# An Algebraic Approach 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@iei.pi.cnr.it, {analyti, panos}@ics.forth.gr, spyratos@lri.fr*

**Abstract.** One way of designing a taxonomy is by identifying a number of different aspects, or *facets* of the domain and then designing one taxonomy per facet. In such a faceted taxonomy, the indexing of objects is done by combining terms from different facets. A faceted taxonomy has several advantages by comparison to a single hierarchical taxonomy, such as conceptual clarity, compactness and scalability. However, a major drawback of faceted taxonomies is the possibility of forming a large number of invalid combinations of terms, i.e. combinations of terms that do not apply to any object of the underlying domain. The presence of such invalid combinations causes serious problems during object indexing or browsing. To alleviate this problem, we propose an algebra of taxonomies whose operators allow the efficient and flexible specification of only valid combinations of terms. This algebra can be used in order to construct very big compound taxonomies in a very systematic and efficient way.

## 1 Introduction

There are several application areas where a *taxonomy* is used for indexing the objects of a 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. 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 search engines have begun to exploit these catalogs in order to enhance the quality of retrieval and 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 searches. For example, using Google, one can first select a category, e.g. `Sciences/CS/DataStructures`, from the taxonomy of Open Directory and then submit

[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.

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 Directory. Clearly, this enhances the precision of the 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 home 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. Now, each object is described by 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 $\{Crete, SeaSki, Windsurfing\}$.

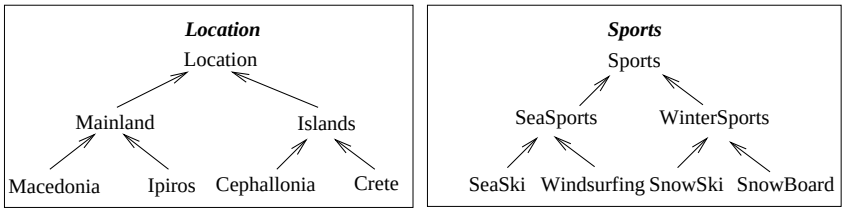


Figure 1: Two facets

The employment 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, the first using a single taxonomy consisting of 100 terms, and the second 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 several drawbacks. 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 compound term. That is, there may exist compound terms, such that there is no object of the underlying domain indexed by all of their terms. These compound terms are called *invalid*. 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. Indeed, there is no hotel in Crete offering snow-board facilities, as there is not enough snow in Crete! It follows that not

all compound terms of a faceted taxonomy are valid, even if all terms of every facet are valid. For example, it may hold that 9 from the 10 billions compound terms are invalid.

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. 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 the valid compound terms may give rise to problems in browsing. In a hierarchical taxonomy one browses until reaching the desired objects. In a faceted taxonomy, an invalid compound 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 of taxonomies 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. For formulating this expression the designer has to declare a small set of valid or invalid compound terms from which other (valid or invalid) compound terms are then inferred. In our example this means that the designer can specify the 1 billion valid (or the 9 billions invalid) compound terms by providing a relatively smaller number of (valid or invalid) compound terms. This is an important feature as it minimizes the effort needed by the designer. Another 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 rest 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 for compound taxonomies. Finally, Section 7 discusses applications and concludes the paper. All proofs are given in the extended version of this paper ([10]).

# 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 $(T, \leq)$, where $T$ is a *terminology* and $\leq$ is a *subsumption* relation over $T$, i.e. a reflexive and transitive relation over $T$.

If $a$ and $b$ are terms of $T$ and $a \leq b$ then we say that $a$ is *narrower* of $b$, or that $a$ is *subsumed* by $b$, or that $b$ *subsumes* $a$. 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 $T$ and that $\sim$ is an equivalence relation over the terms of $T$. 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, subsumption of terms is indicated by a continuous-line arrow from the subsumed term to the subsuming term. Note that we do not represent the entire subsumption relation but a subset of it sufficient to generate the entire relation. 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 connecting the terms that are equivalent. In what follows, we shall often write $T$ instead of $(T, \leq)$, whenever no ambiguity is possible.

We now introduce the concept of compound taxonomy, a basic concept for the remaining 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 $(T, \leq)$.

**Def 2.3** A *compound term* over $T$ is any subset of $T$.

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 $P(T)$ the set of all compound terms over $T$.

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

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

**Def 2.5** Let $s, s'$ be compound terms over $T$. The *compound ordering* over $T$ 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 $s'$. Roughly, $s \preceq s'$ means that $s$ carries more specific indexing information than $s'$. In addition, $s$ may contain terms not present in $s'$. 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 a narrower term of the unique term of $s_3$. On the other hand, $s_1 \not\preceq s_2$, as $s_1$ does not contain a narrower term of $WinterSports$. Finally, $s_2 \preceq s_3$. Note that $s \preceq \emptyset$, for every compound term $s$.

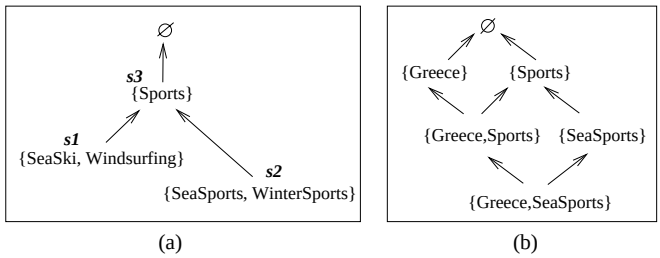


Figure 2: Two examples of compound taxonomies

Note that $\preceq$ is a subsumption relation over $S$, i.e. a reflexive and transitive relation over $S$. Additionally, note that the relation $\leq$ is provided explicitly by the designer when designing the taxonomy $T$, while the relation $\preceq$ is *derived* from $\leq$ according to the previous definition.

We say that two compound terms $s_1, s_2$ are *equivalent*, denoted by $s_1 \sim s_2$ iff $s_1 \preceq s_2$ and $s_2 \preceq s_1$ hold. In our theory, equivalent compound terms are considered the same.

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

Clearly, $(P(T), \preceq)$ is a compound taxonomy over $T$.

Now, as we have seen in the Introduction, 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 that we shall call *facets*. Based on a given set of facets we can define what we shall call *faceted taxonomy*.

**Def 2.7** Let $\{F_1, ..., F_k\}$ be a finite set of taxonomies, where $F_i = (T_i, \leq_i)$, and assume that the terminologies $T_1, ..., T_k$ are pairwise disjoint. Then the pair $\mathcal{F} = (\mathcal{T}, \leq)$ where $\mathcal{T} = \bigcup_{i=1}^{k} T_i$ 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 $\mathcal{F}$.

In the following, we shall use the top term of a facet to refer to the facet. For example, we refer to the facets of Figure 1 as $\{Location, Sports\}$. Clearly, all definitions that we have seen so far are also valid for a faceted taxonomy $(\mathcal{T}, \leq)$. Specifically, in much the same way as in a simple taxonomy, a faceted taxonomy is associated with a set of compound terms. For example, the following 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 our figures.

Let us now introduce some notations that will be used in the sequel. Let $s$ be a compound term. The broader and the narrower compound terms of $s$ are defined as follows: $\mathrm{Br}(s) = \{s' \in C(\mathcal{T}) \mid s \preceq s'\}$ and $\mathrm{Nr}(s) = \{s' \in C(\mathcal{T}) \mid s' \preceq s\}$. Now, let $S$ be a compound terminology over $\mathcal{T}$. The broader and the narrower compound terms of $S$ are defined as follows: $Br(S) = \cup\{\mathrm{Br}(s) \mid s \in S\}$ and $Nr(S) = \cup\{\mathrm{Nr}(s) \mid s \in S\}$.

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 Algebraic Operations

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, in this section we describe an algebra which allows defining a compound terminology over $\mathcal{T}$ (i.e. a subset of $P(\mathcal{T})$) which consists of the desired set of compound terms, i.e. of the compound terms that the designer considers as valid.

Hereafter we view each terminology $T_i$ as a compound terminology [6]:

**Def 3.1** The *basic compound terminology* of $T_i$ is the compound terminology defined as follows: $\{\ \{t\} \mid t \in T_i\} \cup \{\emptyset\}$

[6] By viewing each term as a singleton.

Hereafter we shall use $T_i$ to denote both the terminology and its basic compound terminology. All elements of the basic compound taxonomies $\{T_1, ..., T_k\}$ are considered valid and they constitute the "building blocks" of our algebra.

Let $\mathcal{S}$ denote the set of all compound terminologies over $\mathcal{T}$. Below we describe an algebra over $\mathcal{S}$, which is equipped with four operations, namely: *plus-product*, *minus-product*, *self-plus-product*, and *self-minus-product*. For defining the desired compound taxonomy the designer has to formulate a *well formed algebraic expression*[7] $e$, using these operations and operands the basic compound terminologies $T_1, .., T_k$. Below we describe each operation in detail.

## 3.1 The *Product* operation

We shall first define the auxiliary binary operation $\oplus$ over $\mathcal{S}$, i.e. $\oplus : \mathcal{S} \times \mathcal{S} \rightarrow \mathcal{S}$, 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 a compound terminology whose compound terms are all possible unions of compound terms from its arguments. 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 of the operation $S \oplus S'$ is shown in Figure 3. 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 $S \oplus S'$). The *indexing terms* of the compound terminology that is defined by the expression $S \oplus S'$, i.e. the compound terms that can be used during object indexing, are 8, as shown in Figure 3. It can be easily seen that the product just defined is commutative and associative and that multiple applications of $\oplus$ can be computed using the following: $S_1 \oplus ... \oplus S_n = \{\ s_1 \cup ... \cup s_n \mid s_i \in S_i\}$.

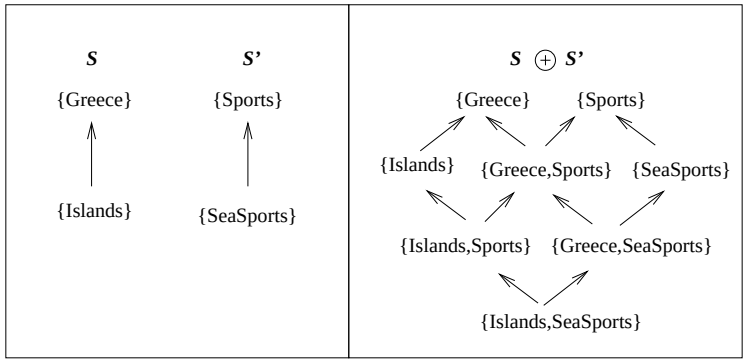


Figure 3: An example of a product $\oplus$ operation

[7]The definition of a well-formed algebraic expression is found in subsection 3.4.

## 3.2 The *plus-product* and the *minus-product* operation

Consider the compound terminologies $S$ and $S'$ shown in Figure 4, and suppose that we want to define a compound terminology which does not contain the compound terms $\{Islands, WinterSports\}$ and $\{Islands, SnowSki\}$. For this purpose below we shall 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 outcome, 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 outcome, i.e. they are invalid.

We first give the auxiliary definition of *genuine compound terms*.

**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:

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

Intuitively, a genuine compound term over a set of compound terminologies $S_1, ..., S_n$, combines non-empty compound terms from more that one $S_i$. For example if $S_1 = \{\{Greece\}, \{Islands\}\}$, $S_2 = \{\{Sports\}, \{WinterSports\}\}$ and, $S_3 = \{\{Pensions\}, \{Hotels\}\}$ then

$$\begin{aligned} \{Greece, WinterSports, Hotels\} &\in G_{S_1,S_2,S_3}, \\ \{WinterSports, Hotels\} &\in G_{S_1,S_2,S_3}, \text{ but} \\ \{Hotels\} &\notin G_{S_1,S_2,S_3} \end{aligned}$$

Now we shall define the *plus-product* operation, $\oplus_P$, i.e. an $n$-ary operation over $\mathcal{S}$ ($\oplus_P : \mathcal{S} \times ... \times \mathcal{S} \rightarrow \mathcal{S}$), where the parameter $P$ is a set of compound terms that we certainly want to appear in the outcome. The set $P$ is a subset of $G_{S_1,...,S_n}$ (i.e., $P \subseteq G_{S_1,...,S_n}$).

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

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

Intuitively, 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, if an object is indexed by a compound term $p$ then all compound terms in Br($p$) also apply to this object. 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 $Br(P)$.

The following proposition gives two simplifications of the operation for 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.

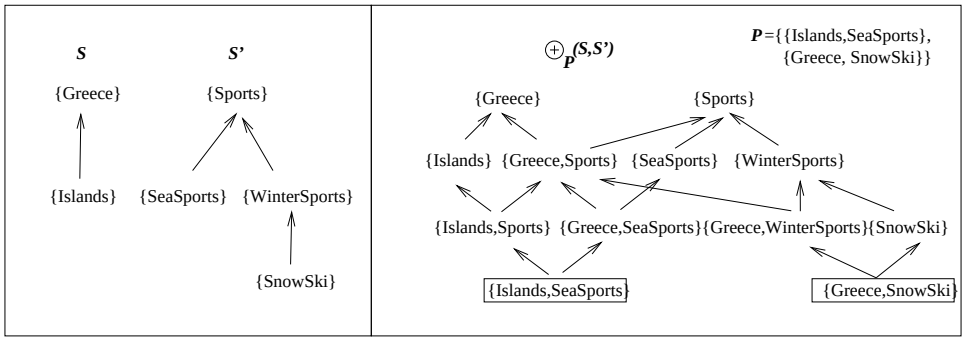


Figure 4: An example of a *plus-product*, $\oplus_P$, operation

**Prop. 3.1** Consider the compound terminologies $S_i$, for $i = 1, ..., n$. It holds:

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 shall define the *minus-product* operation, $\ominus_N$, i.e. 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 compound terms that we certainly do not want to appear in the outcome. The set $N$ is a subset of $G_{S_1,...,S_n}$ (i.e., $N \subseteq G_{S_1,...,S_n}$).

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

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

Intuitively, 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, if no object is indexed by a compound term $n$ then no object is indexed by any compound term in $\mathrm{Nr}(n)$. 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.

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** Let the compound terminologies $S_i$, for $i = 1, ..., n$. It holds:

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 $\ominus_N(S_1, ..., S_n) = S_1 \oplus ... \oplus S_n$.

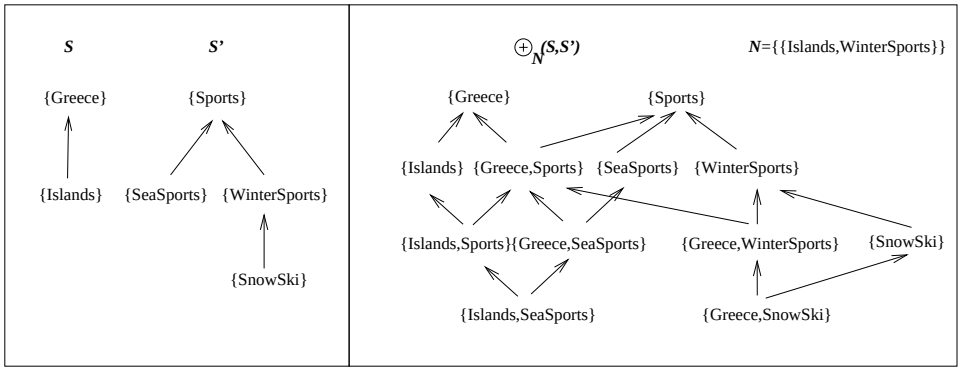


Figure 5: An example of a *minus-product*, $\ominus_N$, operation

## 3.3 The *Self-product* operations

The operators defined so far allow defining a compound terminology which consists of compound terms that contain at most one term from each facet. However, there may exist valid compound terms that contain more than one term from the same facet. For capturing such cases, below we shall define the *self-product*, $\overset{*}{\oplus}$, operation which gives all possible compound terms of one facet. Subsequently, we shall refine this operation with the parameters $P$ and $N$.

Let $\mathcal{BS}$ be the set of basic compound terminologies, that is $\mathcal{BS}= \{T_1, ..., T_k\}$. The following definition introduces the *self-product* operation, $\overset{*}{\oplus}$, i.e. a unary operation ($\overset{*}{\oplus}$: $\mathcal{S} \rightarrow \mathcal{S}$) which gives all possible compound terms of a facet.

**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 follows: $\overset{*}{\oplus}(T_i) = P(T_i)$ , where $P(T_i)$ is the powerser of the terminology $T_i$.

For example, consider the facet *Sports* of Figure 1. The compound terms $\{SeaSports, WinterSports\}$ and $\{SeaSki, Windsurfing, WinterSports\}$ are elements of $\overset{*}{\oplus}(Sports)$. Let now give an auxiliary definition.

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

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

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

Intuitively, 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 $\overset{*}{\oplus}_P(Sports)$,

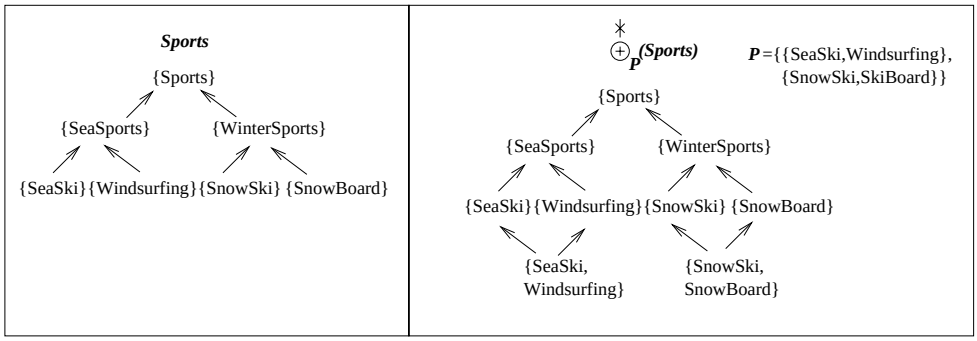


Figure 6: An example of a *self-plus-product*, $\overset{*}{\oplus}_P$, operation

where $P = \{\{SeaSki, Windsurfing\}, \{SnowSki, SkiBoard\}\}$ is shown in Figure 6. It is also easy to see that it holds: $\overset{*}{\oplus}_{G_{T_i}}(T_i) = \overset{*}{\oplus} T_i$, and $\overset{*}{\oplus}_{\emptyset} = T_i$.

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

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

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

Intuitively, 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 $N = \{\{SeaSports, WinterSports\}\}$. It is also easy to see that it holds: $\overset{*}{\ominus}_{G_{T_i}}(T_i) = T_i$, and $\overset{*}{\ominus}_{\emptyset}(T_i) = \overset{*}{\oplus} T_i$.

## 3.4 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** The set of expressions over a facet set $\{F_1, ..., F_k\}$ 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 $e$.

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

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

(ii) each parameter $P$ or $N$ that appears in $e$, is 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.

The constraints (i) and (ii) ensure that we have no conflicts. This implies that the compound terminology of an expression $e$ increases as the length of $e$ increases. For example, let $e_1$, $e_2$, and $e'$ be well-formed expressions such that $e_2 = e_1 \; op \; e'$. Certainly, it holds $S_{e_1} \subseteq S_{e_2}$, i.e. the compound terminology of $e_1$ is subset of the compound terminology of $e_2$. If we eliminate constraint (i) then the compound terminology of the expression $(T_1 \oplus_P T_2) \oplus_N T_1$ could be subset of the expression $T_1 \oplus_P T_2$. In addition, if we eliminate constraint (ii) then the compound terminology $T_1 \oplus_N T_2$ could be subset of $T_1$, or $T_2$.

In this paper, we consider only well-formed expressions.

# 4 Example

Consider that the domain of interest is a set of hotel home pages, and assume that we want to index these pages with respect to a taxonomy. For this purpose, the designer will first define the taxonomy to be used during indexing. Suppose that the designer decides to provide access to these pages 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 faceted taxonomy shown in Figure 7.

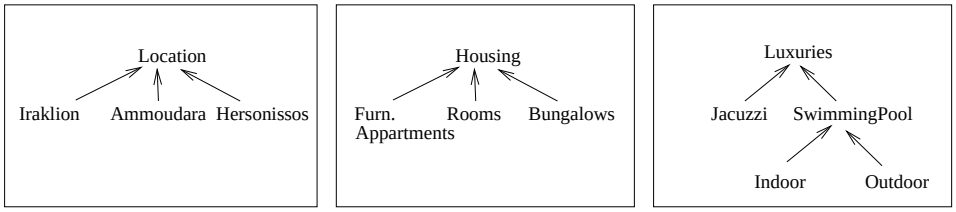


Figure 7: A faceted taxonomy consisting of three facets

This faceted taxonomy has 13 terms ( $|\mathcal{T}|$=13) and $P(\mathcal{T})$ has 890 compound terms[8]. However only a subset of these terms is valid. Specifically, suppose that the experience of the designer (domain expert) allow him/her to conclude 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:

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

[8]Recall that equivalent compound terms are considered the same. Thus, $P(\mathcal{T})$ is not $2^{13}$ but 890.

For specifying the 96 valid compound terms, the designer instead of enumerating them explicitly, he can use our algebra. For example, the designer can use a plus-product operation over facets $Location, Accommodation, Facilities$, that is

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

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

Note that the compound terms in $P$ are only 8. Additionally, the user could get the same result more efficiently through the expression

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

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

Note that the total number of compound terms in $P$ and $N$ is just 4. Summarizing, the faceted taxonomy of our example, has 13 terms, 890 compound terms, and 96 valid compound terms. Using an algebraic expression, the designer provides only 4 compound terms in order to specify the 96 valid compound terms.

# 5 Checking the Validity of a Compound Term

Let $S_e$ be the compound terminology of a well-formed expression $e$. Suppose that we want to check whether a compound term $s$ ($s \in P(\mathcal{T})$) belongs to $S_e$. One 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 $S_e$ may be very large. Moreover, the space requirements for storing $S_e$ may be very large too. For these reasons, here we present an algorithm which can check whether $s \in S_e$ without having to compute $S_e$. This means that only the expression $e$ has to be stored.

Before we present the algorithm, we give a few auxiliary definitions. Let $e$ be a well-formed expression over a facet set $\{F_1, ..., F_k\}$. The *facets* of $e$, denoted by $F(e)$, are defined as follows: $F(e) = \{F_i \mid F_i \text{ appears in } e\}$. Clearly, $F(e) \subseteq \{F_1, ..., F_k\}$. We shall write $F(t)$ to denote the facet to which a term $t \in \mathcal{T}$ belongs, e.g. in Figure 1, we have $F(Crete) = Location$ and $F(SeaSki) = Sports$.

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 the time complexity of $IsValid(e, s)$ is $O(|\mathcal{T}|^3 * |\mathcal{P} \cup \mathcal{N}|)$, where $\mathcal{P}$ denotes the union of all $P$ parameters appearing in $e$, and $\mathcal{N}$ denotes the union of all $N$ parameters appearing in $e$ (for more see [10]).

**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

(1) if $s = \emptyset$ then return (TRUE)
(2) If $\exists\ t \in s$ such that $F(t) \notin F(e)$, then return(FALSE)
(3) if $s$ is singleton then return(TRUE)
(4) case($e$) {
(5) $\oplus_P(e_1, ..., e_n)$: if $\exists\ p \in P$ such that $p \preceq s$ then return(TRUE)
(6) For $i = 1, ..., n$
(7) if $IsValid(e_i, s)$ then return(TRUE)
(8) return(FALSE)
(9) $\ominus_N(e_1, ..., e_n)$: if $\exists n \in N$ such that $s \preceq n$ then return(FALSE)
(10) For $i = 1, ..., n$
(11) Let $s_i = \{t \in s \mid F(t) \in F(e_i)\}$
(12) if $IsValid(e_i, s_i)$=FALSE then return(FALSE)
(13) return(TRUE)
(14) $\overset{*}{\oplus}_P\ (T_i)$: if $\exists p \in P$ such that $p \preceq s$ then return(TRUE)
(15) if $s \in T_i$ then return(TRUE)
(16) else return(FALSE)
(17) $\overset{*}{\ominus}_N\ (T_i)$: if $\exists n \in N$ such that $s \preceq n$ then return(FALSE)
(18) else return(TRUE)
(19) $T_i$: If $\exists s \in T_i$ then return(TRUE) else return(FALSE)
(20) }

# 6 Deriving Navigation Trees

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

- the indexer during *indexing* the objects of the domain. This tree can speed up the indexing process and prevent indexing errors.
- the user during *browsing*. This tree can aid the user to reach the objects that satisfy his/her information need.
- the designer for *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 $\mathrm{Brd}(t)$ the set of all terms that subsume $t$, i.e. $\mathrm{Brd}(t) = \{t' \mid t \leq t'\}$. Given a compound term $s = \{t_1, ..., t_k\}$ we denote by $Brd(s)$ the set of all terms that are broader than a term appearing in $s$, i.e. $Brd(\{t_1, ..., t_k\}) = \mathrm{Brd}(t_1) \cup ... \cup \mathrm{Brd}(t_k)$. By $Brd(s)/\sim$ we denote the set of equivalence classes of the terms[9] in $Brd(s)$. For brevity hereafter we shall use $Brd(s)$

[9]Equivalence of terms was defined in Section 2.

to denote $Brd(s)/\sim$.

We construct a navigation tree $(N, R)$ that 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*[10] of the nodes of the directed acyclic graph $(Brd(s), \preceq)$.

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

Moreover, and in order to further aid the user, whenever we have *facet crossing* 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.

There are two approaches to deriving the navigation tree. The first approach is to generate a "complete" static navigation tree, and for this purpose we need an algorithm that takes as input $e$ and returns a navigation tree [11]. The second approach is to design a mechanism that generates the navigation tree on the fly, i.e during browsing.
Without loss of generality below we assume that each facet $F_i$ has a greatest term from which all terms of the facet are "hung" and we will denote this term by $top(F_i)$. Specifically, each node $n$ of the navigation tree $(N, R)$ has:

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

- a *focus term*, denoted by $Fc(n)$.
  The focus term of a node $n$ is a distinguished term among those that appear in $s(n)$ (i.e., $Fc(n) \in s(n)$).

- a *name*, denoted by $Nm(n)$.
  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

[10] *Topological sort* of a set of terms is a sort that respects the partial order of the terms.
[11] In the domain of the Web the navigation tree would be a set of inter-linked Web pages.

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 $top(F_i)$ too. 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.

Instead of presenting the algorithm for constructing the entire navigation tree, in Algorithm 6.1 we present the first step, i.e. the creation of a node for each facet (Part A), and the steps for creating the children of a node (Part B). These steps can be synthesized to get an algorithm that constructs the entire navigation tree. The algorithm uses 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 immediate children of $t$. The procedure `addChild`$(n, n')$ makes $n'$ child of $n$.

Figure 8 shows a part of the navigation tree that is generated by this algorithm for the compound taxonomy defined by the expression $e = Sports \oplus_N Location$, where $N = \{\{WinterSports, Islands\}, \{SeaSports, Olympus\}\}$, assuming the faceted taxonomy shown in the same figure. In this figure, each node $n$ is presented by its name, $Nm(n)$. As an 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 the part B.1 of the algorithm, while the node $n_{30}$ is generated by the part B.2.

---

**Algorithm 6.1** Navigation Tree
*Input*: An expression $e$ over $\mathcal{F}$
*Output*: A navigation tree $(N, R)$

```
Part A  // Initialization: Creation of one node for each facet
        For each F ∈ F(e)
            createNode( top(F), {top(F)}, top(F))      // createNode(Nm(n), s(n), Fc(n))

Part B  // Creating the children of a node n
B.1     // Creating the children of a node on the basis of the focus term
        For each t ∈ Nar(Fc(n))
            Let s' := (s(n) − Fc(n)) ∪ {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 ∈ F(e) − F(Fc(n))
            Let t_i := s(n) ∩ T_i
            If t_i = ∅ then
                Let s' := s(n) ∪ {top(F_i)}
                If IsValid(e, s') then
                    n'=createNode("by" + top(F_i), s', top(F_i) )
                    AddChild(n, n')
            else
                If ∃t' ∈ Nar(t_i) such that IsValid( (s(n) − {t_i}) ∪ {t'}) then
                    n'=createNode("by"+top(F_i), s(n), t_i)
                    AddChild(n, n')
```

---

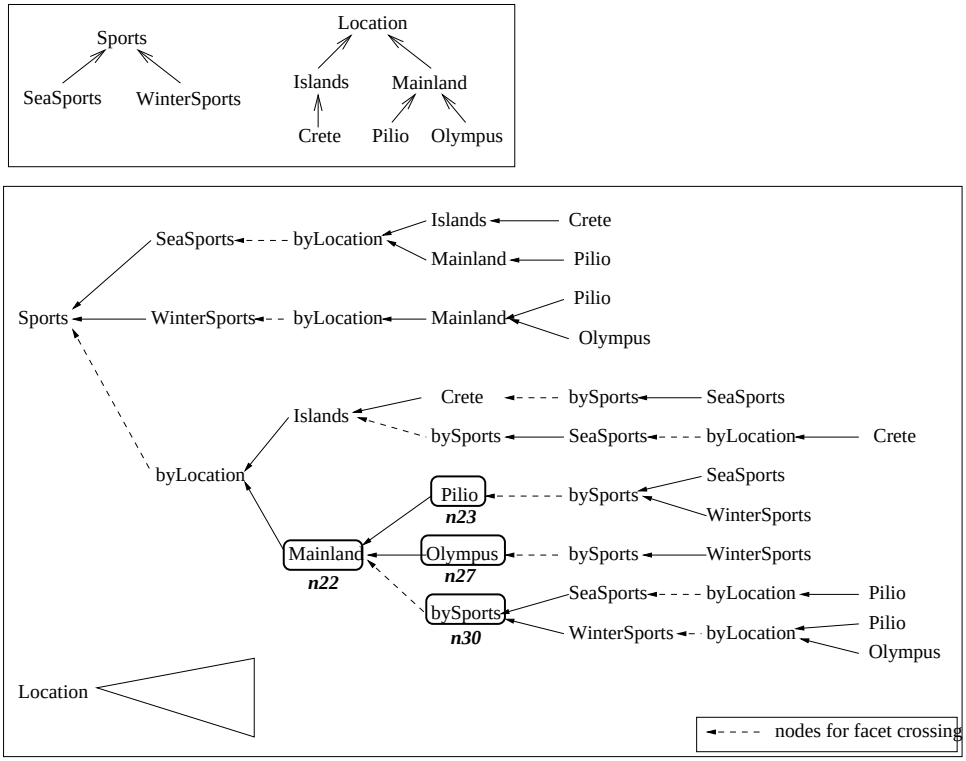


Figure 8: Example of a navigation tree

# 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 thesauri which, unlike existing thesauri, do not have 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], [13], 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. [14, 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 [12]. 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. Moreover, we have to note here that we cannot represent the compound taxonomies defined by our algebra in Description Logics (for more see [10]).

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. 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).

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 the Web. Currently, the Web consists of an estimated 1 billion ($10^9$) pages. Suppose that we want to create indexing terms that allow partitioning the pages of the Web in blocks of 10 pages. For doing this we need at least 100 millions ($10^8$) 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: $10^8 + 10^7 + ... + 10 + 1 = 111{,}111{,}111$ terms in total. By adopting a faceted taxonomy we can obtain the same discrimination capability with much fewer terms. For example, with 4 facets each one having 100 leaves, the number of all compound terms is greater than $100 \times 100 \times 100 \times 100 = 10^8$. If we want the 100 terms of each facet to be the leaves of a complete balanced decimal tree, then the entire faceted taxonomy would have: $(100 + 10 + 1) \times 4 = 444$ terms in total. We can obtain the same discrimination capability even with fewer terms! For example, we can have $10^8$ different combinations by adopting 8 facets each one having 10 leaves. In this case, the entire faceted taxonomy has only 88 terms! Notice the tremendous difference between 111,111,111 and 88. However, it is probably impossible to find 88 terms such that all of their combinations are meaningful for humans. Thus a faceted taxonomy for the entire Web 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 111,111,111 terms of a single 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 a large number 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 [11]), and the *personalization* of Web catalogs (using the approach presented in [9]).

# Acknowledgements

The first author wants to thank Tonia Dellaporta for being the source of his inspiration.

# 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)*, 2000.

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

[3] J. Jannink, S. Pichai, D. Verheijen, and G. Wiederhold. "Encapsulation and composition of ontologies". In *Proceedings of AAAI Workshop on AI & Information Integration, 1998*, 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*, 34(5), 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*, Pensacola, Florida, May 2002.

[10] Yannis Tzitzikas, Anastasia Analyti, Nicolas Spyratos, and Panos Constantopoulos. "An Algebra for Specifying Compound Terms for Faceted Taxonomies". Technical Report TR-314, Institute of Computer Science-FORTH, October 2002.

[11] Yannis Tzitzikas, Nicolas Spyratos, and Panos Constantopoulos. "Mediators over Ontology-based Information Sources". In *Second International Conference on Web Information Systems Engineering, WISE 2001*, Kyoto, Japan, December 2001.

[12] 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*, Singapore, December 2002.

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

[14] Gio Wiederhold. "An Algebra for Ontology Composition". In *Proceedings of 1994 Monterey Workshop on Formal Methods*, September 1994.