An Algebra for Specifying Compound Terms in Faceted Taxonomies

Yannis Tzitzikas [1,5], Anastasia Analyti[2], Nicolas Spyratos[3], Panos Constantopoulos[2,4]

[1] *Istituto di Scienza e Tecnologie dell' Informazione, CNR-ISTI, Italy*
[2] *Institute of Computer Science, ICS-FORTH, Greece*
[3] *Laboratoire de Recherche en Informatique, Universite de Paris-Sud, France*
[4] *Department of Computer Science, University of Crete, Greece*

Email : tzitzik@isti.cnr.it, {analyti, panos}@ics.forth.gr, spyratos@lri.fr

Abstract. A faceted taxonomy is a set of taxonomies, each describing a given domain from a different aspect, or facet. The indexing of domain objects is done through conjunctive combinations of terms from the facets, called compound terms. A faceted taxonomy has several advantages over a single hierarchy of terms, including conceptual clarity, compactness and scalability. A drawback, however, is the cost of avoiding invalid combinations, i.e. compound terms that do not apply to any object in the domain. This need arises in both indexing and retrieval, and typically involves human effort for specifying the valid compound terms one by one. We here propose a compound term composition algebra which can be used to generate valid compound terms in a given faceted taxonomy in an efficient and flexible manner. It works on the basis of the original simple terms of the facets and a small set of positive and/or negative statements. In each algebraic operation, we adopt a closed-world assumption with respect to the declared positive or negative statements. The taxonomy algebra can be exploited in dynamically generating navigation trees, a significant browsing aid.

1 Introduction

There are several application areas where a *taxonomy* is used for indexing the objects of a knowledge domain (e.g. documents, books, product descriptions, Web pages). For instance, Web catalogs, such as Yahoo! or Open Directory, use taxonomies, for indexing the pages of the Web. These catalogs turn out to be very useful for browsing and querying the Web. Although they index only a fraction of the pages that are indexed by search engines using statistical methods (e.g. Google, AltaVista), they are hand-crafted by domain experts and are therefore of high quality. Recently, the various search engines have begun to exploit these catalogs in order to enhance the quality of retrieval and also to offer new functionalities. Specifically, search engines now employ catalogs for computing "better" degrees of relevance, and for determining (and presenting to the user) a set of relevant pages for each page in the answer set. In addition, some search engines (e.g. Google) now employ taxonomies in order to enable limiting the scope (or defining the context) of search. For example, using Google, one can first select a category, e.g. `Sciences/CS/DataStructures`, from the taxonomy of Open Directory and then submit a natural language query, e.g. "`Tree`". The search engine will compute the degree of relevance, with respect to the natural language query, "`Tree`", only of those pages that fall in the category `Sciences/CS/DataStructures` in the catalog of Open

[5]Work done during the postdoctoral studies of the author at CNR-ISTI as an ERCIM fellow. The first part of this work was done when the author was at ICS-FORTH.

Directory. Clearly, this enhances the precision of retrieval and reduces the computational cost (e.g. see [8], [5]).

A taxonomy is a hierarchically-organized set of terms. In designing a taxonomy, one has to define (a priori) appropriate terms and their subterms, according to various criteria. One basic criterion is that each term must be *valid*, in the sense that it applies to, or *indexes*, at least one object of the underlying domain. However, as pointed out long ago [7], the design of a taxonomy can be done in a more convenient and a more systematic manner, if we first identify a number of different aspects, or facets, of the domain and then design one taxonomy per aspect. This process results in a *faceted taxonomy*, i.e. a *set* of taxonomies, called *facets*.

For example, assume that the domain of interest is a set of hotel Web pages in Greece, and suppose that we want to provide access to these pages according to the *Location* of the hotels and the *Sports* facilities they offer. Figure 1 shows these two facets. Each object is described using a *compound term*, i.e., a set of terms containing one or more terms from each facet. For example, a hotel in Crete providing sea ski and wind-surfing facilities would be described by the compound term $\{Crete, SeaSki, Windsurfing\}$.

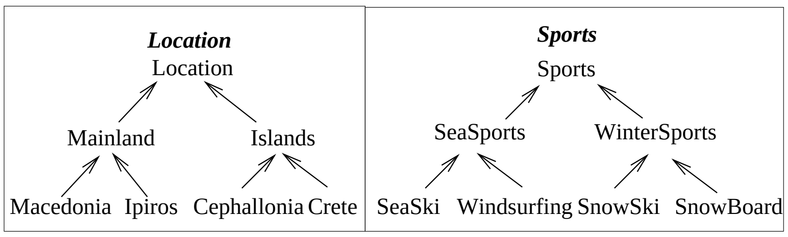

Figure 1: Two facets

The use of a faceted taxonomy, i.e. of a set of taxonomies, instead of a single taxonomy, for indexing the objects of interest, has several consequences. For example, consider two schemes for describing the objects of a domain, one using a single taxonomy consisting of 100 terms, and the other using a faceted taxonomy consisting of 10 facets each having 10 terms. The first scheme has 100 indexing terms while the second has 10^{10}, i.e. 10 billion, compound indexing terms! Although both schemes have the same storage requirements, i.e. each one requires storing 100 terms, the indexing terms of the second scheme are tremendously more than the indexing terms of the first.

Overall, a faceted taxonomy has several advantages by comparison to a single hierarchical taxonomy, such as conceptual clarity, compactness and scalability (e.g. see [6]). Unfortunately, faceted taxonomies also have a serious drawback. Indeed, assuming that each facet has been designed correctly, every single term will be valid. However, even if this assumption holds for every term in every facet, it may not hold for every conceivable compound term. That is, there may exist invalid compound terms, in the sense that there is no object of the underlying domain indexed by all of their terms. For example, it may very well be that the terms $Crete$ (from $Location$) and $SnowBoard$ (from $Sports$) are each valid, i.e., there are hotels located in Crete and there are hotels that offer snow-board facilities. However, this does not guarantee that the compound term $\{Crete, SnowBoard\}$ is valid. In fact, there is no hotel in Crete offering snow-board facilities. It follows that not all compound terms of a faceted taxonomy are valid, even if each term of every facet is valid.

Invalid compound terms cause serious problems in indexing and browsing that prevent the design and deployment of faceted taxonomies for real and large scale applications. Being able to infer the valid compound terms of a faceted taxonomy would be very useful. It could be exploited in the indexing process in order to aid the indexer and prevent indexing errors. Such an aid is especially important in cases where the indexing is done by many people who are not domain experts. For example, the indexing of Web pages in the Open Directory (which is used by Netscape, Lycos, HotBot and several other search engines) is done by more than 20.000 volunteer human editors (indexers). On the other hand, the inability to infer valid compound terms may give rise to problems in browsing, as an invalid term will yield no objects. However, if we could infer the valid compound terms in a faceted taxonomy then we would be able to generate navigation trees *on the fly*, having only valid compound terms as nodes.

The main goal of this paper is precisely to propose an algebra whose operators allow the efficient and flexible specification of compound terms, thus alleviating the main drawback of faceted taxonomies. Following our approach, given a faceted taxonomy, one can use an *algebraic expression* to define the desired set of compound terms. In each algebraic operation, the designer has to declare either a small set of valid compound terms from which other valid compound terms are inferred, or a small set of invalid compound terms from which other invalid compound terms are inferred. Then, a closed-world assumption is adopted for the rest of the compound terms. In our example, this means that the designer can specify the large number of valid compound terms of a faceted taxonomy by providing a relatively small number of (valid or invalid) compound terms. This is an important feature as it minimizes the effort needed by the designer. A distinctive feature of our approach is that there is no need to store the set of compound terms defined by the expression. We only have to store the defining expression as we provide an inference mechanism which can check whether a compound term belongs to the result of an expression. Thus the compound taxonomies defined by our algebra have low storage space requirements.

The remaining of this paper is organized as follows: Section 2 describes formally taxonomies, compound taxonomies and facets. Section 3 describes the proposed algebra, and Section 4 illustrates its application by an example. Section 5 provides an inference mechanism for checking whether a compound term belongs to the compound taxonomy defined by an algebraic expression. Section 6 describes a mechanism for generating navigation trees. Finally, Section 7 discusses applications and concludes the paper.

2 Taxonomies, Compound Taxonomies and Facets

Def 2.1 A *terminology* is a finite set of names, called *terms*.

Def 2.2 A *taxonomy* is a pair $(\mathcal{T}, \leq)$, where $\mathcal{T}$ is a *terminology* and $\leq$ is a reflexive and transitive relation over $\mathcal{T}$, called *subsumption*.

If a and b are terms of $\mathcal{T}$ and $a \leq b$ then we say that a is *subsumed* by b, or that b *subsumes* a. We also say that a is *narrower than* b, or that b is *broader* than a. For example, `Databases` $\leq$ `Informatics`. We say that two terms a and b are *equivalent*, and write $a \sim b$, if both $a \leq b$ and $b \leq a$ hold, e.g., `Computer Science` $\sim$ `Informatics`. Note that the subsumption relation is a preorder over $\mathcal{T}$ and that $\sim$ is an equivalence relation over the terms of $\mathcal{T}$. Moreover $\leq$ is a partial order over the equivalence classes of terms.

When using diagrams to depict a taxonomy, such as the ones of Figure 1, term subsumption is indicated by a continuous-line arrow from the narrower term to the broader term. Note that we do not represent the entire subsumption relation but only a subset sufficient for generating it. In particular, we do not represent the reflexive nor the transitive arrows of the subsumption relation. Equivalence of terms is indicated by a continuous non-oriented line segment. In what follows, we shall often write $\mathcal{T}$ instead of $(\mathcal{T}, \leq)$, whenever no ambiguity is possible.

We now introduce the concept of compound taxonomy, a basic concept for the rest of this paper. First, we define compound terms over a given taxonomy and their ordering. In all definitions that follow, we assume an underlying taxonomy $(\mathcal{T}, \leq)$.

Def 2.3 A *compound term* over $\mathcal{T}$ is any subset of $\mathcal{T}$.

For example, the following sets of terms are compound terms over the taxonomy *Sports* of Figure 1: $s_1 = \{SeaSki, Windsurfing\}$, $s_2 = \{SeaSports, WinterSports\}$, $s_3 = \{Sports\}$, and $s_4 = \emptyset$.

We denote by $P(\mathcal{T})$ the set of all compound terms over $\mathcal{T}$ (i.e. the powerset of $\mathcal{T}$).

Def 2.4 A *compound terminology* S over $\mathcal{T}$ is any set of compound terms that contains the compound term $\emptyset$.

Clearly, $P(\mathcal{T})$ is a compound terminology over $\mathcal{T}$. The set of all compound terms over $\mathcal{T}$ can be ordered using the following ordering derived from $\leq$.

Def 2.5 Let s, s' be two compound terms over $\mathcal{T}$. The *compound ordering* over $\mathcal{T}$ is defined as follows: $s \preceq s'$ iff $\forall t' \in s' \ \exists t \in s$ such that $t \leq t'$

That is, $s \preceq s'$ iff s contains a narrower term for every term of s'. In addition, s may contain terms not present in s'. Roughly, $s \preceq s'$ means that s carries more specific information than s'. Figure 2.(a) shows the compound ordering over the compound terms of our previous example. Note that $s_1 \preceq s_3$, as s_1 contains $SeaSki$ which is a term narrower than the unique term $Sports$ of s_3. On the other hand, $s_1 \not\preceq s_2$, as s_1 does not contain a term narrower than $WinterSports$. Finally, $s_2 \preceq s_3$ and $s_3 \preceq \emptyset$. In fact, $s \preceq \emptyset$, for every compound term s.

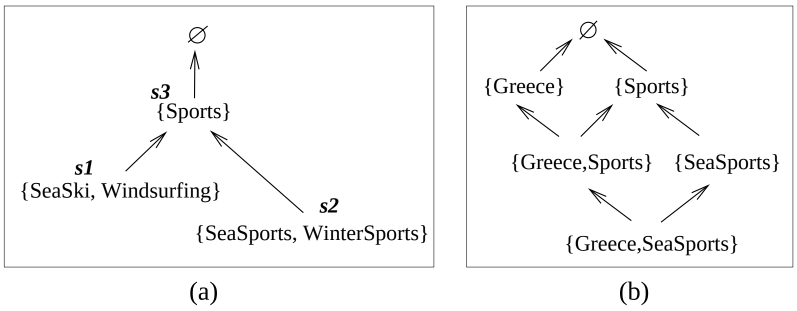

Figure 2: Two examples of compound taxonomies

Clearly, $\preceq$ is a reflexive and transitive relation over S. Also note that while the relation $\leq$ is provided explicitly by the designer of the taxonomy $\mathcal{T}$, the relation $\preceq$ is *derived* from $\leq$ according to the previous definition.
We say that two compound terms s, s' are *equivalent* iff $s \preceq s'$ and $s' \preceq s$. For example, $\{SeaSki, SeaSports\}$ and $\{SeaSki\}$ are equivalent. Intuitively, equivalent compound terms carry the same information.

Def 2.6 A *compound taxonomy* over $\mathcal{T}$ is a pair $(S, \preceq)$, where S is a compound terminology over $\mathcal{T}$, and $\preceq$ is the compound ordering over $\mathcal{T}$ restricted to S.

Figure 2 shows two example compound taxonomies.
Clearly, $(P(\mathcal{T}), \preceq)$ is a compound taxonomy over $\mathcal{T}$. Let s be a compound term. The broader and the narrower compound terms of s are defined as follows:

$$\begin{aligned}\mathrm{Br}(s) &= \{s' \in P(\mathcal{T}) \mid s \preceq s'\} \\ \mathrm{Nr}(s) &= \{s' \in P(\mathcal{T}) \mid s' \preceq s\}\end{aligned}$$

Let S be a compound terminology over $\mathcal{T}$. The broader and the narrower compound terms of S are defined as follows:

$$\begin{aligned}Br(S) &= \cup\{\mathrm{Br}(s) \mid s \in S\} \\ Nr(S) &= \cup\{\mathrm{Nr}(s) \mid s \in S\}\end{aligned}$$

As already mentioned, one way of designing a taxonomy is by identifying a number of different aspects of the domain of interest and then designing one taxonomy per aspect. As a result we obtain a set of taxonomies called *facets*. Given a set of facets we can define a *faceted taxonomy*.

Def 2.7 Let $\{F_1, ..., F_k\}$ be a finite set of taxonomies, where $F_i = (\mathcal{T}_i, \leq_i)$, and assume that the terminologies $\mathcal{T}_1$, ... ,$\mathcal{T}_k$ are pairwise disjoint. Then the pair $\mathcal{F} = (\mathcal{T}, \leq)$, where $\mathcal{T} = \bigcup_{i=1}^{k} \mathcal{T}_i$ and $\leq \; = \bigcup_{i=1}^{k} \leq_i$, is a taxonomy which we shall call the *faceted taxonomy generated* by $\{F_1, ..., F_k\}$. We shall call the taxonomies $F_1, ..., F_k$ the *facets* of $\mathcal{F}$.

It is common practice to refer to a facet through its top term. For example, we refer to the facets of Figure 1 as *Location* and *Sports*. Clearly, all definitions introduced so far apply also to faceted taxonomies. In particular, compound terms can be derived from a faceted taxonomy. For example, the set $S = \{\{Greece\}, \{Sports\}, \{SeaSports\}, \{Greece, Sports\}, \{Greece, SeaSports\}, \emptyset\}$, is a compound terminology over the terminology $\mathcal{T}$ of the faceted taxonomy shown in Figure 1. The set S together with the compound ordering of $\mathcal{T}$ (restricted to S) is a compound taxonomy over $\mathcal{T}$. This compound taxonomy is shown in Figure 2.(b). For reasons of brevity, hereafter we shall omit the term $\emptyset$ from the compound terminologies of our examples and figures.

We say that a compound term s is *valid* (resp. *invalid*), if there is at least one (resp. no) object of the underlying domain indexed by all terms in s. We assume that every term of $\mathcal{T}$ is valid. However, a compound term over $\mathcal{T}$ may be invalid. Obviously, if s is a valid compound term, all compound terms in $\mathrm{Br}(s)$ are valid. Additionally, if s is an invalid compound term, all compound terms in $\mathrm{Nr}(s)$ are invalid.

3 The Compound Term Composition Algebra

Let $\mathcal{F} = (\mathcal{T}, \leq)$ be the faceted taxonomy generated by a given set of facets $\{F_1, ..., F_k\}$. The problem is that $\mathcal{F}$ does not itself specify which compound terms, i.e. which elements of $P(\mathcal{T})$, are valid and which are not. To alleviate this problem, we introduce a method for defining a compound terminology over $\mathcal{T}$ (i.e. a subset of $P(\mathcal{T})$) which consists of the desired compound terms, i.e. those that the designer considers as valid.

The main tool for accomplishing this task is an algebra that we now define. To begin with we associate the terminology $\mathcal{T}_i$ of every facet with a compound terminology T_i that we call the *basic compound terminology* of $\mathcal{T}_i$.

Def 3.1 Let $F_i = (\mathcal{T}_i, \leq)$ be a facet. The *basic compound terminology* of $\mathcal{T}_i$ is the compound terminology:

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

As every term t of a facet is considered valid, all compound terms in $\mathrm{Br}(\{t\})$ are valid. Thus, T_i is the set of compound terms over $\mathcal{T}_i$ that are initially known to be valid. We use the basic compound terminologies as the "building blocks" of our algebra.

Let $\mathcal{S}$ denote the set of all compound terminologies over $\mathcal{T}$. We define an algebra over S, which includes four operations and compound terminologies as operands. Compound terms can be formed by combining terms from different facets, but also terms from the same facet. A binary product operation and a unary self-product operation are defined to generate term combinations respectively. Since not all term combinations are valid, the issue is how to employ available domain knowledge in order to specify only valid compound terms. Such knowledge may be available in positive or negative form: combinations known to be valid or invalid. The issue is dealt by defining more general operations that include positive or negative modifiers, which are sets of known valid or known invalid compound terms. The unmodified product and self-product operations turn out to be special cases with the modifiers at certain extreme values. Thus, the four operations of the algebra are: *plus-product, minus-product, plus-self-product* and *minus-self product.*

For defining the desired compound taxonomy the designer has to formulate an algebraic expression e, using these operations and initial operands the basic compound terminologies $\{T_1, .., T_k\}$.

Before we describe each operation in detail, we define the auxiliary binary operation $\oplus$ over $\mathcal{S}$, i.e. $\oplus : \mathcal{S} \times \mathcal{S} \to \mathcal{S}$, called *product*.

Def 3.2 Let S and S' be two compound terminologies ($S, S' \in \mathcal{S}$). The *product* of S and S', denoted by $S \oplus S'$, is defined as follows:

$$S \oplus S' = \{s \cup s' \mid s \in S,\ s' \in S'\}$$

This operation results in an "unqualified" compound terminology whose compound terms are all possible unions of compound terms from its arguments. The compound terms of the result are ordered according to the compound ordering (see Definition 2.5). For example, consider the compound terminologies $S = \{\{Greece\}, \{Islands\}\}$ and $S' = \{\{Sports\}, \{SeaSports\}\}$. The compound taxonomy corresponding to $S \oplus S'$ is shown in Figure 3, and consists of 8 terms. Recall that for reasons of brevity, we omit the term $\emptyset$ from the compound terminologies of our examples ($\emptyset$ is an element of S, S' and $S \oplus S'$). It can be easily seen that the product operation is commutative and associative and that it can be easily extended to an n-ary operation: $S_1 \oplus ... \oplus S_n = \{\ s_1 \cup ... \cup s_n \mid s_i \in S_i\}$. Additionally, as $\emptyset \in S, S'$, it holds that $S, S' \subseteq S \oplus S'$.

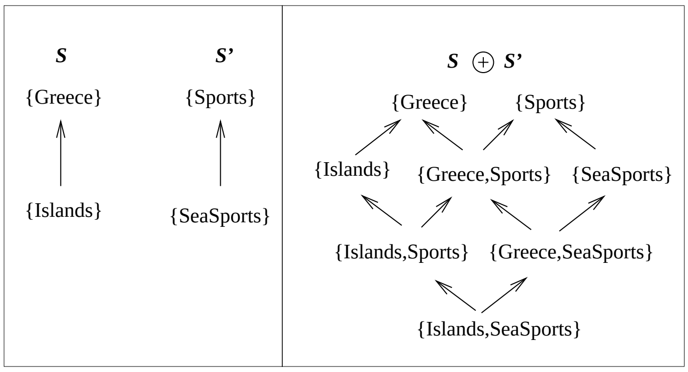

Figure 3: An example of a product $\oplus$ operation

Below we describe each operation of our algebra in detail.

3.1 The *plus-product* and the *minus-product* operations

Consider the compound terminologies S and S' shown in the upper part of Figure 4, and suppose we want to define a compound terminology that does not contain the compound terms $\{Islands, WinterSports\}$ and $\{Islands, SnowSki\}$, because they are invalid. For this purpose we introduce two "variations" of the $\oplus$ operation, namely the *plus-product* and the *minus-product*. Each of these two operations has an extra parameter denoted by P and N, respectively. The set P is a set of compound terms that we certainly want to appear in the result of the operation, i.e. they are valid. On the other hand, the set N is a set of compound terms that we certainly do not want to appear in the result of the operation, i.e. they are invalid.

To proceed we need to distinguish what we shall call *genuine compound terms*. Intuitively, a genuine compound term combines non-empty compound terms from more than one compound terminologies.

Def 3.3 The set of *genuine* compound terms over a set of compound terminologies $S_1, ..., S_n$, denoted by $G_{S_1,...,S_n}$, is defined as follows:

$$G_{S_1,...,S_n} = S_1 \oplus ... \oplus S_n - \bigcup_{i=1}^{n} S_i$$

For example if $S_1 = \{\{Greece\}, \{Islands\}\}$, $S_2 = \{\{Sports\}, \{WinterSports\}\}$, and $S_3 = \{\{Pensions\}, \{Hotels\}\}$ then

$$\begin{aligned}\{Greece, WinterSports, Hotels\} &\in G_{S_1,S_2,S_3},\\ \{WinterSports, Hotels\} &\in G_{S_1,S_2,S_3} \text{ , but}\\ \{Hotels\} &\not\in G_{S_1,S_2,S_3}\end{aligned}$$

Assume that the compound terms of $S_1, ..., S_n$ are valid. We are interested in characterizing the validity of all combinations of compound terms of $S_1, ..., S_n$. As we already know the validity of the compound terms of $S_1, ..., S_n$, we are basically interested in characterizing the validity of the compound terms in $G_{S_1,...,S_n}$. This is done through the following operations, plus-product and minus-product.

We can now define the *plus-product* operation, $\oplus_P$, an n-ary operation over $\mathcal{S}$ ($\oplus_P : \mathcal{S} \times ... \times \mathcal{S} \to \mathcal{S}$), where the parameter P is a set of valid compound terms from the product of the input compound terminologies. The set P is a subset of $G_{S_1,...,S_n}$ (i.e., $P \subseteq G_{S_1,...,S_n}$), as we already know that all compound terms in $\bigcup_{i=1}^{n} S_i$ are valid.

Def 3.4 Let $S_1, ..., S_n$ be compound terminologies and $P \subseteq G_{S_1,...,S_n}$. The *plus-product* of $S_1, ..., S_n$ with respect to P, denoted by $\oplus_P(S_1, ..., S_n)$, is defined as follows:

$$\oplus_P(S_1, ...S_n) = S_1 \cup ... \cup S_n \ \cup \ Br(P)$$

This operation results in a compound terminology consisting of the compound terms of the initial compound terminologies, *plus* the compound terms which are broader than an element of P. This is because, all compound terms in $S_1 \cup ... \cup S_n \ \cup \ Br(P)$ are valid. By adopting a closed-world assumption, all compound terms in $S_1 \oplus ... \oplus S_n - \oplus_P(S_1, ...S_n) = G_{S_1,...,S_n} - Br(P)$ are invalid.

For example, consider the compound terminologies S and S' of Figure 4 and suppose that $P = \{\{Islands, Seasports\}, \{Greece, SnowSki\}\}$. The compound taxonomy of the operation $\oplus_P(S, S')$ is shown in Figure 4. In this figure we enclose in squares the elements of P. We see that the compound terminology $\oplus_P(S, S')$ contains the compound term $s = \{Greece, Sports\}$, as $s \in Br(\{Islands, SeaSports\})$. However, it does not contain the compound terms $\{Islands, WinterSports\}$ and $\{Islands, SnowSki\}$ as they do not belong to $S \cup S' \cup Br(P)$.

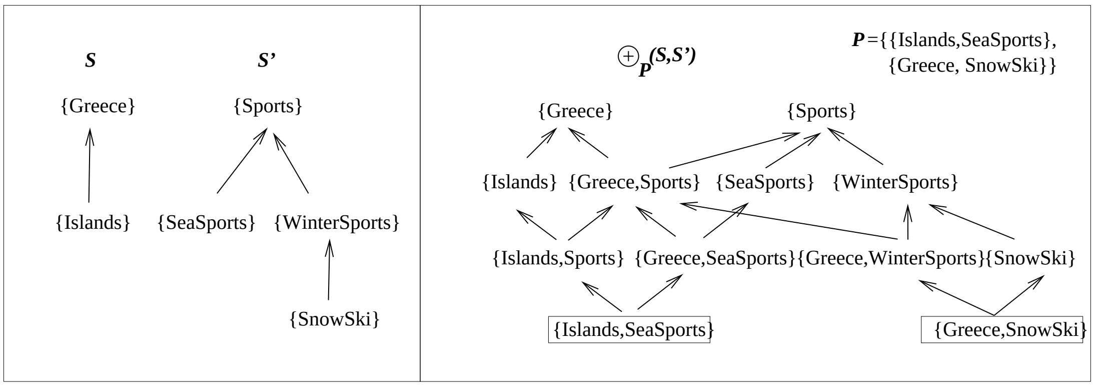

Figure 4: An example of a *plus-product*, $\oplus_P$, operation

The following proposition gives two simplifications of the operation for the two extreme values of the P parameter. The first property says that the product is a special case of the plus-product, while the second property says that if $P = \emptyset$ then the plus-product operation defines a compound terminology that contains only the compound terms of the operands.

Prop. 3.1 Given compound terminologies S_i, $i = 1, ..., n$,

(1) If $P = G_{S_1,...,S_n}$ then $\oplus_P(S_1, ..., S_n) = S_1 \oplus ... \oplus S_n$, and

(2) if $P = \emptyset$ then $\oplus_P(S_1, ..., S_n) = \bigcup_{i=1}^n S_i$.

Now we define the *minus-product* operation, $\ominus_N$, an n-ary operation over $\mathcal{S}$ ($\ominus_N : \mathcal{S}\times...\times\mathcal{S} \rightarrow \mathcal{S}$), where the parameter N is a set of invalid compound terms from the product of the input compound terminologies. The set N is a subset of $G_{S_1,...,S_n}$ (i.e., $N \subseteq G_{S_1,...,S_n}$), as all compound terms in $\bigcup_{i=1}^n S_i$ are valid.

Def 3.5 Let $S_1, ..., S_n$ be compound taxonomies and $N \subseteq G_{S_1,...,S_n}$. The *minus-product* of $S_1, ..., S_n$ with respect to N, denoted by $\ominus_N(S_1, ..., S_n)$, is defined as follows:

$$\ominus_N(S_1, ...S_n) = S_1 \oplus ... \oplus S_n - Nr(N)$$

This operation results in a compound terminology consisting of all compound terms in the product of the initial compound terminologies, *minus* all compound terms which are narrower than an element of N. This is because, all compound terms in $Nr(N)$ are invalid. By adopting a closed-world assumption, all compound terms in $\ominus_N(S_1, ...S_n) = S_1 \oplus ... \oplus S_n - Nr(N)$ are valid.

For example, consider the compound terminologies S and S' of the previous example and suppose that $N = \{\{Islands, WinterSports\}\}$. The result of the operation $\ominus_N(S, S')$ is shown in Figure 5. We see that the compound terminology $\ominus_N(S, S')$ does not contain the compound terms $\{Islands, WinterSports\}$ and $\{Islands, SnowSki\}$, as they are elements of $Nr(N)$. Notice that the compound taxonomies of figures 4 and 5 coincide. These examples demonstrate two alternative ways of defining the desired compound taxonomy.

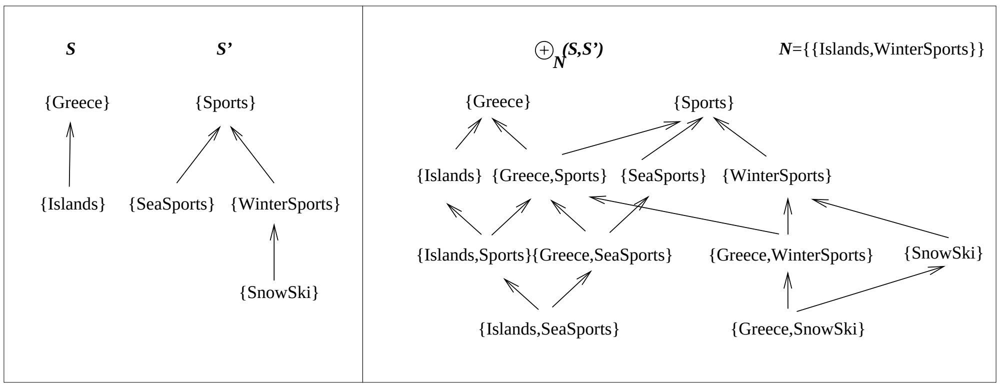

Figure 5: An example of a *minus-product*, $\ominus_N$, operation

The following proposition gives two simplifications of the operation for the two extreme values of the N parameter. Note that these two simplifications are the opposite of these of the $\oplus_P$ operation, given in Proposition 3.1.

Prop. 3.2 Given the compound terminologies S_i, $i = 1, ..., n$,

(1) If $N = G_{S_1,...,S_n}$ then $\ominus_N(S_1, ..., S_n) = \bigcup_{i=1}^n S_i$, and

(2) if $N = \emptyset$ then $\ominus_N(S_1, ..., S_n) = S_1 \oplus ... \oplus S_n$.

3.2 The *Self-product* operations

The operators introduced so far allow defining a compound terminology which consists of compound terms that contain at most one compound term from each basic compound terminology. However, a valid compound term may contain any set of terms of the same facet

(multiple classification). To capture such cases, we define the *self-product*, $\overset{*}{\oplus}$, a unary operation which gives all possible compound terms of one facet. Subsequently, we shall modify this operation with the parameters P and N.
Let $\mathcal{BS}$ be the set of basic compound terminologies, that is $\mathcal{BS}= \{T_1, ..., T_k\}$.

Def 3.6 Let T_i be a basic compound terminology. The *self-product* of T_i, denoted by $\overset{*}{\oplus}(T_i)$, is defined as: $\overset{*}{\oplus}(T_i) = P(T_i)$.

In the above definition, $P(T_i)$ denotes the powerset of the terminology T_i, not the powerset of the basic compound terminology T_i.
For example, consider the facet *Sports* of Figure 1. The compound terms $\{SeaSports, WinterSports\}$ and $\{SeaSki, Windsurfing, WinterSports\}$ are elements of $\overset{*}{\oplus}(Sports)$.
The notion of genuine compound terms is also necessary here.

Def 3.7 The set of *genuine* compound terms over a basic compound terminology T_i, denoted by G_{T_i}, is defined as follows: $G_{T_i} = \overset{*}{\oplus}(T_i) - T_i$

Now we define the *plus-self-product* operation, $\overset{*}{\oplus}_P$, a unary operation ($\overset{*}{\oplus}_P$: $\mathcal{BS} \rightarrow \mathcal{S}$) where the parameter P is a set of compound terms that we want to appear in the result of the operation. The set P is a subset of G_{T_i}.

Def 3.8 Let T_i be a basic compound terminology and $P \subseteq G_{T_i}$. The *plus-self-product* of T_i with respect to P, denoted by $\overset{*}{\oplus}_P(T_i)$, is defined as follows:

$$\overset{*}{\oplus}_P(T_i) = T_i \cup Br(P)$$

This operation results in a compound terminology consisting of the compound terms of the initial basic compound terminology, *plus* all compound terms which are broader than an element of P. For example, the result of the operation $\overset{*}{\oplus}_P(Sports)$, where $P = \{\{SeaSki, Windsurfing\}, \{SnowSki, SkiBoard\}\}$ is shown in Figure 6. The analogous of Prop. 3.1 with regard to the extreme cases of P holds, namely, $\overset{*}{\oplus}_{G_{T_i}}(T_i) = \overset{*}{\oplus} T_i$ and $\overset{*}{\oplus}_{\emptyset} = T_i$.

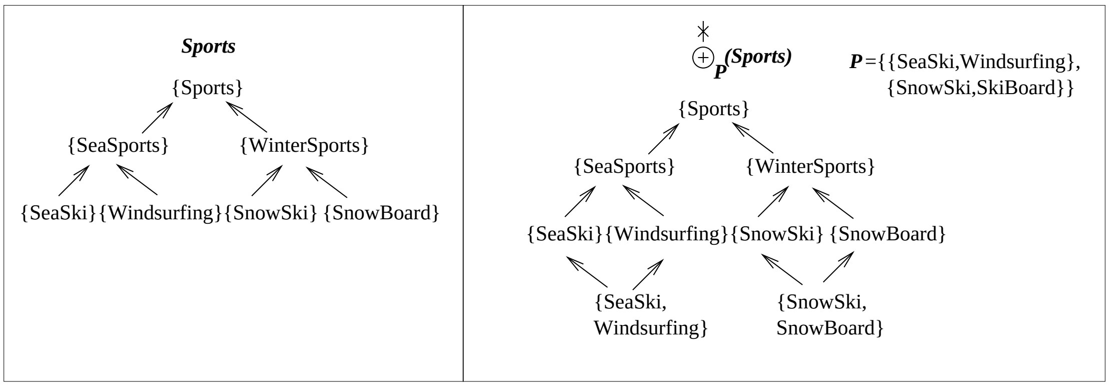

Figure 6: An example of a *plus-self-product*,$\overset{*}{\oplus}_P$, operation

The following definition introduces the *minus-self-product* operation, $\overset{*}{\oplus}_N$, a unary operation ($\overset{*}{\ominus}_N$: $\mathcal{BS} \rightarrow \mathcal{S}$) where the parameter N is a set of compound terms that we do not want to appear in the result of the operation. The set N is a subset of G_{T_i}.

Def 3.9 Let T_i be a basic compound terminology and $N \subseteq G_{T_i}$. The *minus-self-product* of T_i with respect to N, denoted by $\overset{*}{\ominus}_N (T_i)$, is defined as follows:

$$\overset{*}{\ominus}_N (T_i) = \overset{*}{\oplus} (T_i) - Nr(N)$$

This operation results in a compound terminology consisting of all compound terms in the self-product of T_i, *minus* the compound terms which are narrower than an element in N. For example, we can obtain the compound terminology of Figure 6 by the operation $\overset{*}{\ominus}_N$ (*Sports*), where $N = \{\{SeaSports, WinterSports\}\}$. Concerning the extreme cases of N, the analogous of Prop. 3.2 holds: $\overset{*}{\ominus}_{G_{T_i}} (T_i) = T_i$, and $\overset{*}{\ominus}_{\emptyset} (T_i) = \overset{*}{\oplus} T_i$.

3.3 Algebraic Expressions

For defining the desired compound taxonomy, the designer has to formulate an expression e, where an expression is defined as follows:

Def 3.10 An expression over a set of facets $\{F_1, ..., F_k\}$ is defined according to the following grammar:

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

The outcome of the evaluation of an expression e is denoted by S_e and is called the *compound terminology* of e, and any element of S_e is called *compound term* of e. In addition, $(S_e, \preceq)$ is called the *compound taxonomy* of e.
All compound terms in S_e are *valid*, and the rest in $P(\mathcal{T}_e) - S_e$ are *invalid*, where $\mathcal{T}_e$ is the union of the terminologies of the facets appearing in e.

We are interested only in *well-formed* expressions, defined as follows:

Def 3.11 An expression e is *well-formed* iff:

(i) each basic compound terminology T_i appears at most once in e,

(ii) each parameter P that appears in e, is a subset of the associated set of genuine compound terms, and

(iii) each parameter N that appears in e, is a subset of the associated set of genuine compound terms.

For example, the expression $(T_1 \oplus_P T_2) \ominus_N T_1$ is not well-formed as T_1 appears twice in the expression.
Constraints (i), (ii), and (iii) ensure that we have *no conflicts*, meaning that the valid and invalid compound terms of an expression e increase as the length of e increases. For example, if we omit constraint (i) then an invalid compound term according to an expression $T_1 \oplus_P T_2$ could be valid according to a larger expression $(T_1 \oplus_{P_1} T_2) \ominus_{P_2} T_1$. If we omit constraint (ii) then an invalid compound term according to an expression $T_1 \oplus_{P_1} T_2$ could be valid according to a larger expression $(T_1 \oplus_{P_1} T_2) \oplus_{P_2} T_3$. Additionally, if we omit constraint (iii) then a valid compound term according to an expression $T_1 \oplus_P T_2$ could be invalid according to a larger expression $(T_1 \oplus_P T_2) \ominus_N T_3$.
This monotonic behaviour in the evaluation of a well-formed expression results in a number of useful properties. In addition, due to their monotonicity, well-formed expressions can be formulated in a systematic, gradual manner (intermediate results of subexpressions are not invalidated by larger expressions).
In this paper, we consider only well-formed expressions.

4 Example

Suppose that the domain of interest is a set of hotel Web pages and that we want to index these pages using a faceted taxonomy. First, we must define the taxonomy. Suppose it is decided to do the indexing according to three facets, namely the *location* of the hotels, the kind of *accommodation*, and the *facilities* they offer. Specifically, assume that the designer employs (or designs from scratch) the facets shown in Figure 7.

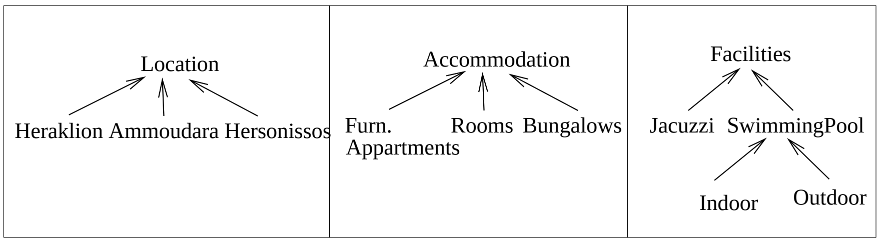

Figure 7: Three-facets

The faceted taxonomy has 13 terms ($|\mathcal{T}|$=13) and $P(\mathcal{T})$ has 890 compound terms[6]. However, available domain knowledge suggests that only 96 compound terms are valid. Omitting the compound terms which are singletons or contain top terms of the facets, the following 23 valid compound terms remain:

$$\begin{aligned}
&\{Heraklion, Furn.Appartments,\}, \{Heraklion, Rooms\},\\
&\{Ammoudara, Furn.Appartments\}, \{Ammoudara, Rooms\},\\
&\{Ammoudara, Bungalows\}, \{Hersonissos, Furn.Appartments\},\\
&\{Hersonissos, Rooms\}, \{Hersonissos, Bungalows\},\\
&\{Hersonissos, SwimmingPool\}, \{Hersonissos, Indoor\},\\
&\{Hersonissos, Outdoor\}, \{Ammoudara, Jacuzzi\},\\
&\{Rooms, SwimmingPool\}, \{Rooms, Indoor\},\\
&\{Bungalows, SwimmingPool\}, \{Bungalows, Outdoor\},\\
&\{Bungalows, Jacuzzi\}, \{Hersonissos, Rooms, SwimmingPool\},\\
&\{Hersonissos, Rooms, Indoor\}, \{Hersonissos, Bungalows, SwimmingPool\},\\
&\{Hersonissos, Bungalows, Outdoor\},\ \{Ammoudara, Bungalows, Jacuzzi\}.
\end{aligned}$$

Rather than being explicitly enumerated, the 96 valid compound terms can be algebraically specified. In this way, the specification of the desired compound terms can be done in a systematic, gradual, and easy manner. For example, the following plus-product operation can be used:

$$\oplus_P(Location, Accommodation, Facilities), \text{ where}$$

$$\begin{aligned}
P = \quad &\{\{Heraklion, Furn.Appartments\},\\
&\{Heraklion, Rooms\},\\
&\{Ammoudara, Furn.Appartments\},\\
&\{Ammoudara, Rooms\},\\
&\{Hersonissos, Furn.Appartments\},
\end{aligned}$$

[6]Recall that equivalent compound terms are considered the same. Thus, $|P(\mathcal{T})|$ is not 2^{13} but 890. This is computed as follows: It holds that $|\overset{*}{\oplus}(Location)|$=8, $|\overset{*}{\oplus}(Accomodation)| = 8$, and $|\overset{*}{\oplus}(Facilities)| = 10$. Thus, $|P(\mathcal{T})| = |(\overset{*}{\oplus}(Location)) \oplus (\overset{*}{\oplus}(Accomodation)) \oplus (\overset{*}{\oplus}(Facilities)|\,| = (8 + 8*8 + 8*10 + 8*8*10) + (8 + 8*10) + 10 = 890$.

$$\{Ammoudara, Bungalows, Jacuzzi\},$$
$$\{Hersonissos, Rooms, Indoor\},$$
$$\{Hersonissos, Bungalows, Outdoor\}\}$$

Note that the compound terms in P are only 8. Alternatively, the same result can be obtained more efficiently through the expression:

$$(Location \ominus_N Accommodation) \oplus_P Facilities,$$

where

$$\begin{aligned} N = \quad & \{\{Heraklion, Bungalows\}\}, \text{ and} \\ P = \quad & \{\{Hersonissos, Rooms, Indoor\}, \\ & \{Hersonissos, Bungalows, Outdoor\}, \\ & \{Ammoudara, Bungalows, Jacuzzi\}\} \end{aligned}$$

Note that now the total number of compound terms in P and N is just 4. In summary, the faceted taxonomy of our example, includes 13 terms, 890 compound terms, and 96 valid compound terms which can be specified by providing only 4 (carefully selected) compound terms and an appropriate algebraic expression.

Let us now discuss the methodology for formulating the expression e and the corresponding parameters P and N. Consider two facets F and F'. If the majority of the compound terms over these two facets are valid then it is better to use a *minus-product* operation so as to specify only the invalid compound terms. Concerning the defining of the set N, it is more efficient to put in N "short" compound terms that consist of "broad" terms. The reason is that from such compound terms a large number of new invalid compound terms can be inferred. Conversely, if the majority of compound terms are invalid, then it is better to employ a *plus-product* operation so as to specify only the valid compound terms. Concerning the definition of the set P, it is more efficient to put in P "long" compound terms that consist of "narrow" terms, since from such compound terms a large number of new valid compound terms can be inferred.

5 Checking the Validity of a Compound Term

We now turn to the problem of checking whether an arbitrary compound term s ($s \in P(\mathcal{T})$) belongs to the compound terminology S_e of a given expression e. The straightforward way to achieve this is to first compute and store the compound terminology S_e and then to check whether $s \in S_e$. However, the number of computations needed for computing S_e, as well as the storage requirements, may be very large. An alternative which we choose is to develop an algorithm which can check whether $s \in S_e$ *without* having to compute S_e. Consequently, only the expression e must be stored.

Below we present the algorithm $IsValid(e, s)$ which takes as arguments a (well-formed) expression e and a compound term s, and returns TRUE if $s \in S_e$ and FALSE otherwise (i.e. if $s \notin S_e$). 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.

To present the algorithm we need some more notations. Let e be an expression over a facet set $\{F_1, ..., F_k\}$. The *facets* of e, denoted by $F(e)$, are defined as: $F(e) = \{F_i \mid F_i \text{ appears in } e\}$. Clearly, $F(e) \subseteq \{F_1, ..., F_k\}$. We shall denote by $F(t)$ the facet to which a term $t \in \mathcal{T}$ belongs, e.g. in Figure 1, we have $F(Crete) = Location$ and $F(SeaSki) = Sports$.

Algorithm 5.1 $IsValid(e, s)$
Input: An expression e and a compound term $s \subseteq \mathcal{T}$
Ouptut: TRUE if s belongs to S_e, or
FALSE, otherwise

```
if $s = \emptyset$ then return (TRUE)
If $\exists\, t \in s$ such that $F(t) \notin F(e)$, then return(FALSE)
if $s$ is singleton then return(TRUE)
case($e$) {
$\oplus_P(e_1, ..., e_n)$:
                    if $\exists\, p \in P$ such that $p \preceq s$ then return(TRUE)
                    For $i = 1, ..., n$
                        if $IsValid(e_i, s)$ then return(TRUE)
                    return(FALSE)
$\ominus_N(e_1, ..., e_n)$: if $\exists n \in N$ such that $s \preceq n$
                    then return(FALSE)
                    For $i = 1, ..., n$
                        Let $s_i = \{t \in s \mid F(t) \in F(e_i)\}$
                        if $IsValid(e_i, s_i)$=FALSE then return(FALSE)
                    return(TRUE)
$\overset{*}{\oplus}_P\ (T_i)$:       if $\exists p \in P$ such that $p \preceq s$ then return(TRUE)
                    if $s \in T_i$ then return(TRUE)
                    else return(FALSE)
$\overset{*}{\ominus}_N\ (T_i)$:       if $\exists n \in N$ such that $s \preceq n$ then return(FALSE)
                    else return(TRUE)
$T_i$:                If $s \in T_i$ then return(TRUE)
                    else return(FALSE)
}
```

The algorithm is based on the parse tree of the expression e. For example, consider the faceted taxonomy of Figure 7 and assume that the desired compound taxonomy is defined by the expression $e = (Location \ominus_N Accommodation) \oplus_P Facilities$, where

$$\begin{aligned} N = \quad & \{\{Heraklion, Bungalows\}\} \\ P = \quad & \{\{Hersonissos, Rooms, Indoor\}, \\ & \{Hersonissos, Bungalows, Outdoor\}, \\ & \{Ammoudara, Bungalows, Jacuzzi\}\} \end{aligned}$$

Figure 8 shows the parse tree of this expression.

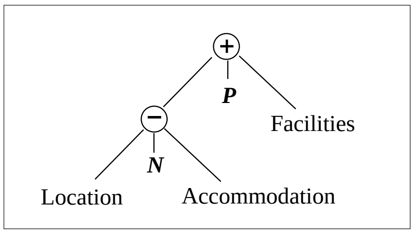

Figure 8: The parse tree of an expression

Then, it holds:
$IsValid(e, \{Hersonissos, Bungalows, SwimmingPool\})$=TRUE
$IsValid(e, \{Heraklion, Furn.Appartments\})$=TRUE

The trace of the execution of the call
$IsValid(e, \{Hersonissos, Bungalows, SwimmingPool\})$ is:

```
call $IsValid(e, \{Hersonissos, Bungalows, SwimmingPool\})$
/* $\exists p \in P$ s.t. $p \preceq \{Hersonissos, Bungalows, SwimmingPool\}$ */
  return(TRUE)
```

The trace of the execution of the call $IsValid(e, \{Heraklion, Furn.Appartments\})$ is:

```
call $IsValid((Location \ominus_N Accommodation) \oplus_P Facilities, \{Heraklion, Furn.Appartments\})$
/* $\nexists p \in P$ s.t. $p \preceq \{Heraklion, Furn.Appartments\}$ */
  call $IsValid((Location \ominus_N Accommodation), \{Heraklion, Furn.Appartments\})$
    call $IsValid(Location, \{Heraklion\})$
      return(TRUE)
    call $IsValid(Accommodation, \{Furn.Appartments\})$
      return(TRUE)
    return(TRUE)
  return(TRUE)
```

Additionally, we give the trace of the execution of the call $IsValid(e, \{Hersonnissos, Bungalows, Jacuzzi\})$.

```
call $IsValid((Location \ominus_N Accommodation) \oplus_P Facilities, \{Hersonnissos, Bungalows, Jacuzzi\})$
/* $\nexists p \in P$ s.t. $p \preceq \{Hersonissos, Bungalows, Jacuzzi\}$ */
  call $IsValid((Location \ominus_N Accommodation), \{Hersonnissos, Bungalows, Jacuzzi\})$
    return(FALSE)
  call $IsValid(Facilities, \{Hersonnissos, Bungalows, Jacuzzi\})$
    return(FALSE)
  return(FALSE)
```

6 Generating Navigation Trees

Let e be the expression that defines a desired compound taxonomy $(S_e, \preceq)$. In this section we describe a method for deriving a *navigation tree* for $(S_e, \preceq)$, that can be used during the following activities:

- *Indexing* the objects of the domain. This tree can speed up the indexing process and prevent indexing errors.
- *Browsing.* This tree can aid the user to reach the objects that satisfy a given information need.
- *Testing* whether the compound taxonomy contains only the desired set of compound terms.

A *navigation tree* is a directed acyclic graph (N, R) where N is the set of *nodes* and R is the set of *edges*. The nodes in N correspond to valid compound terms. Moreover, N contains nodes that enable the user to start browsing in one facet and then cross to another, and so on, until reaching the desired level of specificity.

Let us now introduce some notations. Given a term t, we denote by Brd(t) the set of all terms that are broader than t, i.e. Brd(t) $= \{t' \mid t \leq t'\}$. Given a compound term $s = \{t_1, ..., t_k\}$, let $Brd(s)$ be the set of all terms t such that t is broader than some term t_i, $i = 1, ..., k$, i.e. $Brd(s) = \text{Brd}(t_1) \cup ... \cup \text{Brd}(t_k)$. By $Brd(s)/\sim$ we denote the set of equivalence classes of the terms[7] in $Brd(s)$. For brevity hereafter we shall use $Brd(s)$ to denote $Brd(s)/\sim$.

[7]Equivalence of terms was defined at the beginning of Section 2.

The navigation tree (N, R) that we construct has the following property:

> for each compound term $s \in S_e$, the navigation tree has a path (starting from the root) for each *topological sort*[8] of the terms of the directed acyclic graph $(Brd(s), \leq)$.

For example consider the faceted taxonomy shown in Figure 1, and suppose that $\{Crete, SeaSports\} \in S$. The navigation tree in this case will include the following paths:

Location.Islands.Crete.Sports.SeaSports
Location.Islands.Sports.Crete.SeaSports
Location.Islands.Sports.SeaSports.Crete
Location.Sports.Islands.SeaSports.Crete
...
...
Sports.SeaSports.Location.Islands.Crete

As a further user aid whenever facet crossing occurs, a new node is created which presents the name of the facet (specifically its top term prefixed by the string "by") that we are crossing to. This facet crossing mechanism corresponds to the use of so-called "guide terms" for thesaurus expansion.

There are two approaches to deriving the navigation tree. The first approach is to generate a "complete" static navigation tree through an algorithm that takes e as input and returns a navigation tree. The second approach is to design a mechanism that generates the navigation tree dynamically during browsing.

Without loss of generality below we assume that each facet F_i has a greatest term with respect to subsumption which we denote by $top(F_i)$. Each node n of the navigation tree (N, R) is associated with a triple $(s(n), Fc(n), Nm(n))$ where:

- $s(n)$ is a *compound term*.
  As we shall see below, we construct navigation trees with nodes whose compound terms are valid.

- $Fc(n)$ is a so-called *focus term*.
  The focus term of a node n is a distinguished term among those that appear in $s(n)$ such that the children of n that are not used for facet-crossing are immediate children of $Fc(n)$. This means that from n, we either proceed to a different facet or we expand $Fc(n)$.

- $Nm(n)$ is a *name* for n.
  The name of a node is used for presenting the node at the user interface. It coincides with the focus term of n, unless n is a node for facet crossing. In the latter case the name of n is the name of the top term of the facet we are crossing to, prefixed by the string "by".

Below we describe an algorithm which takes as input the expression e that defines the compound taxonomy and returns a navigation tree (N, R). Roughly, the navigation tree is constructed as follows: At first we create a node for the top term of each facet that appears in e. Specifically, for each facet F_i we create a node whose compound term is the top term of F_i i.e. $top(F_i)$; we set as name and focus term of each such node the term $top(F_i)$. Now, for each node n we create *two* groups of children. The compound terms of the nodes in the first group are the results of replacing the focus term of n (i.e. $Fc(n)$) by an immediately narrower term of $Fc(n)$, while the second group consists of nodes for facet crossing.

[8] *Topological sort* of a set of terms is a sort that respects the partial order of the terms. That is, if $t, t' \in Brd(s)$ and $t \leq t'$, then t should always appear to the left of t' in the topological sort.

Instead of presenting the algorithm for constructing the entire navigation tree, in Algorithm $NavTreeInit(e)$, we present the initialization step, i.e. the creation of a top node for each facet appearing in e and, in Algorithm $CreateChildren(n)$, we present the steps for creating the children of a node n of the navigation tree. These steps can be synthesized (in a depth-first-search manner) to get an algorithm that constructs the entire navigation tree. The algorithms use the function `IsValid`(e, s) which returns `True` if s is a valid compound term according to e and `False` otherwise. The procedure `createNode` $(Nm(n), s(n), Fc(n))$ creates a node with the given parameters. The function $Nar(t)$ returns the immediately narrower terms of t. The procedure `AddChild` (n, n') makes n' child of n.

Algorithm 6.1 $NavTreeInit(e)$
Input: An algebraic expression e
Output: An initial top node for each facet in e

```
// Initialization: Creation of a top node for each facet
For each $F \in F(e)$
   createNode( top($F$), {top($F$)}, top($F$))
```

Algorithm 6.2 $CreateChildren(n)$
Input: A node n of the navigation tree
Output: The children of n

```
B.1   // Creating the children of a node on the basis of the focus term
      For each $t \in$ Nar(Fc(n))
         Let $s' := (s(n) - Fc(n)) \cup \{t\}$
         If IsValid($e, s'$) then
            $n'$:=createNode($t, s', t$)
            AddChild($n, n'$)

B.2   // Creating the children of a node for "facet crossing"
      For each $F_i \in F(e) - F(Fc(n))$
         Let $t_i := s(n) \cap \mathcal{T}_i$
         If $t_i = \emptyset$ then
            Let $s' := s(n) \cup \{top(F_i)\}$
            If IsValid($e, s'$) then
               $n'$:=createNode("by" + top($F_i$), $s'$, top($F_i$) )
               AddChild($n, n'$)
         else
            If $\exists t' \in$ Nar($t_i$) such that
            IsValid( $(s(n) - \{t_i\}) \cup \{t'\}$) then
               $n'$:=createNode("by"+top($F_i$), $s(n)$, $t_i$)
               AddChild($n, n'$)
```

Figure 9 shows a part of the navigation tree that is generated by this algorithm for the taxonomy shown in that Figure and expression $e = Sports \oplus_N Location$, where $N = \{\{WinterSports, Islands\}, \{SeaSports, Olympus\}\}$. In the navigation tree, each node n is presented by its name, $Nm(n)$. For example, the node n_{22} has $Nm(n_{22})$ = `Mainland`, $s(n_{22}) = \{\{$`Sports`, `Mainland`$\}\}$, $Fc(n_{22})$ = `Mainland`. The nodes n_{23} and n_{27} are generated by part B.1 of the algorithm, while node n_{30} is generated by part B.2.

7 Concluding Remarks

The novelty of our approach lies in enriching a faceted scheme with an algebra for specifying the valid compound terms. This method can be used in order to construct taxonomies or

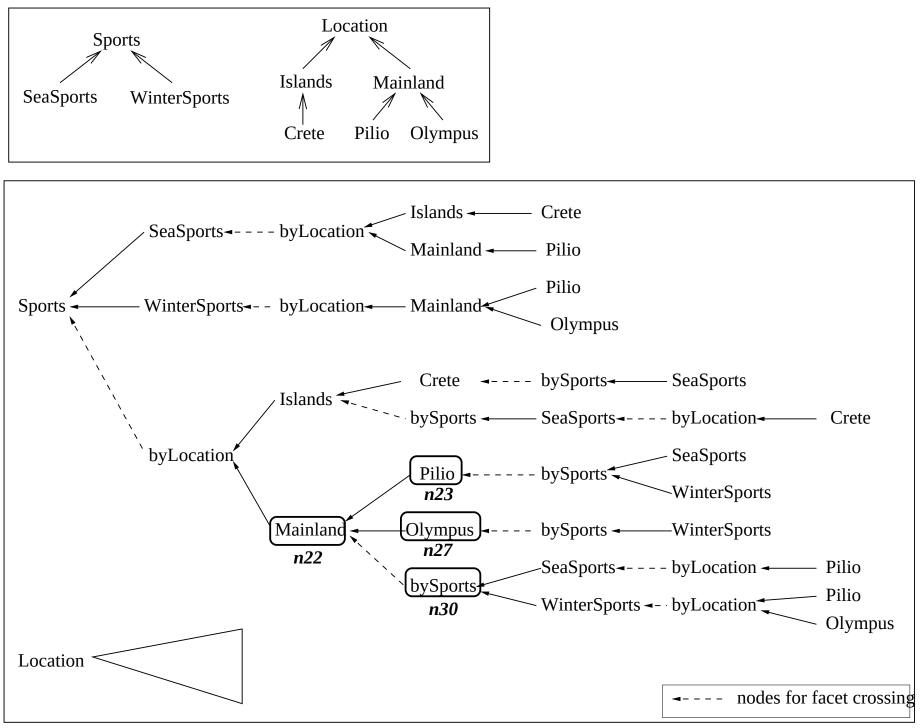

Figure 9: Example of a navigation tree

thesauri, which unlike existing thesauri, do not present the problem of missing terms or missing relationships (for more about this problem see [1]). We have not elaborated facet analysis, i.e. which facets should be selected and how they should be constructed. This process can be carried out either formally (see for example [4], [12], or [2]), or informally, as it is usually done by the designers of Web catalogs. Moreover, and in order to avoid misunderstanding we have to note here that our algebra is not related with the algebras that have been proposed for ontology engineering (e.g. [13, 3]). Our algebra is the only one that focuses on the problem of compound terms. It actually combines into a unified theory the extensions presented in [11]. There, the ideas of the plus and minus-product operations are called $PEFT$ and $NEFT$ respectively and they could be applied once on all facets. That is, $PEFT$ and $NEFT$ could not be synthesized. In the current work we presented an algebra which allows combining these two operations. In addition the algebra provides operators for capturing the cases of multiple classification within a facet, i.e. the self-product operations.

The advantages of our approach are the following:

- The algebra that we propose is quite *flexible* and quite *easy* to use. The designer does not have to write a program or to be familiar with logic-based languages. He just decides the order by which the facets appear in the expression and sets the parameters P and N which are just sets of compound terms. The simplicity of the compound terms considered (conjunctions of terms only) apart from allowing a very efficient inference mechanism, makes our approach easy to use and scalable. We believe that it can be adopted by catalog designers (librarians, etc) who are not familiar with logic-based representation languages.

- The operations are defined in a way that ensures that *no consistency problems* arise. This means that when the designer adds a new facet to the expression and defines the parameters P or N, he does not have to worry about inconsistencies.

- The compound terminologies defined by our algebra have *low storage space* requirements. There is no need to store the compound terminology of an expression. Only the expression has to be stored, as we provided an *efficient* inference mechanism which can check whether a compound term belongs to the compound terminology of the expression.

Our algebra can be used in any application that indexes objects using a controlled structured vocabulary, i.e. a taxonomy. For example it can be used for designing taxonomies for products, for fields of knowledge (e.g. for indexing the books of a library), etc. Moreover, we demonstrated how we can generate dynamically *navigation trees* which are suitable for browsing and can be also exploited during the indexing process (to aid the indexer and prevent indexing errors).

Currently, our algebra is been used for building the taxonomy of a tourist portal. The results that the designers report to us, concerning flexibility and ease of use, are so far very encouraging. An interesting application that we are going to investigate and implement in the near future, is to employ this algebra in order to design compound taxonomies for Web portals. Suppose that we want to create indexing terms that allow partitioning of 10^6 Web pages, in blocks of 10 pages. For doing this, we need at least 100 thousand (10^5) different terms, if we assume that each page is indexed by one term. If we want these terms to be the leaves of a complete balanced decimal tree, then this tree would have: $10^5 + 10^4 + ... + 10 + 1=$ 111,111 terms in total. By adopting a faceted taxonomy, we can obtain the same discrimination capability with much fewer terms. For example, with 5 facets each one having 10 leaves, the number of compound terms is greater that $10 \times 10 \times 10 \times 10 \times 10 = 10^5$. Assume that each facet is a complete balanced decimal tree, then the entire faceted taxonomy would have: $(10 + 1) \times 5 = 55$ terms in total. Notice the tremendous difference between 111,111 and 44. However, it is probably impossible to find 88 terms such that all of their combinations are valid.

The faceted taxonomy is expected to have many more terms and many combinations of these terms are expected to be invalid. However, our algebra offers a powerful means for specifying the valid compound terms. Returning back to our example, we believe that using our algebra we can obtain the desired discrimination capability with a relatively smaller number of terms and stored descriptions in P and N, by comparison to the 111,111 terms of a single hierarchical taxonomy.

Summarizing, instead of building huge hierarchical taxonomies we propose the employment of faceted taxonomies plus the usage of our algebra. In this way the designer can obtain taxonomies consisting of big numbers of valid indexing terms with less effort. Moreover the resulting compound taxonomies have low storage space requirements. Finally we have to note that the advantages of the compound faceted taxonomies that we propose (compactness, conceptual clarity, scalability, valid compound terms) can facilitate several other associated tasks. Specifically, they can certainly facilitate the design of *mediators* over several taxonomy-based sources (using the approach presented in [10]), and the *personalization* of Web catalogs (using the approach presented in [9]).

References

[1] Peter Clark, John Thompson, Heather Holmback, and Lisbeth Duncan. "Exploiting a Thesaurus-based Semantic Net for Knowledge-based Search". In *Procs of 12th Conf. on Innovative Applications of AI (AAAI/IAAI'00)*, pages 988–995, 2000.

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

[3] J. Jannink, S. Pichai, D. Verheijen, and G. Wiederhold. "Encapsulation and composition of ontologies". In *Proceedings of 1998 AAAI Workshop on AI & Information Integration*, 1998.

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

[5] Deborah L. McGuinness. “Ontological Issues for Knowledge-Enhanced Search”. In *Proceedings of FOIS’98*, Trento, Italy, June 1998. Amsterdam, IOS Press.

[6] Ruben Prieto-Diaz. “Implementing Faceted Classification for Software Reuse”. *Communications of the ACM*, page 88, 1991.

[7] 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.

[8] G. Salton. *“Introduction to Modern Information Retrieval”*. McGraw-Hill, 1983.

[9] Nicolas Spyratos, Yannis Tzitzikas, and Vassilis Christophides. “On Personalizing the Catalogs of Web Portals”. In *15th International FLAIRS Conference, FLAIRS’02*, pages 430–434, Pensacola, Florida, May 2002.

[10] Yannis Tzitzikas, Nicolas Spyratos, and Panos Constantopoulos. “Mediators over Ontology-based Information Sources”. In *Proceedings of the 2nd International Conference on Web Information Systems Engineering, WISE 2001*, pages 31–40, Kyoto, Japan, December 2001.

[11] Yannis Tzitzikas, Nicolas Spyratos, Panos Constantopoulos, and Anastasia Analyti. “Extended Faceted Taxonomies for Web Catalogs”. In *Proceedings of the 3rd International Conference on Web Information Systems Engineering, WISE 2002*, pages 192–204, Singapore, December 2002.

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

[13] Gio Wiederhold. “An Algebra for Ontology Composition”. In *Proceedings of 1994 Monterey Workshop on Formal Methods*, pages 56–61, September 1994.