Extended Faceted Taxonomies for Web Catalogs

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

[1] Department of Computer Science, University of Crete, Greece
[2] Institute of Computer Science, ICS-FORTH
[3] Laboratoire de Recherche en Informatique, Universite de Paris-Sud, France
Email : tzitzik@ics.forth.gr, spyratos@lri.fr, panos@csi.forth.gr, analyti@csi.forth.gr

Abstract

Indexing and retrieval in Web catalogs can benefit from using faceted taxonomies. A faceted taxonomy consists of a set of facets, where each facet consists of a predefined set of terms structured by a subsumption relation. We propose two extensions of faceted taxonomies, which allow inferring conjunctions of terms that are valid in the underlying domain. We give a model-theoretic interpretation to these extended faceted taxonomies and we provide mechanisms for inferring the valid conjunctions of terms. This inference service can be exploited for preventing errors during the indexing process and for dynamically generating navigation trees that are suitable for browsing through the Web.

The proposed scheme has several advantages by comparison to the hierarchical classification schemes that are currently employed by Web catalogs, namely: (a) conceptual clarity*: it is easier to understand; (b)* compactness*: it takes less space; and (c)* scalability*: the update operations can be formulated easier and be performed more efficiently.*

1. Introduction

Web catalogs, such as Yahoo! or Open Directory[1], use structured vocabularies, known as *taxonomies*, for indexing the objects of a domain. 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. AltaVista, Google[2]), 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 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 [14], [10]).

Commonly, the taxonomies of these catalogs have a tree or a directed acyclic graph structure. The nodes correspond to terms (e.g. `Sciences`, `Mathematics`) and the edges correspond to subsumption relationships (e.g. `Mathematics` $\preceq$ `Sciences`). Such taxonomies may contain thousands of terms, e.g. the taxonomy of Yahoo! contains 20K terms while the taxonomy of OpenDirectory contains 300K terms! We can consider these taxonomies as *hierarchical classification schemes* like that of the Library of Congress or the Dewey Decimal System. For all their usefulness, these taxonomies have certain drawbacks with regard to comprehensibility, storage requirements and scalability. Hierarchical classification schemes require an a priori division of concepts into subconcepts, according to combinations of various criteria, such that every object in the domain can be assigned unambiguous, sufficiently detailed descriptive terms. Refining or revising such a scheme is a complex operation that also entails possibly extensive re-classification of the objects. By contrast, a *faceted classification scheme* (see [9] for a review) does not rely on the breakdown of a universe, but on building up, or *synthesizing*, from the subject statements of the objects. In a faceted scheme, subject statements are analyzed into their component elemental concepts, and these concepts are listed in the scheme. Their generic relationships are the only relationships displayed. To express a compound concept, one has to assemble its elemental concepts. This process is called *synthesis*, and

[1] http://dmoz.org
[2] www.google.com

the arranged groups of elemental concepts that make up the scheme are called *facets*. A faceted classification scheme consists of a finite set of *facets* where each facet consists of a *terminology*, i.e. a finite set of words, or *terms*, describing a domain from a particular aspect, and a binary relation over this terminology, namely *subsumption*. Within a faceted classification scheme, the objects of interest are indexed (classified) by associating each object with zero, one, or more terms from each facet. For example, to index a book we select from each facet the term that best describes the concepts in the title. Thus, a faceted scheme comprises independent sets of terms, corresponding to aspects, or facets, of the domain and the objects are assigned terms from the different facets. This process is more dynamic and it also limits the effects of a possible concept re-organization within the relevant facet. Faceted classification schemes seem to be superior to hierarchical classification schemes with regard to comprehensibility, storage requirements and scalability. They are also better suited for indexing collections that are subject to continuous expansion and change ([12]). Specifically:

- A faceted taxonomy is easier to understand. Since a facet is relatively smaller in size than an hierarchical taxonomy, the indexer or the user can browse each facet separately in order to understand the taxonomy. In contrast, understanding the taxonomy of a Web catalog such as Yahoo! through browsing is practically impossible due to its size.
- A faceted taxonomy is more compact, thus it requires less storage space. This is so because a faceted taxonomy does not contain terms for each compound concept.
- A faceted taxonomy is more scalable and can be maintained more easily. The additions/deletions of terms, as well as the structural changes, are easier and can be implemented more efficiently. The addition of a new term in a facet implies that many new combinations of terms are now available for indexing the objects of the domain. Moreover, to delete (or rename) a term, one only has to delete (or rename) that term from the facet in which it belongs. In contrast, in a hierarchical classification scheme one might have to delete (or rename) several terms in the tree structure.

A Web catalog can reap concrete benefits from adopting a faceted taxonomy. However, these will be discounted by impingements to the indexing and browsing processes. In a faceted taxonomy *invalid* conjunctions of terms coming from different facets may occur. A conjunction of terms is considered invalid if it cannot be applied to any of the objects of the domain. For example, in a tourist information application, the conjunction `Crete.WinterSports` would be invalid because there is not enough snow in Crete (here, we use a dot to denote the conjunction of the terms `Crete` and `WinterSports`). In contrast, the conjunction `Crete.SeaSports` is certainly valid. We can expect the frequency of this phenomenon to increase with the number of facets. Note that a hierarchical catalog for a tourist application would only contain the term `Crete.SeaSports`, while in a faceted taxonomy both conjunctions, `Crete.WinterSports` and `Crete.SeaSports`, would be possible during indexing or browsing.

Being able to infer the validity of a conjunction in a faceted taxonomy would be very useful. It can 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 conjunctions of 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 conjunction of terms will yield no objects. However if we could infer the valid conjunctions of terms in a faceted taxonomy then we would be able to generate navigation trees *on the fly*, which consist of nodes that correspond to valid conjunctions of terms.

In this paper we present two extensions of faceted taxonomies, which enable inferring valid or invalid conjunctions of terms through an inference mechanism. The designer simply declares a small set of valid or invalid conjunctions and other (valid or invalid) conjunctions are then inferred by the proposed mechanism. We demonstrate how to dynamically generate navigation trees for the extended faceted taxonomies, thus addressing the difficulties involved in browsing, or in describing objects by means of such taxonomies. The contributions of this paper are:

- a formalization of the extended faceted taxonomies,
- an efficient inference mechanism used for inferring description validity, and
- the dynamic generation of navigation trees for the extended faceted taxonomies.

The remaining of this paper is organized as follows: Section 2 presents some examples that illustrate the advantages of faceted taxonomies and the need for extending them to include inference capabilities. Section 3 defines faceted taxonomies and Section 4 introduces the extended faceted taxonomies. Section 5 describes the proposed inference mechanism and Section 6 describes a mechanism for generating navigation trees for the extended faceted taxonomies. Finally, Section 7 concludes the paper. All proofs are given in [22].

2. Examples of Faceted Taxonomies

As a first example, assume that the domain of interest is a set of hotel home pages, and suppose that we want to provide access to these pages according to the *location* of the hotels and the *sport facilities* they offer. Figure 1.(I) shows a hierarchical taxonomy such as those that are nowadays employed by Web Catalogs. Notice that it consists of nodes that correspond to valid conjunctions of terms. For instance, there is no node that corresponds to the conjunction of the terms `SeaSports` and `Olympus` (since Olympus is a mountain). Similarly, there is no node that corresponds to the conjunction `WinterSports.Crete` (since there is not enough snow in Crete). A faceted organization would look as in Figure 1.(II) and consist of two facets: one for *sports* and the other for *locations*. Let us now compare the organizations (I) and (II): (I) consists of 21 nodes and 20 edges, while (II) consists of 9 nodes and 7 edges. Now suppose that it is decided to delete the term `Mainland`. Then,

- for updating (I) we have to delete three terms and three relationships plus redirect five relationships, while
- for updating (II) we have to delete one term and one relationship plus redirect two relationships.

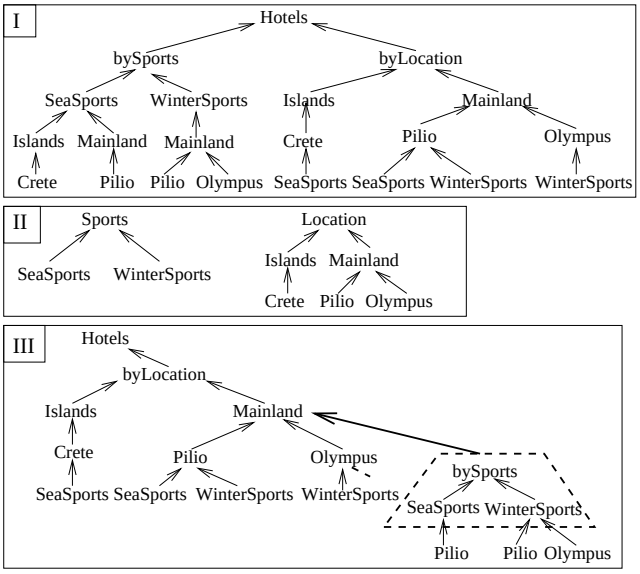

Figure 1. A hierarchical and a faceted taxonomy

Moreover, (I) cannot be considered as being "complete" for supporting browsing because it does not allow the user to *cross facets* while navigating from the root towards the leaves. For example, Figure 1.(III) shows that under the term `Mainland` we can create nodes for the terms of the facet Sports. A user who has followed the path `Hotels`>`ByLocation`>`Mainland` can now select the desired kind of sports, e.g. `SeaSports`, and subsequently select the desired place in the mainland. Note that here the nodes under the dotted triangle correspond to mainland locations only, so `Crete` does not appear under the node `SeaSports`.

Now consider the faceted taxonomy shown in Figure 2.(I) which has the additional facet `Housing`. Figure 2.(II) sketches the corresponding hierarchical organization. We can see that the number of nodes grows exponentially, and notice that this figure does not sketch all the nodes needed for supporting complete "facet crossing".

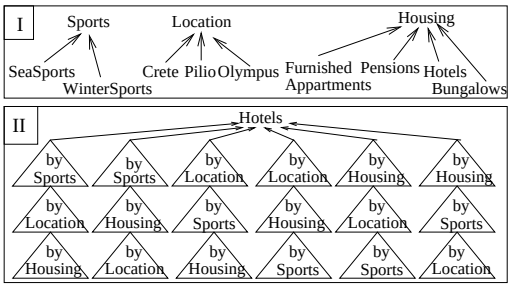

Figure 2. An taxonomy with three facets and a sketch of the corresponding hierarchical taxonomy

From the above examples it is evident why the taxonomies of existing Web catalogs contain so many terms, and are still not complete (in terms of nodes), especially at the deep levels of their hierarchies. For example, in OpenDirectory under the node
`Artificial Intelligence` > `Knowledge Representation` > `Ontologies` > `People`
there could be a node `byRegion` or a node `byProfessionalStatus`, e.t.c. The reason for this is that a "complete" navigation tree might require millions of nodes. In the extended faceted scheme that we will introduce shortly, we are able to derive complete navigation trees *dynamically* and these trees consist of valid nodes only.

3. Faceted Taxonomies

Roughly, a *faceted taxonomy* consists of a finite set of facets. Each *facet* consists of a *terminology*, i.e. a finite set of words or *terms*, structured by a *subsumption* relation. Each facet is designed separately, and models a distinct aspect of the domain. For instance, the faceted taxonomy for the domain of UNIX tools, presented in [12], consists of four facets, namely, "ByAction", "ByObject", "ByDataStructure" and "BySystem". Each facet consists of a set of terms. For example, the facet "ByDataStructure" consists of the following set of terms: { `buffer, tree, table, file, archive` }. Some other well known faceted taxonomies are listed in Table 1.

Below we define precisely what we call taxonomy and

Purpose:Source	*Facets*
Document Classification: Ranganathan [13]	Space, Time, Energy, Matter, Personality
Art and Architecture: AAT [5]	Associated Concepts, Physical Attributes, Styles and Periods, Agents, Activities, Materials, Objects
Science and Technology Classification:Vickery[24]	Place, Time, Attributes, Properties, Object, Parts, Processes, Substances, Recipient
Dissertation abstracts: Sugarman [18]	Operation, Object, Properties
Software classification: Prieto-Diaz [11]	Function, Object, Medium

Table 1. Examples of faceted schemes

what we call faceted taxonomy in this paper, and we introduce some notations.

Def 3.1 A *taxonomy* is a pair $(T, \preceq)$ where T is a *terminology*, i.e. a finite set of words, or *terms*, and $\preceq$ is a *subsumption* relation over T, i.e. a reflexive and transitive relation over T.

If a and b are terms of T, we say that a is *subsumed* by b if $a \preceq b$; we also say that b *subsumes* a; for example, `Databases` $\preceq$ `Informatics`, `Canaries` $\preceq$ `Birds`. We say that two terms a and b are *equivalent*, and write $a \sim b$, if both $a \preceq b$ and $b \preceq a$ 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 T, i.e. a reflexive, symmetric and transitive relation over T. Moreover $\preceq$ is a partial order over the equivalence classes of terms, i.e. a reflexive, transitive and anti-symmetric relation over the set of equivalence classes.

Def 3.2 A *faceted taxonomy* is a finite set $\mathcal{F} = \{F_1, ..., F_k\}$ of *taxonomies* in which each $F_i = (T_i, \preceq_i)$ is called a *facet*.

Without loss of generality we assume that every term of a facet in a faceted taxonomy $\mathcal{F}$ has a name that is unique within $\mathcal{F}$. If the same term appears in two different facets then we consider the two appearances as two different terms. This is denoted here by subscripting each term of a facet F_i by the subscript i, and can be implemented in practice by, say, prefixing terms by the names of the facets. We will use $\mathcal{T}$ to denote the union of the terminologies of all facets of $\mathcal{F}$, i.e. $\mathcal{T} = \bigcup_{i=1}^{k} T_i$, and $\preceq$ to denote the union of all subsumption relations $\preceq_i$, i.e. $\preceq = \bigcup_{i=1}^{k} \preceq_i$. Clearly the pair $(\mathcal{T}, \preceq)$ is a taxonomy (according to Def. 3.1). Note that a facet $F_i = (T_i, \preceq_i)$ can be unstructured, that is, $\preceq_i$ consists only of trivial relationships of the form $t \preceq_i t$. In such a case the facet is just a "flat" terminology. For instance, the faceted taxonomy in [12] consists of flat facets.

Facets are not arbitrary partitions of an underlying taxonomy. Rather, they are designed separately, effectively modeling different aspects of a common underlying domain. Consequently, the description of an object by means of a faceted taxonomy is achieved by associating the object with a conjunction of terms from different facets.

In this paper we are *not* interested in 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 [7], [24], [3], or [20]), or informally, as it is usually done by the designers of Web catalogs. Moreover in this paper, we are *not* interested in assigning descriptions to objects, but rather in inferring valid or invalid descriptions. Here by *description* we simply mean a conjunction of terms.

Def 3.3 A *description* d over $\mathcal{F}$ is either a term $t \in \mathcal{T}$ or a sequence of terms separated by "$\cdot$", i.e. any string derived by the following grammar $d ::= d \cdot t \mid t$

In Figure 2, for example, the following string is a description: `SeaSports.Crete.Pension`. The order of terms in a description does not make any difference. Hereafter we will use $D_{\mathcal{F}}$, or simply D, to denote the set of all descriptions over a faceted taxonomy $\mathcal{F}$. Note that D is an infinite set, even if T contains just a single term[3]. However, the set of equivalence classes of D is finite (equivalence of descriptions will be defined shortly).

Now, the problem with descriptions is that $\mathcal{F}$ does not itself specify which descriptions are valid and which are not. One approach to overcome this problem, would be to

- first enumerate all descriptions (specifically, all equivalence classes of descriptions),
- then partition this set into two disjoint subsets: one containing all valid descriptions and the other containing all invalid descriptions,
- and finally store one of the sets (either the set of valid, or the set of invalid descriptions).

For example, assume that the designer of the faceted taxonomy shown in Figure 1, considers descriptions which consist of exactly one term from each facet. In this case, the designer would define:

Valid Descriptions = {
Sports.Location, SeaSports.Location, WinterSports.Location
Sports.Islands, SeaSports.Islands, Sports.Mainland,
SeaSports.Mainland, WinterSports.Mainland, Sports.Crete,
SeaSports.Crete, Sports.Pilio, SeaSports.Pilio,
WinterSports.Pilio, Sports.Olympus, WinterSports.Olympus }
Invalid Descriptions = {
WinterSports.Islands, WinterSports.Crete, SeaSports.Olympus }

Defining these sets manually even for facets of relatively small size, would be a formidable task for the designer. Even in the special case where a description consists of exactly one term from each facet, there are $|T_1| * ... * |T_k|$ different combinations of terms, e.g. if $k = 5$ and $|T_i| = 10$ for each $i = 1..k$, then there are 100.000 different combinations. Moreover, this approach has prohibitive storage space requirements.

[3] Indeed $t,\ t.t,\ t.t.t$, and so on, are descriptions over T.

In what follows, we first give a semantic interpretation to $\mathcal{F}$ and then (in section 4) we define the extended faceted taxonomies and we exploit this semantic interpretation in order to infer the validity or invalidity of a description.

We conceptualize the world as a set of objects, that is, we assume an arbitrary domain of discourse and a corresponding set of objects Obj. A typical example of such a domain is the set of all pointers to Web pages. The only constraint that we impose on the set Obj is that it must be a denumerable set. The set of objects described by a term is the *interpretation* of that term, i.e. :

Def 3.4 Given a terminology T, we call *interpretation* of T over Obj any function $I : T \rightarrow 2^{Obj}$.

Here we use the symbol 2^{Obj} to denote the power set of Obj. Thus each term t denotes a set of objects in Obj and its interpretation $I(t)$ is the set of objects to which the term t is correctly applied. In our discussion the set Obj will usually be understood from the context. So, we shall often say "an interpretation" instead of "an interpretation over Obj". Interpretation, as defined above, assigns to a term a denotational or extensional meaning ([25]).

Def 3.5 Let $(T, \preceq)$ be a taxonomy. An interpretation I of T is a *model* of $(T, \preceq)$ if for all $t, t' \in T$, if $t \preceq t'$ then $I(t) \subseteq I(t')$.

Recall that if $\mathcal{F} = \{F_1, ..., F_k\}$ then $\mathcal{T} = T_1 \cup ... \cup T_k$ and $\preceq = \preceq_1 \cup ... \cup \preceq_k$. We shall call $(\mathcal{T}, \preceq)$ *the* taxonomy of $\mathcal{F}$.

Def 3.6 An interpretation I of $\mathcal{T}$ is a *model* of $\mathcal{F}$, if it is a model of the taxonomy $(\mathcal{T}, \preceq)$.

Note that if I is a model of $\mathcal{F}$ then the restriction of I on T_i, denoted $I_{|T_i}$, is a model of the facet F_i for each $i = 1, .., k$.

An interpretation I of $\mathcal{T}$ can be extended to an interpretation of D as follows: for any description $d = t_1 \cdot t_2 \cdot .. \cdot t_k$ in D we define $I(d) = I(t_1) \cap I(t_2) \cap ... \cap I(t_k)$.

Def 3.7 Let $\mathcal{F}$ be an taxonomy with terminology $\mathcal{T}$, let d, d' be two descriptions over $\mathcal{T}$, and let I be an interpretation of $\mathcal{T}$. We say that:

- d is *subsumed by* d' in I, and we write $I \models d \preceq d'$, if $I(d) \subseteq I(d')$,
- d *is subsumed by* d' in $\mathcal{F}$, and we write $\mathcal{F} \models d \preceq d'$, if $I \models d \preceq d'$ in every model I of $\mathcal{F}$,
- d and d' are *equivalent* in I, and we write $I \models d \sim d'$, if $I(d) = I(d')$,
- d and d' *are equivalent* in $\mathcal{F}$, and we write $\mathcal{F} \models d \sim d'$, if $I \models d \sim d'$ in every model I of $\mathcal{F}$.

For example, in the faceted taxonomy of Figure 1 we have: $\mathcal{F} \models$ `Sports.Location` $\sim$ `Location.Sports`, $\mathcal{F} \models$ `SeaSports.Crete` $\preceq$ `SeaSports.Islands`. It can be easily seen that $\sim$ is an equivalence relation over the set of descriptions D, which partitions the set D into a set of equivalence classes $D/\sim$.

Lemma 3.1 The set $D/\sim$ is finite.

Now, we define what we shall call valid and invalid description. A description d is *valid* in $\mathcal{F}$ if $I(d) \neq \emptyset$ in every model I of $\mathcal{F}$. A description d is *invalid* in $\mathcal{F}$ if $I(d) = \emptyset$ in every model I of $\mathcal{F}$. We shall use the symbols VD and ID to denote the set of all valid descriptions and all invalid descriptions, respectively, that is:

$$VD = \{d \in D \mid I(d) \neq \emptyset \text{ in every model } I \text{ of } \mathcal{F}\}$$
$$ID = \{d \in D \mid I(d) = \emptyset \text{ in every model } I \text{ of } \mathcal{F}\}$$

However both sets turn out to be empty, i.e. $VD = ID = \emptyset$, because for any description d we can construct a model I such that $I(d) \neq \emptyset$, and a model I' such that $I'(d) = \emptyset$. In the following section we propose two different extensions of a taxonomy that allows us to infer valid or invalid descriptions from other descriptions that have been declared as valid or invalid by the designer of the faceted taxonomy.

4. Establishing Description Validity by Taxonomy Extensions

In this section, given a faceted taxonomy $\mathcal{F}$, we introduce two extensions of $\mathcal{F}$.

In the first extension we consider $\mathcal{F}$ together with a given set of descriptions that the designer considers *valid*. Using this set our inference mechanism will infer *all* valid descriptions. In this extension, a description is considered invalid if it is not valid. Note that, in doing so, we adopt a closed world assumption.

In the second extension we consider $\mathcal{F}$ together with a given set of descriptions that the designer considers *invalid*. Using this set our inference mechanism will infer *all* invalid descriptions. In this extension, a description is considered valid if it is not invalid. Note that, in doing so, we again adopt a closed world assumption.

Def 4.1 A *Positive Extended Faceted Taxonomy*, or *PEFT* for short, is a pair $< \mathcal{F}, P >$ where $\mathcal{F}$ is a faceted taxonomy and P is a set of descriptions over $\mathcal{T}$. An interpretation I of $\mathcal{T}$ is a *model* of $< \mathcal{F}, P >$ if:
(a) I is a model of $\mathcal{F}$, and
(b) for each $d \in P$, $I(d) \neq \emptyset$.

Now, the set of valid descriptions VD of a *PEFT* $< \mathcal{F}, P >$ is defined as follows:
$VD = \{d \in D \mid I(d) \neq \emptyset \text{ in every model } I \text{ of } < \mathcal{F}, P >\}$

If a description is not an element of the set VD, then it is considered invalid, i.e. $ID = D \setminus VD$.

Lemma 4.1 If $d' \in P$ then for every description d such that $\mathcal{F} \models d' \preceq d$, $I(d) \neq \emptyset$ in every model I of $< \mathcal{F}, P >$.

If we write $< \mathcal{F}, P > \models d' \preceq d$ whenever $I(d') \subseteq I(d)$ in every model I of $< \mathcal{F}, P >$, then we can proceed to the following lemma:

Lemma 4.2

$$VD = P \cup \{d \in D \mid \exists\, d' \in P \text{ s.t. } <\mathcal{F}, P> \models d' \preceq d\}$$
$$= P \cup \{d \in D \mid \exists\, d' \in P \text{ s.t. } \mathcal{F} \models d' \preceq d\}$$

This implies that the designer does not have to specify all the valid descriptions. He only provides some of them and from these other valid descriptions can be inferred. For example, consider the faceted taxonomy of Figure 1.(II). If `Crete.SeaSports` $\in P$ then the description `Islands.SeaSports` and `Crete.Sports` are valid too. We can also easily see that if a description in P consists of m terms, then all descriptions that can be formed by using a subset of these terms are valid too, e.g. if `Crete.SeaSports.Bungalows` $\in P$ then the descriptions `Crete.SeaSports`, `Crete.Bungalows`, and `SeaSports.Bungalows`, are valid too.

We assume that all atomic descriptions, i.e. all descriptions that consist of only one term, are always valid, therefore $\mathcal{T}$ should be a subset of P. We consider this by default, thus the designer does not have to explicitly include the elements of $\mathcal{T}$ in P.

Let us now introduce the Negative Extended Faceted Taxonomies.

Def 4.2 A *Negative Extended Faceted Taxonomy*, or *NEFT* for short, is a pair $<\mathcal{F}, N>$ where $\mathcal{F}$ is a faceted taxonomy and N is a set of descriptions over $\mathcal{T}$. An interpretation I of $\mathcal{F}$ is a *model* of $<\mathcal{F}, N>$ if:

(a) I is a model of $\mathcal{F}$, and
(b) for each $d \in N$, $I(d) = \emptyset$.

Now, the set of invalid descriptions ID of a *NEFT* $<\mathcal{F}, N>$ is defined as follows:

$ID = \{d \in D \mid I(d) = \emptyset \text{ in every model } I \text{ of } <\mathcal{F}, N>\}$

If a description is not an element of the set ID, then it is considered valid, i.e. $VD = D \setminus ID$.

Lemma 4.3 If $d' \in N$ then for every description d such that $\mathcal{F} \models d \preceq d'$, $I(d) = \emptyset$ in every model I of $<\mathcal{F}, N>$.

If we write $<\mathcal{F}, N> \models d \preceq d'$ whenever $I(d) \subseteq I(d')$ in every model I of $<\mathcal{F}, N>$, then we can proceed to the following lemma:

Lemma 4.4

$$ID = N \cup \{d \in D \mid \exists\, d' \in N \text{ s.t. } <\mathcal{F}, N> \models d \preceq d'\}$$
$$= N \cup \{d \in D \mid \exists\, d' \in N \text{ s.t. } \mathcal{F} \models d \preceq d'\}$$

This implies that from the set N of invalid descriptions, other invalid descriptions can be inferred. For instance, if $t.t' \in N$ then every description that contains the term t, or a term subsumed by t, and the term t', or a term subsumed by t', is invalid too. For example, if `Crete.WinterSports` $\in N$ then this implies that the description `Crete.WinterSports.Hotel` is invalid too, as well as the description `Crete.SnowBoarding` (assuming that `SnowBoarding` $\preceq$ `WinterSports`).

4.1 Choosing between a *PEFT* and a *NEFT*

The designer can employ a *PEFT* or a *NEFT* extension depending on the faceted taxonomy of the application. If the majority of the descriptions are valid then it is better to employ a *NEFT*, so as to specify only the invalid descriptions. Conversely, if the majority of the descriptions are invalid, then it is better to employ a *PEFT* so as to specify only the valid descriptions.

Concerning the methodology for defining the set N, it is more efficient for the designer to put in N "short" descriptions that consist of "broad" terms. The reason is that from such descriptions a large number of new invalid descriptions can be inferred. For example in the hypothetical case that we want to specify that all descriptions over the faceted taxonomy of Figure 1 are invalid, it suffices to put in N one description, i.e. the description `Sports.Location`.

Concerning the methodology for defining the set P, it is more efficient for the designer to put in P "long" descriptions that consist of "narrow" terms, since from such descriptions a large number of new valid descriptions can be inferred. For example in the hypothetical case that we want to specify that all descriptions over the faceted taxonomy of Figure 1 are valid, it suffices to put in P just the following description: `SeaSports.WinterSports.Crete.Pilio.Olympus`.

Figure 3 shows how we can specify the valid/invalid descriptions of the faceted taxonomy of Figure 1 (i.e. the sets "Valid Descriptions" and "Invalid Descriptions" as presented in Section 3) by employing a *PEFT* or a *NEFT* extension.

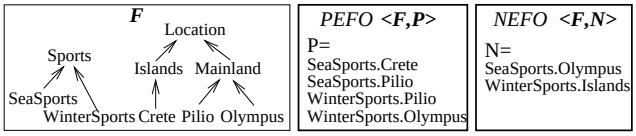

Figure 3. Example of a *PEFT* and a *NEFT* extension for the same domain

Also note that the previous theory can be applied even in the case where the faceted taxonomy consists of a single facet. However, more interesting results occur in the case where the numbers of facets is greater than one.

5. The Inference Mechanism for Deciding Description Validity

In Section 4 we introduced two taxonomy extensions and we gave a semantic interpretation to each of them. In this section we describe the inference mechanism needed in each case for deciding description validity.

Let $<\mathcal{F},P>$ be a *PEFT* extension. A description d is valid in $<\mathcal{F},P>$ if there is a description $p \in P$ such that $<\mathcal{F},P> \models p \preceq d$. Thus for checking whether d is valid one can check whether there is a $p \in P$ such that $<\mathcal{F},P> \models p \preceq d$. If there is such a p then certainly

the description d is valid, otherwise it is invalid. Thus for checking the validity of a description we need to perform $|P|$ inclusion checks. For checking inclusions we can exploit the inference mechanism described in [19]. In that mechanism, the complexity of an inclusion check $d \preceq d'$ is $O(|\preceq| * k)$ where k is the maximum number of terms that appear in the involved descriptions. Thus the validity of a description can be checked in $O(\mid P \mid * |\preceq| * k)$ time.

An alternative inference procedure is described below. Specifically we look for a special model, denoted by m, such that $m(d) \neq \emptyset$ iff $I(d) \neq \emptyset$ in every model I of $< \mathcal{F}, P >$. Below we provide an algorithm (Algorithm 1) which takes as input a pair $< \mathcal{F}, P >$ and returns the model m. In this algorithm, we assume a function *witness* that takes as argument a description d in P and returns a set containing a single object from Obj. We assume that *witness* is injective over the set of descriptions. The main idea is the following: In Step 1 we construct an interpretation m so that every description in P has non empty interpretation. Then, in Steps 2 and 3, we "enlarge" this interpretation to a model of $\mathcal{F}$. Clearly, the resulting interpretation is a model of $< \mathcal{F}, P >$.

Algorithm 1
Input: A *PEFT* $< \mathcal{F}, P >$
Ouptut: The model m of $< \mathcal{F}, P >$

Step 1: For each $d = t_1 \cdot ... \cdot t_m \in P$
  If $m(d) = \emptyset$ then
    $m(t_1) := m(t_1) \cup witness(d)$
    ...
    $m(t_m) := m(t_m) \cup witness(d)$
  EndIf
Step 2: For each $t \preceq t'$
  If $m(t) \not\subseteq m(t')$ then
    $m(t') := m(t) \cup m(t')$
Step 3: If changes in Step 2 then
  Goto Step 2
 else return m

The table that follows shows the model constructed by this algorithm for the *PEFT* extension shown in Figure 3. As mentioned earlier, we assume that P by default *includes* the set of atomic terms $\mathcal{T}$. This means that the set P in this example consists of the 9 terms of $\mathcal{T}$ plus the 4 descriptions of P that are shown in Figure 3. The objects which are returned by the function *witness* are represented by natural numbers.

$t \in \mathcal{T}$	$m(t)$ after Step 1	$m(t)$ returned
Sports	{1}	{1,2,3,10,11,12,13 }
SeaSports	{2,10,11}	{2,10,11 }
WinterSports	{3,12,13}	{3,12,13 }
Location	{4}	{4,5,6,7,8,9,10,11,12,13 }
Islands	{5}	{5,7,10 }
Mainland	{6}	{6,8,9,11,12,13 }
Crete	{7,10}	{7,10 }
Pilio	{8,11,12}	{8,11,12 }
Olympus	{9,13}	{9,13}

Prop. 5.1 Algorithm 1 produces a model of $< \mathcal{F}, P >$.

Prop. 5.2 $m(d) \neq \emptyset$ iff $I(d) \neq \emptyset$ in every model I of $< \mathcal{F}, P >$.

Thus for checking the validity of a description we run Algorithm 1 once, which produces the model m and then it suffices to check the validity of descriptions in this model. Thus for checking the validity of a description $d = t_1 \cdot ... \cdot t_k$ we have to compute $m(t_1 \cdot ... \cdot t_k)$, i.e. we have to perform k intersections of sets. As the Algorithm 1 creates at most $|T| + |P|$ objects, an intersection can be perfomed in $O(|T| + |P|)$ time. Thus $m(d)$ can be computed in $O(k*(|T|+|P|))$ time. Clearly, this method is more efficient than performing $|P|$ inclusion checks (which has execution time in $O(\mid P \mid * |\preceq| * k)$).

Now, let $<\mathcal{F},N>$ be a *NEFT* extension. A description d is *invalid* in $<\mathcal{F},N>$ if there is an $n \in N$ such that $<\mathcal{F},N> \models d \preceq n$. If there is such n then certainly the description d is invalid, otherwise it is valid. Thus for checking whether d is invalid one can check whether there is an $n \in N$ such that $< \mathcal{F}, N > \models d \preceq n$, thus we need to perform $|N|$ inclusion checks. By using the inference mechanism described in [19], we can check the validity of a description in $O(\mid N \mid * |\preceq| * k)$ time. For the *NEFT* extensions we have not yet managed to find an inference procedure that is based on a single special model m. This is an issue for further research.

The reasoning services needed for inferring the validity of a description in a *PEFT*, can alternatively be obtained by representing the $PEFT$ by an appropriate set of Horn Rules and exploiting the inference mechanism of Prolog. On the other hand, the reasoning services needed for inferring the validity of a description in a *NEFT*, can also be obtained by representing the *NEFT* by an appropriate free Tbox of Description Logics [2], and exploiting the corresponding inference mechanisms (for more see [22]).

Note that the simplicity of the descriptions considered here (conjunctions of terms only) not only allows a very efficient inference mechanism, but also makes our approach easy to use and scalable. We believe that this approach can be adopted by catalog designers who are not familiar with logic-based representation languages. Of course, there are limitations in terms of expressive power, which is not necessarily bad: consider relational databases that are doing pretty well in the world of large applications.

6. Generating Navigation Trees

Faceted taxonomies cannot be browsed easily especially when they consist of many facets. For browsing a faceted taxonomy one would have to select one or more terms from each facet (and some facets may contain many terms) in order to formulate a description that reflects one's information

need. However many of the resulting descriptions might be meaningless, i.e. invalid, thus yielding no objects. Therefore we would like a *navigation tree*, that is, a tree with nodes that correspond to only valid descriptions. The tree should have 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. It is important to note that the same navigation tree can be used by the indexer during the indexing of objects in the domain in order to speed up the indexing process and prevent indexing errors.

Let us first introduce some notations before presenting the algorithm for generating the navigation tree. Given a term t, we denote by $Br(t)$ the set of all terms that subsume t, i.e. $Br(t) = \{t' \mid t \preceq t'\}$. Given a description $d = t_1 \cdot ... \cdot t_k$ we denote by $Br(d)$ the union of the broader terms of all terms that appear in d, i.e. $Br(t_1 \cdot ... \cdot t_k) = Br(t_1) \cup ... \cup Br(t_k)$. By $Br(d)/\sim$ we denote the set of equivalence classes of the set $Br(d)$. For brevity hereafter we shall use $Br(d)$ to denote $Br(d)/\sim$.

A *navigation tree* is a directed acyclic graph (N, R) where N is the set of *nodes* and R is the set of *edges*. Let $\mathcal{G}$ be an extended faceted taxonomy (either *PEFT* or *NEFT*). The navigation tree that we construct for $\mathcal{G}$ has the following property: for each valid description $d \in VD$, the navigation tree has a path (starting from the root) for each *topological sort* of the nodes of the directed acyclic graph $(Br(d), \preceq_{|Br(d)})$. For example consider the faceted taxonomy shown in Figure 3, and suppose that $\texttt{Crete.SeaSports} \in VD$. The navigation tree in this case will include the following paths:

Location > Greece > Crete > Sports > SeaSports
Location > Greece > Sports > Crete > SeaSports
Location > Greece > Sports > SeaSports > Crete
Location > Sports > Greece > SeaSports > Crete
...
Sports > SeaSports > Location > Greece > 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 $\mathcal{G}$ and returns a navigation tree[4]. 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 "hanged" and we will denote this term by $top(F_i)$. Specifically, each node n of the navigation tree (N, R) has:

- a *description*, denoted by $D(n)$.
  As we shall see below, we construct navigation trees with nodes whose descriptions 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 the description $D(n)$ of the node.
- 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 an extended faceted taxonomy $\mathcal{G}$ and returns a navigation tree (N, R). Roughly, the navigation tree is constructed as follows: At first we create a node for the greatest element of each facet. Specifically for each facet F_i we create a node whose description is the greatest element 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 descriptions 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 2 we present the first step, i.e. the creation of a node for each facet, and the steps for creating the children of a node. These steps can be synthesized to get an algorithm that constructs the entire navigation tree. An algorithm that constructs the entire navigation tree in a depth-first-search manner is given in [22]. The algorithm uses a function $\texttt{IsValid}(< description >)$ which returns `True` if its argument is a valid description and `False` otherwise. The command $\texttt{addChild}(< name >, < descr >, < focusterm >)$ creates a new child of the current node. In the presentation of the algorithm we adopt the notation described next.

We denote each facet F_i of a faceted taxonomy $\mathcal{F} = \{F_1, ..., F_k\}$ by the integer i. For example for the faceted taxonomy $\mathcal{F} = \{\texttt{Sports}, \texttt{Location}\}$ shown in Figure 3, the facet `Sports` is denoted by the integer 1 and the facet `Location` is denoted by the integer 2. Now, by $f(t)$ we denote the facet in which the term t belongs, e.g. $f(\texttt{SeaSports}) = 1$ and $f(\texttt{Crete}) = 2$. By $\Pi_i(d)$ we denote the narrowest subterm of d that belongs to facet i, or the empty string if there is no such subterm, e.g. $\Pi_1(\texttt{SeaSports.Crete}) = \texttt{SeaSports}$, $\Pi_2(\texttt{SeaSports.Crete}) = \texttt{Crete}$, $\Pi_1(\texttt{Crete}) = ""$. By $top(i)$ we denote the greatest term of facet i. By $Nar(t)$ we denote the set of immediately narrower terms of t, e.g. $Nar(\texttt{Location}) = \{\texttt{Islands}, \texttt{Mainland}\}$.

Figure 4 shows a part of the navigation tree that is generated by this algorithm for the taxonomy of Figure 3. In

[4] In the domain of the Web the navigation tree would be a set of interlinked Web pages.

this figure, each node n is presented by its name, $Nm(n)$. As an example, the node n_{22} has $Nm(n_{22}) =$ `Mainland`, $D(n_{22}) =$ `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 2 Navigation Tree
Input: An extended faceted taxonomy $\mathcal{G}$
Output: A navigation tree (N, R)

Part A // Initialization: Creation of one node for each facet
For each $i = 1..k$
`addChild`(top(i), top(i), top(i))

Part B // Creating the children of a node n
B.1 // Creating the children of a node on the basis of the focus term
NarF := Nar(Fc(n))
For each $t \in$ NarF
If `IsValid`(D(n).t) then
`addChild`(t, D(n).t, t)

B.2 // Creating the children of a node for "facet crossing"
For each $i = 1..k$ and $i \neq$ f(Fc(n))
If Π_i(D(n)) = "" then // $D(n)$ does not contain any term in F_i
If `IsValid`(D(n).top(i)) then
`addChild`("by" + top(i), D(n).top(i),top(i))
Else
If $\exists t' \in$ Nar(Π_i(D(n)) such that `IsValid`(D(n).t') then
`addChild`("by" + top(i), D(n), Π_i(D(n)))

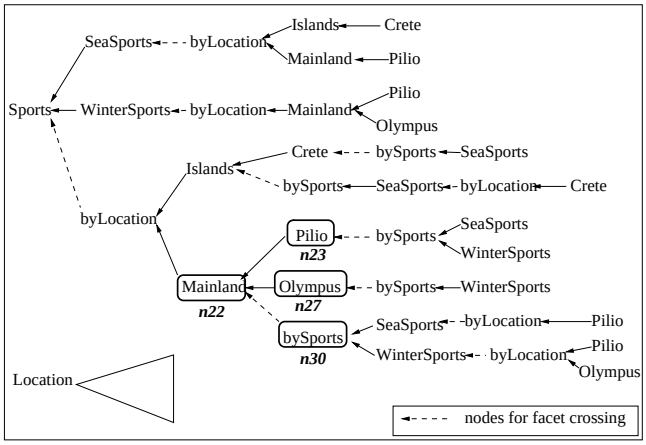

Figure 4. Example of a navigation tree

7. Concluding Remarks

We presented a novel approach for indexing and retrieving objects based on multiple aspects, or facets. Although faceted classification was suggested quite long ago (by Ranganathan in the 1920s [13]) the associated issues have not received adequate attention by the computer science community. However, there are several works about facet analysis (e.g. see [3], [24],[7]). Facets have also been studied in library and information science (for a review see [9]). For instance, thesauri ([6]) may have facets that group the terms of the thesaurus in classes. The contribution of this work lies in enriching a faceted scheme with a rigorous method for specifying valid combinations of terms. This method can be used in order to construct taxonomies or thesauri which unlike existing thesauri, do not present the problems of missing terms or missing relationships (for more about this problem see [1]). Moreover, we demonstrated how to generate navigation trees dynamically which are suitable for browsing and can also be useful in preventing errors during the indexing process.

Specifically, we provided two methods for establishing description validity. In the first, the designer provides a set of valid descriptions P, while in the second the designer provides a set of invalid descriptions N. The designer does not have to specify all the valid descriptions in the first case, nor all the invalid in the second. By virtue of the semantic interpretation of these extended faceted schemes, new valid descriptions (in the first case) and new invalid descriptions (in the second) can be inferred. This reduces the effort needed in order to establish description validity.

An interesting application that we intend to investigate and implement in the near future, is to employ these schemes in order to design enhanced taxonomies for the Web. Currently the Web consists of an estimated 1 billion (10^9) pages. Suppose we want to create terms that allow partitioning the pages of the Web in blocks of 10 pages. For doing so we need at least 100 millions (10^8) different terms, assuming 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 terms, the number of all combinations of terms equals $100 \times 100 \times 100 \times 100 = 10^8$. 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 with even fewer terms! For example, we can have 10^8 different combinations by adopting 8 facets, each one having 10 terms. 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 to be meaningful for humans. Thus a faceted taxonomy for the entire Web is expected to have many more terms and many combinations of those terms are expected to be invalid. However, *PEFT* and *NEFT* offer a flexible and powerful method for specifying the valid combinations. Returning back to our example, we believe that using a *PEFT* (or a *NEFT*) we should be able to obtain the desired discrimination capability with a relatively small number of terms and stored descriptions in P (or N), by comparison to the 111,111,111 terms of a hierarchical

taxonomy.

Recently, ontologies which consist of a taxonomy, attributes, relations, and axioms, are being employed for the Semantic Web (e.g. see [8], [4], [23]). However, we have to note that in a very broad domain such as the set of all Web pages indexed by a Web catalog like Yahoo!, it is not so easy to identify the classes of the domain because the domain is too wide and different users with different needs conceptualize it differently, i.e. one class of a conceptual model according to one user may correspond to a value of an attribute of a class of a conceptual model according to another user. Thus if the purpose of the application is the indexing and retrieval of objects (and not answering structured queries) then our scheme seems to be appropriate. Also recall that in the domain of the Web, the queries submitted by ordinary users are mostly bags of words (and not structured queries).

The advantages of the extended faceted taxonomies that we propose (compactness, conceptual clarity, scalability, description validity) can facilitate several other associated tasks. Specifically, the adoption of extended faceted taxonomies can certainly facilitate the design of *mediators* over several Web catalogs (using the approach presented in [21]), and the *personalization* of Web catalogs (using the approach presented in [17]). The model-theoretic interpretation that we gave to the extended faceted taxonomies is an extension of the work presented in [19], which was inspired by the approach of [15] and [16]. An issue for further research is to study descriptions which are ordered sets of terms and to investigate the possibility of having both "positive" and "negative" descriptions.

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

References

[1] P. Clark, J. Thompson, H. Holmback, and L. 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] F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. *"Reasoning in Description Logics"*, chapter 1. CSLI Publications, 1997.

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

[4] J. Heflin, J. Hendler, and S. Luke. "Coping with Changing Ontologies in a Distributed Environment". In *Proceedings of AAAI-99 Workshop on Ontology Management*, 1999.

[5] G. R. Institute. "Art & Architecture Thesaurus". (http://www.getty.edu/research/tools/vocabulary/aat/).

[6] International Organization For Standardization. "Documentation - Guidelines for the establishment and development of monolingual thesauri", 1986. Ref. No ISO 2788-1986.

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

[8] S. Luke, L. Spector, D. Rager, and J. Hendler. "Ontology-based Web Agents". In *Proceedings of First International Conference on Autonomous Agents*, 1997. (http://www.cs.umd.edu/projects/plus/SHOE/).

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

[10] D. L. McGuinness. "Ontological Issues for Knowledge-Enhanced Search". In *Proceedings of FOIS'98*, Trento, Italy, June 1998. Amsterdam, IOS Press.

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

[12] R. Prieto-Diaz. "Implementing Faceted Classification for Software Reuse". *Communications of the ACM*, 34(5), 1991.

[13] S. R. Ranganathan. "The Colon Classification". In S. 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.

[14] G. Salton. *"Introduction to Modern Information Retrieval"*. McGraw-Hill, 1983.

[15] N. Spyratos. "The Partition Model: A Deductive Database Model". *ACM Transactions on Database Systems*, 12(1):1–37, 1987.

[16] N. Spyratos and C. Lecluse. "Incorporating Functional Dependencies in Deductive Query Answering". In *Proceedings of the Third International Conference on Data Engineering, ICDE-87*, February 1987.

[17] N. Spyratos, Y. Tzitzikas, and V. Christophides. "On Personalizing the Catalogs of Web Portals". In *15th International FLAIRS Conference, FLAIRS'02*, Pensacola, Florida, May 2002.

[18] J. H. Sugarman. *"The development of a classification system for information storage and retrieval purposes based upon a model of scientific knowledge generation"*. PhD thesis, School of Education, Boston University, 1981.

[19] Y. Tzitzikas, N. Spyratos, and P. Constantopoulos. "Deriving Valid Expressions from Ontology Definitions". In *11th European-Japanese Conference on Information Modelling and Knowledge Bases*, Maribor, Slovenia, May 2001.

[20] Y. Tzitzikas, N. Spyratos, and P. Constantopoulos. "Faceted Ontologies for Web Portals". Technical Report TR-293, Institute of Computer Science-FORTH, October 2001.

[21] Y. Tzitzikas, N. Spyratos, and P. Constantopoulos. "Mediators over Ontology-based Information Sources". In *Second International Conference on Web Information Systems Engineering, WISE 2001*, Kyoto, Japan, December 2001.

[22] Y. T. Tzitzikas. *"Collaborative Ontology-based Information Indexing and Retrieval"*. PhD thesis, Department of Computer Science - University of Crete, September 2002.

[23] F. van Harmelen and D. Fensel. "Practical Knowledge Representation for the Web". In *Workshop on Intelligent Information Integration, IJCAI'99*, 1999.

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

[25] W. A. Woods. "Understanding Subsumption and Taxonomy". In *Principles of Semantic Networks*, chapter 1. Morgan Kaufmann Publishers, 1991.