameter elements of the current or of a subsuming operation. So an alternative policy, in which each parameter element n_i is revised at most once, would be:
(1) for each $n_i \in N$ check all parameter elements of all subsuming operations, and
(2) collect the problematic parameter elements, and revise n_i appropriately.

satisfied. When $(\beta^=)$ is impossible to satisfy, we gave two different methods for reaching a well-formed expression that also satisfies a minimum distance criterion locally.

5.6 leaf addition, `addLeaf`(a, Par)

As mentioned in Sec. 3, this operation is analyzed to an operation $add(a)$ and an operation $add(a \leq p)$ for each $p \in Par$. Although we could satisfy $(\beta^=)$, here it is more reasonable to assume that the designer would prefer a to "follow its parents", i.e. the terms in Par.

Let's first suppose that $|Par| = 1$, and assume that $Par = \{tp\}$. "Follow the parent" means that if a valid compound term s contains tp, then $s\#tp\#a$ should be valid too. For example, if we add the term $Crete$ under the term $Greece$ (in the example of Figure 1), then $\{Feta, Crete\}$ should be valid too. So if $|Par| = 1$, it is clear what $S_{e'}^{F'}$ should be. Concerning expression revision and plus-products, note that a compound term s that contains tp can be valid only due to a parameter p that contains a term $t' \in Nr^F(tp)$. This implies that for each such parameter element (that contains a term $t' \in Nr^F(tp)$), we should add a new parameter where t' is replaced by a. Consequently, reaching to the sought e' requires revising each parameter P as follows:

$$P' = P \cup \{\, p\#Nr^F(tp)\#\{a\} \mid p \in P\}$$

Let's now consider the minus-product operations. Note that if $s \in Nr^F(N)$ then $s\#tp\#a \in Nr^{F'}(N)$ because $a \in Nr^{F'}(tp)$. Consequently, we don't have to revise the N parameters of e.

Let's now consider the case where $|Par| > 1$ and suppose that Par comprises two terms $tp1$ and $tp2$. Let $s1$ and $s2$ be two compound terms that contain the same terms except that $s1$ contains $tp1$ and $s2$ contains $tp2$. Consider now the case where $s1$ is valid, while $s2$ is not valid. In that case the designer must decide about the validity of $s1\#tp1\#a$ (similarly $s2\#tp2\#a$). For example, suppose we add the (unrealistic) term $RoquefortFeta$ under the terms $Roquefort$ and $Feta$. Should $\{RoquefortFeta, Greece\}$ be valid or not? Note that $\{Roquefort, Greece\}$ is invalid, while $\{Feta, Greece\}$ is valid. According to the minimum distance criterion, it is better to update e so as $s1\#tp1\#a$ to be invalid (e.g. $\{RoquefortFeta, Greece\}$ is invalid). In the opposite case, i.e. if we update the expression so as $s1\#tp1\#a$ to be valid, then all compound terms that are broader than $s1\#tp1\#a$ would be valid (so $s2$ would no longer be invalid). The revision algorithm follows easily from the above.

5.7 intermediate term addition, `addIntermediate`(a, Chi, Par).

As mentioned in Sec. 3, this operation is analyzed to an operation $add(a)$ and an operation $add(a \leq p)$ for each $p \in Par$ and an operation $add(c \leq a)$ for each $c \in Chi$.

The discussion of `addLeaf` applies here as well, i.e. a should "follow" its parents (if all of them are valid or all of them are invalid). In case their validity is not the same, and there is a conflict (as described in `addLeaf`), a should follow the invalid compound terms.

Concerning the children Chi, if a compound term s contains a term $c \in Chi$ and is valid, then $s\#c\#a$ should be valid too (according to the minimum distance criterion). Now in case all parents are valid and all children are invalid, then $s\#c\#a$ should be invalid, according to the minimum distance criterion, but in practice the decision is up to the designer. For example, consider the case a new term X is placed between $Ingredients$ and $Truffle$. Whether $\{Greece, X\}$ should be valid or not depends on the meaning of X and the domain knowledge of the designer.

5.8 Epilogue

The results reported so far apply also for the case where e contains self-product operations. One slight difference is that in case of an operation $\texttt{Add}(a)$ applied on a facet T_i that participates in a minus-self-product operation with parameter N, we should add to N a pair $\{a, t\}$ for each maximal element t of T_i.

One might wonder, if we could do any better (i.e. satisfy condition ($\beta^=$), or reach to a compound terminology closer to S_e^F) by an expression e' with structure (parse tree) different than e. The answer is negative. At first note that previous work [21] has proved that if A is a subset of $\mathcal{P}(\mathcal{T})$ such that $Br(A) = A$, then there is always an expression e such that $S_e = A$. Moreover, we have shown that this is true, for every possible parse tree of e (i.e. for every possible order of operations, operands and parentheses). This means that set of compound terminologies that can be specified by an expression with a given parse tree equals the set of compound terminologies that can be specified by an expression of any parse tree. Thus, it is worthless to investigate whether a differently structured expression can be closer to the original. So we can study the problem of expression revision, without wondering whether the revised expression should have a different parse tree.

6 Similar Problems and Related Work

There is not any directly related work on the problem at hand because CTCA emerged relatively recently and its distinctive characteristics (range-restricted closed world assumptions) differentiate it from other logic-based languages and the corresponding literature on updates and revisions.

We could however draw some analogies to some well-known problems. For instance, we could consider F as a database and $CTCA$ as a query language, meaning that each expression e can be construed as a query that returns a subset of $\mathcal{P}(\mathcal{T})$. Under this view, expression revision resembles the problem of view definition revision in databases. The latter problem is formulated as follows: given a (e.g. relational) database db and one view definition (named query) q, how we should revise q (to a q') after an update operation on db (that resulted in db'), so that to satisfy the following: $ans_{db}(q) = ans_{db'}(q')$. Note that this is not the classical problem of updating databases through views, i.e. how to update the database after an update upon the contents (tuples) of the view, i.e. the transition $ans(q)' \rightarrow db'$ (e.g. see [2, 8]), nor the problem of (incremental) updating the contents of a view after a database update, i.e. the transition $db' \rightarrow ans(q)'$ (e.g. see [9, 10]). In our case we want to revise the definition of the view, i.e. the focus is given on the transition $db' \rightarrow q'$.

From the perspective of KB (knowledge base) revision (e.g. see [3, 7]) we could say that CTCA expression revision corresponds to a special case of KB revision. Under this point of view we could consider the pair (F, e) as a KB from which other sentences (here, term conjunctions) can be inferred. Let's denote the later by $Cons(F, e)$ (where $Cons$ comes from Consequence) and write $Cons(F, e) = \{s \mid (F, e) \models s\} = S_e^F$, where "$\models$" is based on the semantics of CTCA. Consider now an update u on F and let $F' = u \circ F$. From this perspective, our objective is to revise e to an expression e' such that (F', e') is well-formed and the difference between $Cons(F, e)$ and $Cons(F', e')$ is minimal. Take into account that all KB revision approaches also adopt a minimum change criterion, i.e. conform to the new information but retain as much as possible the old knowledge. Notice that we do not focus on the update $F' = u \circ F$, although one can easily see that the definition of the primitive taxonomy update operations tacitly adopt a minimum change criterion too[5]. In our case, we consider F' as

[5]In each taxonomy update operation, we retain as much of the old knowledge that can be retained while restricting ourselves within the expressive power of the framework, i.e. within the expressive power of taxonomically organized terms. For instance, in *delete*(*term*) we "retain" the transitively induced subsumption links of the precedent state of the

unquestionable and try to update/revice e according to our objectives. Another remark is that the classical KB revision focuses on how to revise a KB when new contradictory information is obtained. Of course, the notion of contradiction can be defined in several different ways. If we would like to identify the most contradictory case, then this would be the operation $\texttt{Add}(\texttt{a} \leq \texttt{b})$, because this operation sometimes obliges us to update the parameters of operands out of the scope of the taxonomy operation (even to obtain well-formedness).

At last we have to note that a representation of (F, e) in logic would not offer much, firstly because CTCA cannot be directly expressed, and secondly because this would rather complicate the problem and the notations. In addition, it is well-known [7, 28] that in KB revision there is no general method that will "do the right thing" under all circumstances.

7 Concluding Remarks

This paper showed how we can revise a CTCA expression e after a taxonomy update and reach an expression e' that is both well-formed and whose semantics (specified compound terms) is as close as possible to the original expression e before the update. Various cases were analyzed and the revising algorithms were given.

In summary, the deletion of terms or subsumption relationships can be handled by extending the P/N parameters (so as to recover the missing compound terms from the semantics of the original expression). On the other hand, the addition of subsumption relationships cannot be handled always. The reason is that since the semantics of the operations $\oplus_P/\ominus_N$ are defined on the basis of the transitive relation $\preceq$ (which is derived by $\leq$), after the addition of a subsumption relationship we may no longer be able to separate (from the semantics) compound terms that were previously separable (i.e. compound terms which were not $\preceq$-related before the addition of the subsumption link). We saw that after such taxonomy updates, the resulting compound terminology may neither be subset nor superset of the original compound terminology. This happens because the effects of adding a subsumption relationship is different in $\oplus_P$ and $\ominus_N$. Specifically, the compound terminologies defined by $\oplus_P$ operations become larger, while those defined by $\ominus_N$ operations become smaller. Now the combination of $\oplus_P$ and $\ominus_N$ operations leads to compound terminologies which are neither larger nor smaller than the original one. In such cases, we saw how we can revise e to an e' that is well-formed with respect to F'. Two policies were identified. For each of them we gave a revision algorithm that satisfies the minimum distance criterion locally (i.e. in the operation's context). The above results are summarized in Table 6.

u_F	relationship between S_e^F and $S_{e'}^{F'}$	Notes
$\texttt{rename}(a, a')$	$(\beta^=)$	up to term renaming
$\texttt{delete}(a)$	(β)	$S_{e'}^{F'} = S_e^F - \{s \mid a \in s\}$, thus $S_{e'}^{F'} \subseteq S_e^F$
$\texttt{add}(a)$	$\sim (\beta^=)$	$S_{e'}^{F'} = S_e^F \cup \{\{a\}\}$
$\texttt{delete}(b \leq a)$	$(\beta^=)$	$S_{e'}^{F'} = S_e^F$
$\texttt{add}(b \leq a)$	$(\beta^=)$, or (β), or (γ), or (δ), or none	several cases

Table 6: Synopsis of Expression Revision

This work can significantly aid the application of faceted classification and CTCA in real world applications where updates are very frequent. Without such a service, designers are obliged to reformulate their expressions after taxonomy updates.

taxonomy (old knowledge).

An issue for further research is to study the problem of expression revision after a *sequence* of taxonomy updates. In this case, F' would be the result of applying a sequence of updates U on F. Instead of deriving one revised e after each update in U, a more efficient approach is to consider and preprocess the entire set of updates U, because we could eliminate the "balancing" update operations (e.g. `delete`(a) vs. `add`(a), `delete`$(a \leq b)$ vs. `add`$(a \leq b)$, etc) that may be contained in U. This would allow managing efficiently "long (taxonomy update) transactions" which are quite common in design applications.

Acknowledgements

Many thanks to Anastasia Analyti for proof reading the paper (twice) and to Nicolas Spyratos for providing me with some useful bibliographic references.

References

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

[2] F. Bancilhon and N. Spyratos. "Update Semantics of Relational Views". *ACM Transactions on Database Systems*, December 1981.

[3] Mukesh Dalal. "Investigations Into a Theory of Knolwedge Base Revision". In *Conference on Artificial Intelligence, AAAI-88*, pages 475–479, St. Paul, Minesota, August 1988.

[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] F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. "Reasoning in Description Logics". In Gerhard Brewka, editor, *Principles of Knowledge Representation*, chapter 1, pages 191–236. CSLI Publications, 1996.

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

[7] Thomas Eiter and Georg Gottlob. On the complexity of propositional knowledge base revision, updates, and counterfactuals. In *Proceedings of the Eleventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 261–273, 1992.

[8] Arthur M. Keller. "The Role of Semantics in Translating View Updates". *IEEE Computer*, pages 63–73, January 1986.

[9] D. Laurent, J. Lechtenberger, N. Spyratos, and G. Vossen. "Monotonic Complements for Independent Data Warehouses". *VLDB Journal*, 10(4):295–315, December 2001.

[10] Jens Lechtenberger and Gottfried Vossen. "On the Computation of Relational View Complements". In *Proceedings of PODS 2003*, pages 142–149, Madison, Wisconsin, USA, June 2002.

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

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

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

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

[15] U. Priss and E. Jacob. "Utilizing Faceted Structures for Information Systems Design". In *Proceedings of the ASIS Annual Conf. on Knowledge: Creation, Organization, and Use (ASIS'99)*, October 1999.

[16] Uta Priss. "Faceted Knowledge Representation". *Electronic Transactions on Artificial Intelligence*, 4:21–33, 2000. (Available at http://www.ep.liu.se/ej/etai/2000/002/).

[17] 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.

[18] Kenneth A. Ross and Angel Janevski. "Querying Faceted Databases". In *Procs of the 2nd Intern. Workshop on Semantic Web and Databases, SWDB'2004 (satellite of VLDB'04)*, Toronto, Canada, August 2004.

[19] Yannis Tzitzikas. "An Algebraic Method for Compressing Symbolic Data Tables". *Journal of Intelligent Data Analysis (IDA).* (accepted for publication).

[20] 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.

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

[22] 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.

[23] Yannis Tzitzikas, Anastasia Analyti, and Nicolas Spyratos. "The Semantics of the Compound Term Composition Algebra". In *Procs. of the 2nd Intern. Conference on Ontologies, Databases and Applications of Semantics, ODBASE'2003*, pages 970–985, Catania, Sicily, Italy, November 2003.

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

[25] 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.

[26] 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/).

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

[28] M. Winslett. *Updating Logical Databases.* Cambridge University Press, 1990.

A An example of CTCA

Valid	
Feta, Greece	Feta, Europe
Feta, Earth	Cheese, Greece
Cheese, Europe	Cheese, Earth
Ingredients, Greece	Ingredients, Europe
Ingredients, Earth	Roquefort, France
Roquefort, Europe	Roquefort, Earth
Cheese, France	Ingredients, France
Truffle, France	Truffle, Europe
Truffle, Earth	Truffle, Italy
Cheese, Italy	Ingredients, Italy
Cheese, Japan	Cheese, Asia
Ingredients, Japan	Ingredients, Asia
Feta, Oven	Feta, Wok
Feta, C.Style	Roquefort, Oven
Roquefort, Wok	Roquefort, C.Style
Cheese, Oven	Cheese, Wok
Cheese, C.Style	Truffle, Oven
Truffle, Wok	Truffle, C.Style
Ingredients, Oven	Ingredients, Wok
Ingredients, C.Style	Greece, Oven
Greece, C.Style	Italy, Oven
Italy, C.Style	France, Oven
France, C.Style	Europe, Oven
Europe, C.Style	Earth, Oven
Earth, Wok	Earth, C.Style
Japan, Oven	Japan, Wok
Japan, C.Style	Asia, Oven
Asia, Wok	Asia, C.Style
Feta, Greece, Oven	Feta, Greece, C.Style
Feta, Europe, Oven	Feta, Europe, C.Style
Feta, Earth, Oven	Feta, Earth, Wok
Feta, Earth, C.Style	Cheese, Greece, Oven
Cheese, Greece, C.Style	Cheese, Europe, Oven
Cheese, Europe, C.Style	Cheese, Earth, Oven
Cheese, Earth, Wok	Cheese, Earth, C.Style
Ingredients, Greece, Oven	Ingredients, Greece, C.Style
Ingredients, Europe, Oven	Ingredients, Europe, C.Style
Ingredients, Earth, Oven	Ingredients, Earth, Wok
Ingredients, Earth, C.Style	Roquefort, France, Oven
Roquefort, France, C.Style	Roquefort, Europe, Oven
Roquefort, Europe, C.Style	Roquefort, Earth, Oven
Roquefort, Earth, Wok	Roquefort, Earth, C.Style
Cheese, France, Oven	Cheese, France, C.Style
Ingredients, France, Oven	Ingredients, France, C.Style
Truffle, France, Oven	Truffle, France, C.Style
Truffle, Europe, Oven	Truffle, Europe, C.Style
Truffle, Earth, Oven	Truffle, Earth, Wok
Truffle, Earth, C.Style	Truffle, Italy, Oven
Truffle, Italy, C.Style	Cheese, Italy, Oven
Cheese, Italy, C.Style	Ingredients, Italy, Oven
Ingredients, Italy, C.Style	Cheese, Japan, Oven
Cheese, Japan, Wok	Cheese, Japan, C.Style
Cheese, Asia, Oven	Cheese, Asia, Wok
Cheese, Asia, C.Style	Ingredients, Japan, Oven
Ingredients, Japan, Wok	Ingredients, Japan, C.Style
Ingredients, Asia, Oven	Ingredients, Asia, Wok
Ingredients, Asia, C.Style	

Invalid	
Feta, Italy	Feta, France
Feta, Japan	Feta, Asia
Roquefort, Greece	Roquefort, Italy
Roquefort, Japan	Roquefort, Asia
Truffle, Greece	Truffle, Japan
Truffle, Asia	Europe, Wok
Greece, Wok	Italy, Wok
France, Wok	Feta, Greece, Wok
Feta, Europe, Wok	Cheese, Greece, Wok
Cheese, Europe, Wok	Ingredients, Greece, Wok
Ingredients, Europe, Wok	Roquefort, France, Wok
Roquefort, Europe, Wok	Cheese, France, Wok
Truffle, France, Wok	Truffle, Europe, Wok
Truffle, Italy, Wok	Cheese, Italy, Wok
Ingredients, Italy, Wok	Feta, Italy, Oven
Feta, Italy, Wok	Feta, Italy, C.Style
Feta, France, Oven	Feta, France, Wok
Feta, France, C.Style	Feta, Japan, Oven
Feta, Japan, Wok	Feta, Japan, C.Style
Feta, Asia, Oven	Feta, Asia, Wok
Feta, Asia, C.Style	Roquefort, Greece, Oven
Roquefort, Greece, Wok	Roquefort, Greece, C.Style
Roquefort, Italy, Oven	Roquefort, Italy, Wok
Roquefort, Italy, C.Style	Roquefort, Japan, Oven
Roquefort, Japan, Wok	Roquefort, Japan, C.Style
Roquefort, Asia, Oven	Roquefort, Asia, Wok
Roquefort, Asia, C.Style	Truffle, Greece, Oven
Truffle, Greece, Wok	Truffle, Greece, C.Style
Truffle, Japan, Oven	Truffle, Japan, Wok
Truffle, Japan, C.Style	Truffle, Asia, Oven
Truffle, Asia, Wok	Truffle, Asia, C.Style

Table 7: The valid and invalid compound terms of the example of Section 1

As the facet *Ingredients* has 5 terms, the facet *LocationOfOrigin* has 7 terms, and the facet *CookingStyle* has 3 terms, the number of compound terms that contain at most 1 term from each facet is 6*8*4 = 192. This table contains 113 valid and 62 invalid compound terms, thus 175 in total. By adding the (5+7+3=15) singletons (which were omitted from the column of valid) and the empty set we reach the 192 compound terms.