

Modeling and Manipulating the Structure of Portal Catalogs

Theodore Dalamagas, Alexandra Meliou, Timos Sellis

Knowledge and Database Systems Lab
School of Electrical and Computer Engineering
National Technical University of Athens
Zographou 157 73, Athens, Hellas
{dalamag,ameliou,timos}@dmlab.ece.ntua.gr

Abstract

Portal catalog sites are Web information nodes that provide querying and browsing capabilities on data organized in a thematic hierarchy. Up to now, portal catalogs have not been treated as full-fledged objects so as to allow for their manipulation. A major challenge is thus to complement catalog search and browsing with operations that manipulate structural information from similar vertical portal catalog sites, that is sites of a specific subject or domain.

This work proposes models and operators that allow for structural manipulation of vertical portal catalog sites, considering them as first-class citizens in the information search space. First, we explore the algebraic properties of trees representing hierarchies of portal catalogs, and define a lattice algebraic structure on them. Then, turning this structure into a boolean algebra, we present the operators S -union, S -intersection and S -difference to support structural manipulation of such trees. These operators have certain algebraic properties to provide clear semantics and assist the transformation, simplification and optimization of sequences of operations, using laws similar to those of set theory. Finally, we identify the conditions under which this framework is applicable.

1. Introduction

The Web information space is a set of information nodes. Each information node maintains a data source, a data schema, a query engine and a query interface. The data source serves as the storage place of data. The data schema describes the content of the data source using concept/role or entity/relationship or semistructured models. The query engine implements the query mechanism, and, finally, the query interface provides query capabilities for users. Portal catalogs are Web information nodes that provide querying and browsing capabilities on data organized in a hierarchy, on a category/subcategory basis.

New factors have changed the requirements for an efficient information retrieval in the Web information space. There are many *vertical* portal catalogs, that is catalogs of a specific subject or domain (e.g. e-marketplaces for hardware, photo equipment, cultural resources, etc). These sites provide better querying capabilities compared to the traditional internet search engines: keyword search, category browsing, category search and sometimes SQL-like querying. Moreover, there is a huge amount of information which can be retrieved in the form of dynamically-generated Web pages due to the high usage of database systems for storing data. Such information is often stored in many data sources with similar (but not identical) schemas. Taking the above into consideration, effective information retrieval is the process which supports the identification and usage of Web information nodes that provide querying capabilities for data relevant to an information need.

Portal catalog sites are quite useful in such retrieval tasks, since they maintain large volumes of information resources which is not possible for a single user to classify and exploit. However, up to now,

portal catalogs have not been treated as full-fledged objects so as to allow for their manipulation. Moreover, the XML infrastructure has become a popular means for enabling automatic processing of Web information nodes. Given that XML data are organized in a hierarchically structured and self-describing manner, there is a need that the structural aspect should receive stronger attention. There is a major challenge to complement catalog search and browsing with operations that manipulate structural information from *similar vertical* portal catalogs, that is a set of portal catalogs providing querying and browsing capabilities on a specific subject or domain. Such a set of portal catalogs can be a powerful tool to assist the retrieval task in the Web information space. The next section clarifies such a motivation.

1.1. Motivating example

Adorama, B&H and RitzCamera¹ are three e-marketplaces for photo equipment. The first two maintain a complex hierarchy with categories/subcategories to assist searching and browsing for their products, while the last one provides a simple one-level hierarchy to categorize products and only browsing capabilities. Figure 1 shows parts of their hierarchy. Notice that a schema matching pre-processing (see also related work in Section 1.3) helps the identification of all matching categories (nodes) from both catalogs, since there may be categories with different labels, although they are semantically similar. Where matching is not straightforward due to different naming, we provide the necessary information giving the matching categories in Figures 1, 2 and 3.

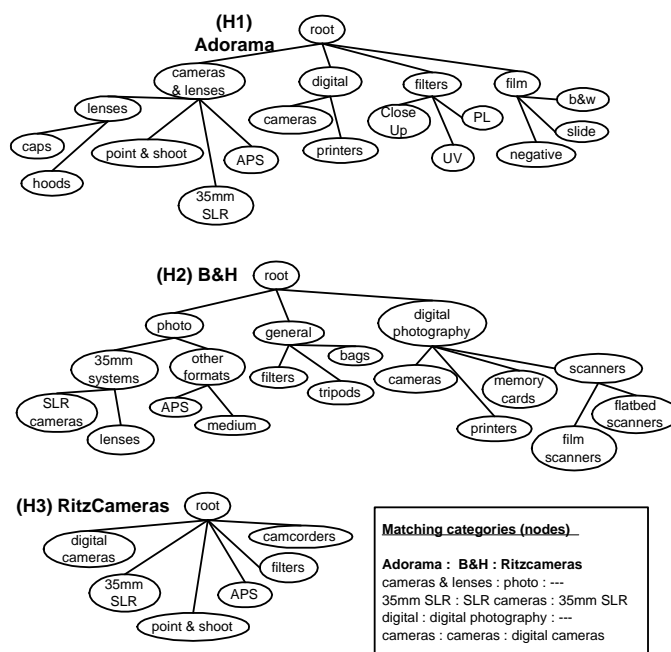


Figure 1. Part of Adorama’s, B&H’s and RitzCamera’s hierarchy.

One can think of various query operations on data provided by those catalogs, for example browse the hierarchies to find 35mm SLR cameras in all catalogs or pose path expression queries on Adorama’s hierarchy like `/Filters/UV/"price<40"`, that is find ultra-violet filters with price less than 40euros. However, looking at the three e-marketplaces as *a set of similar vertical portal catalogs* maintaining resources relevant to photo equipment, there is a need to complement such kind of querying with operations that manipulate structural information from the hierarchies of these catalogs, treating them as first class citizens. Such structural information acts as a semantic guide for query operations on data categorized in such catalogs.

Under this perspective, one can think of various query operations on structural information provided by the hierarchies of portal catalogs:

¹www.adorama.com, www.bhphotovideo.com, www.ritzcamera.com

1. (Q_1) Find the integrated structural information provided by Adorama's and B&H' catalogs: Such a query has a 'union' flavor and its answer should include structural information present either in Adorama or in B&H or in both, merging Adorama with B&H.
2. (Q_2) Find the common structural information provided by Adorama's and B&H's catalogs: Such a query has an 'intersection' flavor and its answer should include structural information present in both Adorama and B&H.
3. (Q_3) Find the part of Adorama's catalog which is not present in B&H's catalog: Such a query has a 'difference' flavor and its answer should include structural information present in Adorama but not in B&H.
4. (Q_4) Find the integrated structural information provided by the common part of Adorama's and B&H catalogs, and RitzCamera's catalog: Such a query is a composition of 'intersection' and 'union' sub-queries.

To support such queries and clarify their semantics, we should identify related operators. Looking back at the first three example queries Q_1, Q_2, Q_3 , we identify a set of three operators with union-like, intersection-like and difference-like properties, the output of which gives the answers to those queries:

- Figure 2 presents the answer to query Q_1 (union-like): a merged catalog from Adorama's and B&H catalogs. In this new catalog, there are all categories from Adorama's and B&H's catalogs. Its structural relationships may not be present in all the original trees. For example, the new catalog has caps and hoods for lenses, a categorization which is applicable only in Adorama's catalog but not in B&H's catalog. On the other hand, APS is common in both catalogs. In Figure 3, the catalog produced in Figure 2 is merged with RitzCameras' catalog.
- Figure 4(a) presents the answer to query Q_2 (intersection-like): the common part of Adorama's and B&H catalogs. In this new catalog, there are categories common in Adorama's and B&H's catalogs. Its structural relationships are present in both catalogs. For example, the new catalog has APS and lenses for photo, a categorization which is applicable in both Adorama's (just below cameras & lenses, the matching category of photo) and B&H's catalog (following the path from photo down to the leaves).
- Finally, Figure 4(b) presents the part of Adorama's catalog not present in B&H's catalog, which is the answer to query Q_3 (difference-like). Its structural relationships are present only in Adorama's catalog. For example, the new catalog has negative, slide and b&w for film, a categorization found only in Adorama's catalog.

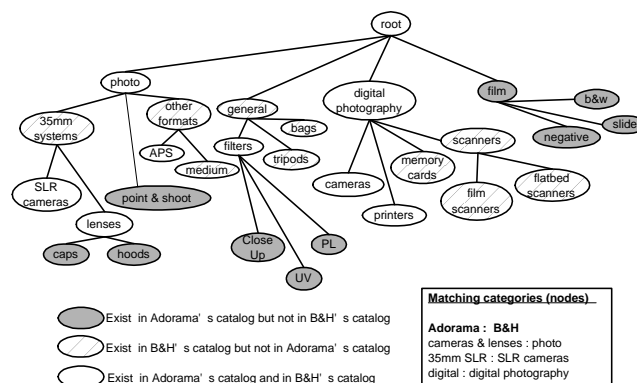


Figure 2. Merging Adorama's and B&H's catalogs.

A composition of those operators can give results like in Figure 4(c) which gives the answer to query Q_4 : the catalog produced by merging RitzCamera's catalog with the common part of both Adorama's and B&H catalogs.

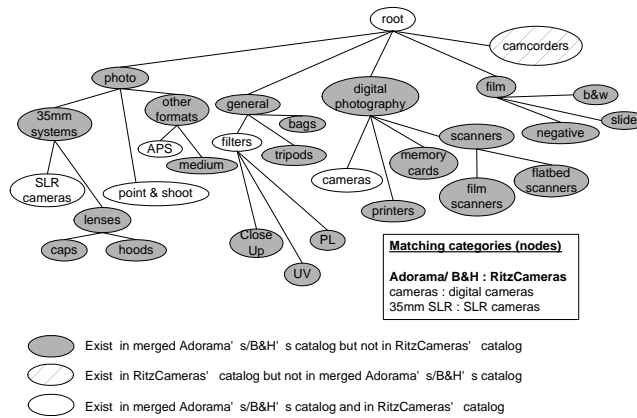


Figure 3. Merging Adorama/B&H's and RitzCameras' catalogs.

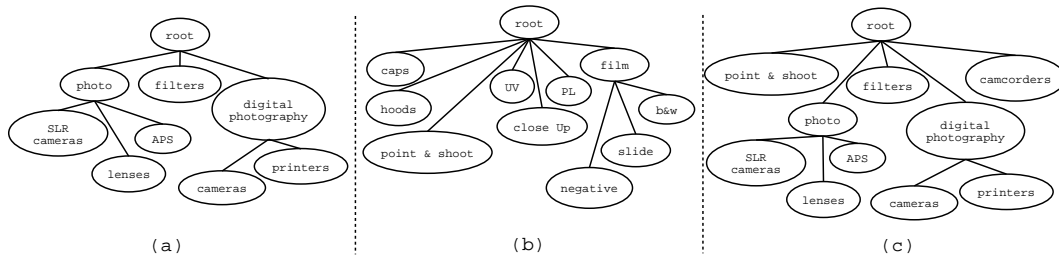


Figure 4. (a) The common part of Adorama's and B&H's catalogs. (b) The part of Adorama's catalog which is not present in B&H's catalog. (c) Merging RitzCamera's catalog with the common part of both Adorama's and B&H catalogs.

We classify such kind of query requirements as part of the *generic model management* framework, presented in [4, 5, 24]. According to this framework, models are manipulated as abstractions rather than sets of individual elements, using model-at-a-time and mapping-at-a-time operators. In our case, we aim to manipulate hierarchical tree-like schemas with low-level, set-like query operators applied on trees, emphasizing on the ability to have certain algebraic properties to provide clear semantics and assist the transformation, simplification and optimization of sequences of operations, which is not explored in [4, 5, 24]. There are several examples where certain properties are required:

1. When applying a set of operators on a set of portal catalogs, one should ensure that the result is the same regardless their sequence. For example, merging Adorama's with B&H's catalogs, and then merging the result with RitzCamera's catalog should produce the same catalog with the one after merging B&H's catalog with RitzCamera's catalog, and then merging the result with Adorama's catalog (see Figure 3).
2. Even better, since the operators have union-like, intersection-like and difference-like flavor, one could transform, simplify and optimize sequences of operations, given that laws similar to those of set theory hold. For example, one could infer that merging Adorama's catalog with the common part of B&H's catalog and a third catalog, same as the one in Adorama, will result to Adorama's catalog itself, without performing any operation at all, given that a kind of an absorption law holds. Or, knowing the merged catalog of RitzCamera and Adorama, and the merged catalog of RitzCamera and B&H's from previous operations, one could answer query Q_4 by just finding their common part, given that a kind of a distributive law holds.
3. Finally, we should examine the conditions under which these operators are applicable in the presence of structural inconsistencies. Having for example the catalog H_1 in Figure 1 and a similar

catalog but with camera to be the parent of digital, we cannot get a merged catalog unless we decide what will be the new relationship between these two nodes whose current relationship is in conflict.

1.2. Contribution

The main contribution of this work is the proposal of low-level, set-like query operators applied on portal catalogs, having certain algebraic properties to provide clear semantics and assist the transformation, simplification and optimization of sequences of operations. Specifically:

1. We present a tree-like model for hierarchies of portal catalog sites.
2. We explore the algebraic properties of trees representing hierarchies of portal catalogs, and define a lattice algebraic structure on them, based initially on the assumption that trees have certain properties with respect to a given tree called global tree (an assumption that is relaxed later).
3. Turning this structure into a boolean algebra, we present a set of operators (S -union, S -intersection, S -difference) to manipulate the structure of portal catalogs, treating them as first class citizens in the information search space.
4. We give the laws that hold for these operators so that one can transform, simplify and optimize sequences of operators. The laws are similar to those in set theory.
5. Finally, we provide a framework to ensure that similar algebraic properties and laws hold, in the absence of a global tree. We show that even if such a tree is not given, we can construct one based on the available trees, under certain conditions.

1.3. Related work

Related issues have been exploited in research areas like schema merging and integration, ontology construction, semistructured data management and complex object management.

Schema merging and information integration is a major subject in the area of Database Systems. Much work has been done on building an integrated schema from heterogeneous sources using a common data model, especially using some variant of the ER model [17, 13, 20, 9, 29, 21]. Theoretical aspects of schema merging have been discussed in [6, 25]. Finally, finding similarities between objects of different schemas and dealing with semantic conflicts to support schema matching is crucial for schema merging and information integration (see [27] for a detailed survey). Most recent research focuses on ontologies [31, 14, 30], where methods for ontology composition and merging are presented, and XML schemas trees [23, 3, 19, 7], where methods to unify and integrate trees representing XML data are discussed. Finally, general schema definition and structural querying capabilities are provided from RDF/XML schema languages, like for example RQL [15, 18]. However, the presented related work is either focused on the detection and resolving of semantic conflicts to support the schema integration task in the presence of strict typed and semantically rich schemas, or puts emphasis only on structural transformation and querying of schemas. Our work, on the other hand, provides an integrated framework for structural manipulation on hierarchical schemas with lightweight semantics, in the form of set-like query operators on trees considered as first-class citizens.

In this sense, prior research on complex object management is closely related to our work. Complex objects propose nested relations against normalized ones in the attempt to overcome the restrictions imposed by flat relations. A calculus for complex objects is presented in [2]. A lattice structure is defined on nested objects and operators on these objects are introduced. However, the operators are used to provide an extension of Horn clauses as a calculus to define formally a path-expression query language on structures. Neither complements are suggested nor a difference operator is defined. Operation and implementation issues for complex objects are presented in [16]. Again the target is to modify and query the configuration of a complex object. A join operator is introduced, similar to the relational

join, but neither union, nor intersection, nor difference operators are used. In [26], an extension of the relational algebra is suggested to capture complex objects. Other works, like [28], model complex objects with object-oriented schemas, and introduce class union using attribute merging, class intersection using attribute matching and class difference using missing attributes. Generally, the main focus of the related work in complex objects is on how to select and reconstruct substructures of the original structures, as one can see in [1], and not on supporting structural manipulation on the basis of either merging structures or finding their common parts or their different parts.

Finally, as we have already noticed previously, operators for generic model management are suggested in [4, 5, 24]. Our emphasis is to provide operators with clear semantics and certain algebraic properties to assist the transformation, simplification and optimization of sequences of operations.

1.4. Outline

Section 2 deals with modeling issues for representing hierarchies of portal catalogs. Section 3 explores the algebraic properties of trees representing portal catalog hierarchies and introduces the S -union and S -intersection operators. Section 4 defines a lattice algebraic structure on these trees, and turns this structure into a boolean algebra, introducing the S -difference operator and a set of laws hold. Section 5 discusses further the presented framework and explores the conditions under which is applicable. Section 6 re-examines the motivation example of Section 1, and, finally, Section 7 concludes this paper.

2. Representing hierarchies for portal catalogs

Portal catalogs provide data organized in hierarchies, on a category/subcategory basis. Such kind of hierarchies can be seen as lightweight ontologies, with their concepts (categories) related with primitive semantics *isA* and *hasA*, a simplified version of data abstraction mechanisms discussed in the semantic data modeling literature [12, 10]. *isA* represents the supertype/subtype relationship, while *hasA* refers to aggregation or association. For example in Arts/(Painting,Dance), Arts is a supertype while Painting and Dance are subtypes, Also, in Computer/(Hardware,Software), Computer is an aggregation of Hardware and Software, that is a relationship between objects is considered as a higher level object. Finally, in Computer/Companies, Companies is an association of Computers, that is a collection of objects is considered as a higher level object.

Faceted classification is another aspect of portal catalogs hierarchies [11]. Data provided by portal catalogs can be classified in *facets*. For example range of prices, geographical areas, condition, etc are facets. low-price, mid-price, high-price are the *headings* of the facet range of prices, while new, second hand are the headings of the facet condition. Figure 5 shows a hierarchy S that has the categorization basic, dark room and books, headings of the facet product type, as a top level choice during browsing. Other headings could be used, combined with already used headings to further expand the hierarchy, for example low-price, mid-price, high-price. Facet headings can form many different hierarchies in the form of tree structures, that alone or combined can classify data provided by a portal catalog, depending on the user's need.

However, the user is neither aware of the semantics provided nor of the different facets used to construct the hierarchy. What is available in a portal catalog is usually a tree-like structured hierarchy that will assist her during a browsing retrieval task. The construction of a such a tree-like structured hierarchy for a portal catalog can be based on a *global* catalog available for the specific application domain or community.

Figure 5 presents an example of such a catalog S for photo equipment. Figure 6 shows two catalogs S_1 and S_2 based on global catalog S . In such case, portal catalogs are actually parts of this global catalog and may have only inconsistencies related to naming resolving, but not structural inconsistencies. For example, in Figure 6, Single Reflex and ultra violet in S_1 are SLR and uv in S_2 . On the other hand, if such a global catalog is not available, structural inconsistencies may occur. See for example the catalog S_1 in Figure 6. The author of another catalog similar to S_1 might prefer having the categorization new and second hand as a top level choice during browsing, which is inconsistent with S_1 .

In the following sections we start exploring the algebraic properties of trees representing hierarchies of portal catalogs, based on the assumption that there is a global catalog available for the specific application

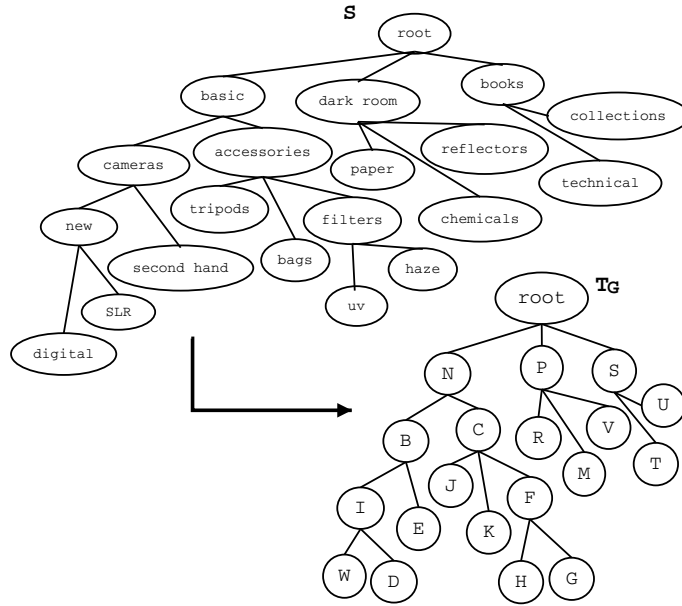


Figure 5. A global catalog S for photo equipment and its representative global tree T_G .

domain or community. Then, we discuss a framework to ensure that similar algebraic properties hold without having such a global catalog, based on the construction of a new catalog to act as a global one.

3. Algebraic properties of portal catalog hierarchies

We represent hierarchies for portal catalogs as rooted labeled trees. Such trees may be the output of schema matching tools that identify semantic similarities and resolve conflicts (see Section 1.3). Figures 5, 6 show examples of such trees that we will use throughout this work as presentation examples. We note that similar labels are used for nodes in one catalog that match nodes of another catalog.

Definition 3.1. A rooted labeled tree, or simply tree, T is

1. a root node r or
2. a root node r and a set \mathcal{E} of edges, $\mathcal{E} = \{e_1, e_2, \dots, e_k\}$, with $e_1 : r \rightarrow r_1$, $e_2 : r \rightarrow r_2$, \dots , $e_k : r \rightarrow r_k$, where r_1, r_2, \dots, r_k are root nodes of k (sub)trees.

A function $label(n)$ assigns a string label as the identifier for every node n of T . Especially for r , $label(r) = root$. The tree that has root r as its only node is denoted as T_0 .

Node A is the *parent* of node B , denoted as $p(A, B)$, if there is a direct edge from A to B . Node A is an *ancestor* of node B , denoted as $a(A, B)$, if there is a direct path of k edges, $k > 1$, from A to B . A tree T with nodes $N = \{n_1, n_2, \dots, n_k\}$ and root r can be represented as a relation \mathcal{P} on the set $\{r\} \cup N$, denoted as $\langle \{r\} \cup N, \mathcal{P} \rangle$: $x\mathcal{P}y$ holds if $p(x, y)$ holds, that is node x is the parent of node y , with $x, y \in \{r\} \cup N$.

Two trees are equal if they have the same nodes and the same structural relationships among them, as the following definition shows.

Definition 3.2. $T_i = T_j$, iff $N_i = N_j$ and $\forall x, y \in \{r\} \cup N_i$ with $x\mathcal{P}_i y$ then $x\mathcal{P}_j y$.

\mathcal{P}^{tr} denotes the transitive closure of \mathcal{P} , that is $x\mathcal{P}^{tr}y$ holds if for any elements $x, y \in \{r\} \cup N$ there exist c_0, c_1, \dots, c_n with $c_0 = x$, $c_n = y$ and $c_p\mathcal{P}c_{p+1}$ for all $0 \leq p < n$.

We now define the S -subset (\subseteq^S) relation for 2 trees. Intuitively, when $T_1 \subseteq^S T_2$, all nodes of T_1 exist in T_2 . Also, by deleting all nodes of T_2 not present in T_1 and moving their children in higher levels we will obtain T_1 .

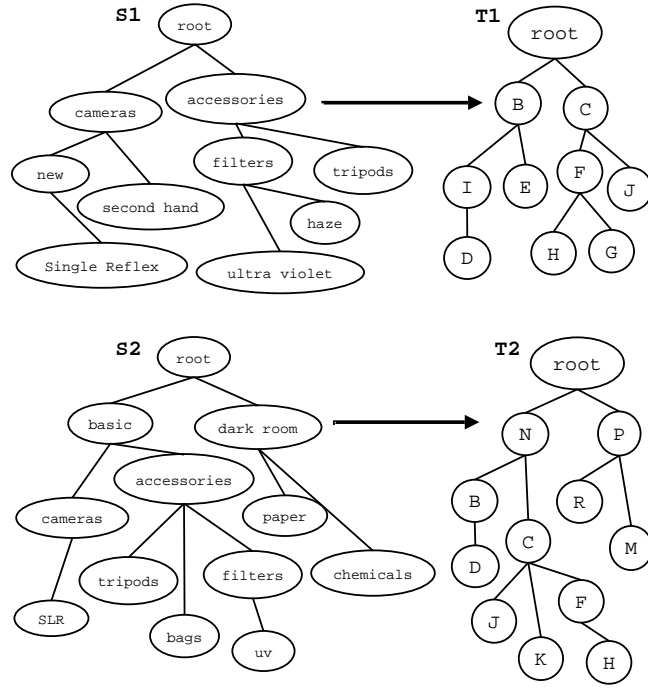


Figure 6. Catalogs S_1 , S_2 and their representative trees T_1 and T_2 .

Definition 3.3. A tree $T_i = \langle \{r\} \cup N_i, \mathcal{P}_i \rangle$ is an S -subset of a tree $T_j = \langle \{r\} \cup N_j, \mathcal{P}_j \rangle$, denoted as $T_i \subseteq^s T_j$, if $\forall x, y \in \{r\} \cup N_i$ with $x\mathcal{P}_i y$ then $x\mathcal{P}_j^{tr} y$ and if $\forall x, y \in \{r\} \cup N_i$ with $x\mathcal{P}_j^{tr} y$ then $x\mathcal{P}_i^{tr} y$. Especially for T_0 , $T_0 \subseteq^s T$ for any $T = \langle \{r\} \cup N, \mathcal{P} \rangle$.

For example, in Figure 7, $T_1 \subseteq^s T_3$. T_1 can be constructed by deleting nodes D and C from T_3 and moving E and F up in a higher level. $T_2 \not\subseteq^s T_3$ since $p(E, C)$ holds in T_2 but not in T_3 . Also, $T_2 \not\subseteq^s T_4$ since $p(\text{root}, C)$ in T_4 but $a(\text{root}, C)$ in T_2 . According to the next lemma, two trees are equal if they have the same number of nodes and are related with an S -subset relation.

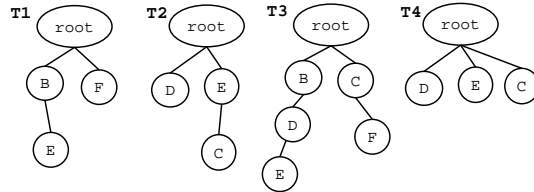


Figure 7. $T_1 \subseteq^s T_3$, $T_2 \not\subseteq^s T_3$.

Lemma 3.1. Let $T_i = \langle \{r\} \cup N_i, \mathcal{P}_i \rangle$ and $T_j = \langle \{r\} \cup N_j, \mathcal{P}_j \rangle$ be two trees. If $N_i = N_j$ and $T_i \subseteq^s T_j$ then $T_i = T_j$.

Proof. Directly from Definitions 3.3 and 3.2. □

Let $T_G = \langle \{r\} \cup N_G, \mathcal{P}_G \rangle$ be a tree which we call *global tree* and $\mathcal{S}_G = \{T_i : T_i \subseteq^s T_G\}$, that is all trees which are S -subsets of T_G . Figure 5 shows an example of a global tree T_G and Figure 6 shows trees T_1 and T_2 , with $T_1 \subseteq^s T_G$ and $T_2 \subseteq^s T_G$. Note that \mathcal{S}_G includes T_0 , which is the tree with only one node (its root), and T_G . Using Theorem 3.1, we claim that the set \mathcal{S}_G equipped with the relation \subseteq^s , $\langle \mathcal{S}_G, \subseteq^s \rangle$, is a partial order.

Theorem 3.1. The \subseteq^s relation on \mathcal{S}_G , $\langle \mathcal{S}_G, \subseteq^s \rangle$, is

1. reflexive: $T_i \subseteq^s T_i$,

2. *antisymmetric*: $T_i \subseteq^s T_j$ and $T_j \subseteq^s T_i$ imply $T_i = T_j$, and
3. *transitive*: $T_i \subseteq^s T_j$ and $T_j \subseteq^s T_k$ imply $T_i \subseteq^s T_k$.

Proof. Reflexivity holds (directly from Definition 3.3). Antisymmetry holds: if $T_i \subseteq^s T_j$ and $T_j \subseteq^s T_i$, then $N_i \subseteq N_j$ and $N_j \subseteq N_i$, and, thus, $N_i = N_j$. Given $T_i \subseteq^s T_j$ and $N_i = N_j$, $T_i = T_j$ holds (Lemma 3.1). Transitivity holds, too. If $T_i \subseteq^s T_j$ and $T_j \subseteq^s T_k$, then

$$\forall x, y \in \{r\} \cup N_i \text{ with } x\mathcal{P}_i y \Rightarrow x\mathcal{P}_j^{tr} y \quad (1)$$

$$\forall x, y \in \{r\} \cup N_j \text{ with } x\mathcal{P}_j y \Rightarrow x\mathcal{P}_k^{tr} y \quad (2)$$

From (1), for all $x, y \in \{r\} \cup N_i$ there exist c_0, c_1, \dots, c_n with $c_0 = x$, $c_n = y$ and $c_p\mathcal{P}_i c_{p+1}$ for all $0 \leq p < n$. For $n = 1$, $x\mathcal{P}_j y$ holds, and, thus, (2) gives $x\mathcal{P}_k^{tr} y$. For $n > 1$, $x\mathcal{P}_j c_1$, $c_1\mathcal{P}_j c_2$, \dots , $c_{n-1}\mathcal{P}_j c_n$ hold, thus, $x\mathcal{P}_k^{tr} c_1$, $c_1\mathcal{P}_k^{tr} c_2$, \dots , $c_{n-1}\mathcal{P}_k^{tr} c_n$ hold, and this gives $x\mathcal{P}_k^{tr} y$. \square

We note that the partial order $\langle \mathcal{S}_G, \subseteq^s \rangle$ has T_0 as its bottom element \perp , with the property that $\perp \subseteq^s T_i$, and T_G as its top element \top , with the property that $T_i \subseteq^s \top$, for all $T_i \in \mathcal{S}_G$.

We next define the first 2 operators for structural manipulation of trees in \mathcal{S}_G , representing hierarchies for portal catalogs: *S-union* and *S-intersection*. Both operators are binary, in the sense that they get two trees as input and produce a new tree. Exploiting these operators we will show that the partial order $\langle \mathcal{S}_G, \subseteq^s \rangle$ is a lattice structure.

3.1. S-union (\cup^s)

Intuitively, the *S-union* of two trees $T_i, T_j \in \mathcal{S}_G$, $T_i \cup^s T_j$, is the tree $T \in \mathcal{S}_G$ which provides integrated structural information from T_i and T_j . Figure 8 presents an example of an *S-union* operation on trees T_1 and T_2 , both *S*-subsets of T_G (Figure 5), resulting to the tree $T_3 = T_1 \cup^s T_2$. T_3 contains all nodes from both trees T_1 and T_2 . Common parent or ancestor relationships in T_1 and T_2 are preserved in T_3 . For example $p(C, J)$ and $a(C, H)$, common in T_1 and T_2 , exist in T_3 , too. In the case where different relationships involve nodes common in T_1 and T_2 , the ancestor one is preserved in T_3 . For example, given $a(B, D)$ in T_1 and $p(B, D)$ in T_2 , T_3 keeps $a(B, D)$. Also, every unique relationship, either in T_1 or in T_2 , is preserved in T_3 . For example, T_3 contains $p(B, E)$ which is unique in T_1 . A more

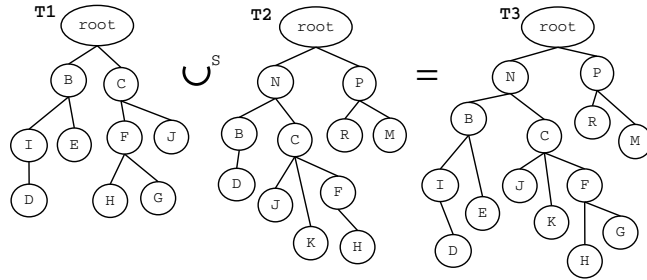


Figure 8. Union operation: $T_3 = T_1 \cup^s T_2$.

interesting example is presented in Figure 9, which shows a variation of trees T_1 and T_2 of the previous example. Following the same way to construct $T_1 \cup^s T_2$, we get T_4 . However, since T_4 lacks $p(F, H)$ which is present in T_G , $T_4 \not\subseteq^s T_G$. Thus, $T_1 \cup^s T_2$ should also contain relationships neither present in T_1 nor in T_2 , but imposed by T_G , like for example $p(F, H)$. Using $p(F, H)$ we get T_3 . So, $T_1 \cup^s T_2$, with $T_1 = \langle \{r\} \cup N_1, \mathcal{P}_1 \rangle$ and $T_2 = \langle \{r\} \cup N_2, \mathcal{P}_2 \rangle$, can be constructed by deleting all nodes of T_G not present in $N_1 \cup N_2$ and moving their children in higher levels accordingly. The definition for the *S-union* operation follows.

Definition 3.4. Let $T_i = \langle \{r\} \cup N_i, \mathcal{P}_i \rangle$ and $T_j = \langle \{r\} \cup N_j, \mathcal{P}_j \rangle$ be two trees in \mathcal{S}_G . The *S-union* of T_i and T_j , $T_i \cup^s T_j$, is the tree $T = \langle \{r\} \cup N, \mathcal{P} \rangle$ constructed as follows:

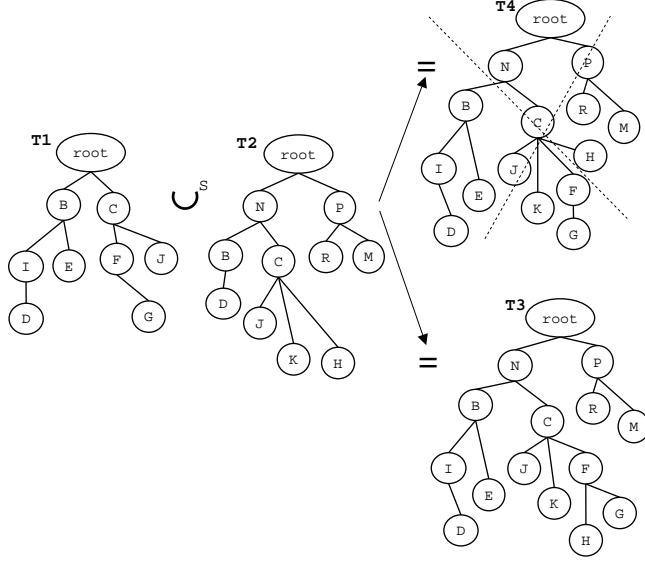


Figure 9. Union operation: $T_3 = T_1 \cup^s T_2$.

- $N = N_i \cup N_j$:
T has all nodes from both T_i and T_j .
- $x\mathcal{P}y$ if $x\mathcal{P}_G y$, with $x, y \in \{r\} \cup N$:
T keeps all parent structural relationships $p(x, y)$ from T_G that involve nodes x, y present in either T_i or T_j .
- $x\mathcal{P}y$, if $\neg x\mathcal{P}_G y$ and $x\mathcal{P}_G^{tr} y$ and all c_1, c_2, \dots, c_{n-1} in $x\mathcal{P}_G c_1, c_1\mathcal{P}_G c_2, \dots, c_{n-1}\mathcal{P}_G y$, $n \geq 2$, are not in N ($x, y \in \{r\} \cup N$):
T uses as parent relationships all ancestor relationships $a(x, y)$ from T_G that involve nodes x, y present in either T_i or T_j , in the path of which all nodes belong neither to T_i nor to T_j .

We note that

1. $(T_i \cup^s T_j) \subseteq^s T_G$, thus $(T_i \cup^s T_j) \in \mathcal{S}_G$.
2. $(T_i \cup^s T_j)$ is an *upper bound* of $\{T_i, T_j\} \subset \mathcal{S}_G$, i.e. $T_i \subseteq^s (T_i \cup^s T_j)$ and $T_j \subseteq^s (T_i \cup^s T_j)$, since by the way we construct $T_i \cup^s T_j$, it keeps all parent relationships $p(x, y)$ from T_G that involve nodes x, y present either in T_i or in T_j .

3.2. *S*-intersection (\cap^s)

Intuitively, the *S-intersection* of two trees $T_i, T_j \in \mathcal{S}_G$, $T_i \cap^s T_j$, is the tree $T \in \mathcal{S}_G$ which provides structural information common in T_i and T_j . Figure 10 presents an example of an *S-intersection* operation on trees T_1 and T_2 , both *S*-subsets of T_G (Figure 5), resulting to the tree $T_3 = T_1 \cap^s T_2$. T_3 contains all common nodes in trees T_1 and T_2 . Common parent relationships in T_1 and T_2 are preserved in T_3 . For example $p(C, J)$ and $p(C, F)$, common in T_1 and T_2 , exist in T_3 , too. In the case where different relationships involve nodes common in T_1 and T_2 , the parent one is preserved in T_3 . For example, given $a(B, D)$ in T_1 and $p(B, D)$ in T_2 , T_3 keeps $p(B, D)$. Common ancestor relations in T_1 and T_2 are preserved in T_3 only if the intermediate nodes are common in T_1 and T_2 , otherwise they become parent relations. For example, since $a(C, H)$ in T_1 , $a(C, H)$ in T_2 and the intermediate node F is a common node, T_3 keeps $a(C, H)$. Similarly to *S*-union, $T_1 \cap^s T_2$, with $T_1 = \langle \{r\} \cup N_1, \mathcal{P}_1 \rangle$ and $T_2 = \langle \{r\} \cup N_2, \mathcal{P}_2 \rangle$, can be constructed by deleting all nodes of T_G not present in $N_1 \cap N_2$ and moving their children in higher levels accordingly. The definition for the *S-intersection* operation follows.

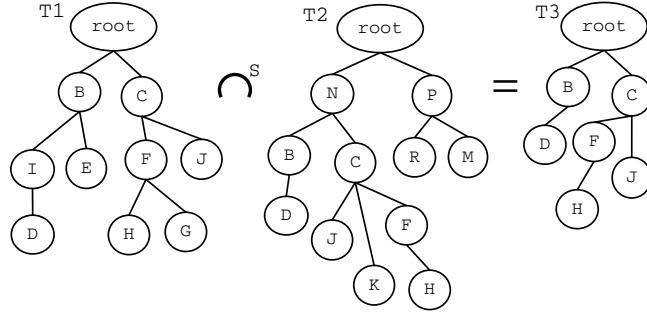


Figure 10. Intersection operation: $T_3 = T_1 \cap^S T_2$.

Definition 3.5. Let $T_i = \langle \{r\} \cup N_i, \mathcal{P}_i \rangle$ and $T_j = \langle \{r\} \cup N_j, \mathcal{P}_j \rangle$ be two trees in \mathcal{S}_G . The S -intersection of T_i and T_j , $T_i \cap^S T_j$, is the tree $T = \langle \{r\} \cup N, \mathcal{P} \rangle$ constructed as follows:

- $N = N_i \cap N_j$:
 T has all nodes common in T_i and T_j .
- $x\mathcal{P}y$ if $x\mathcal{P}_G y$, with $x, y \in \{r\} \cup N$:
 T keeps all parent structural relationships $p(x, y)$ from T_G that involve nodes x, y present in both T_i and T_j .
- $x\mathcal{P}y$, if $\neg x\mathcal{P}_G y$ and $x\mathcal{P}_G^{tr} y$ and all c_1, c_2, \dots, c_{n-1} in $x\mathcal{P}_G c_1, c_1\mathcal{P}_G c_2, \dots, c_{n-1}\mathcal{P}_G y$, $n \geq 2$, are not in N ($x, y \in \{r\} \cup N$):
 T uses as parent relationships all ancestor relationships $a(x, y)$ from T_G that involve nodes x, y present in both T_i and T_j , in the path of which all nodes do not belong to the set of common nodes of T_i and T_j .

We note that

1. $(T_i \cap^S T_j) \subseteq^S T_G$, thus $(T_i \cap^S T_j) \in \mathcal{S}_G$.
2. $(T_i \cap^S T_j)$ is a lower bound of $\{T_i, T_j\} \subset \mathcal{S}_G$, i.e. $(T_i \cap^S T_j) \subseteq^S T_i$ and $(T_i \cap^S T_j) \subseteq^S T_j$, since by the way we construct $T_i \cap^S T_j$, it keeps only the parent structural relationships $p(x, y)$ from T_G that involve nodes x, y present in both T_i and T_j .

4. Portal catalog hierarchies as algebraic structures

This section goes further by using the \cup^S and \cap^S operators to show that the partial order $\langle \mathcal{S}_G, \subseteq^S \rangle$ is a lattice and keeps the properties of a boolean algebra. In the previous section we showed that \cup^S gives an upper bound and \cap^S gives a lower bound of the two trees from \mathcal{S}_G involved in these two operations. As Theorem 4.1 shows, \cup^S gives the *least upper bound*, i.e. the upper bound which is an S -subset, \subseteq^S , of all upper bounds of the two trees involved, while \cap^S gives the *greatest lower bound*, i.e. the lower bound which all lower bounds of the two trees involved are S -subset, \subseteq^S , of.

Theorem 4.1. Let $T_i = \langle \{r\} \cup N_i, \mathcal{P}_i \rangle$ and $T_j = \langle \{r\} \cup N_j, \mathcal{P}_j \rangle$ be two trees in \mathcal{S}_G . The set of all upper bounds of \mathcal{S}_G is $\mathcal{S}_G^u = \{T_k \in \mathcal{S}_G : T_i \subseteq^S T_k \text{ and } T_j \subseteq^S T_k\}$. The set of all lower bounds of \mathcal{S}_G is $\mathcal{S}_G^l = \{T_k \in \mathcal{S}_G : T_k \subseteq^S T_i \text{ and } T_k \subseteq^S T_j\}$. Then

1. $(T_i \cup^S T_j) \subseteq^S T_p$, for all T_p in \mathcal{S}_G^u , that is $T_i \cup^S T_j$ gives the least upper bound of T_i and T_j ,
2. $T_p \subseteq^S (T_i \cap^S T_j)$, for all T_p in \mathcal{S}_G^l , that is $T_i \cap^S T_j$ gives the greatest lower bound of T_i and T_j .

Proof. Let T be a tree for which $T \subseteq^s (T_i \cup^s T_j)$, $T_i \subseteq^s T$ and $T_j \subseteq^s T$. T should have either less nodes than $T_i \cup^s T_j$, or the same number of nodes. In the first case, either $T_i \not\subseteq^s T$ or $T_j \not\subseteq^s T$. In the second case, $T = T_i \cup^s T_j$ (Lemma 3.1). So there is not such a tree T other than $T_i \cup^s T_j$.

Let now T be a tree for which $(T_i \cap^s T_j) \subseteq^s T$, $T \subseteq^s T_i$ and $T \subseteq^s T_j$. T should have either more nodes than $T_i \cap^s T_j$, or the same number of nodes. In the first case, either $T \not\subseteq^s T_i$ or $T \not\subseteq^s T_j$. In the second case, $T = T_i \cap^s T_j$ (Lemma 3.1). So there is not such a tree T . \square

The least upper bound and the greatest lower bound exist for all trees T_i, T_j in the partial order on \mathcal{S}_G equipped with the relation \subseteq^s , $\langle \mathcal{S}_G, \subseteq^s \rangle$. Thus, $\langle \mathcal{S}_G, \subseteq^s \rangle$ is a lattice.

Theorem 4.2. *The \mathcal{S}_G equipped with the relation \subseteq^s , $\langle \mathcal{S}_G, \subseteq^s \rangle$ is a lattice.*

Proof. Directly from Theorems 3.1 and 4.1. \square

Since $\langle \mathcal{S}_G, \subseteq^s \rangle$ is a lattice, idempotency, commutative, associative and absorption laws hold [8], as Table 1 shows. We note (see [8]) that for any a, b, c in a lattice $\langle L, \geq \rangle$, with \vee denoting the least upper

Idempotency laws	$T_i \cap^s T_i = T_i$ $T_i \cup^s T_i = T_i$
Commutative laws	$T_i \cap^s T_j = T_j \cap^s T_i$ $T_i \cup^s T_j = T_j \cup^s T_i$
Associative laws	$T_i \cap^s (T_j \cap^s T_k) = (T_i \cap^s T_j) \cap^s T_k$ $T_i \cap^s (T_j \cap^s T_k) = (T_i \cap^s T_j) \cap^s T_k$
Absorption laws	$T_i \cap^s (T_i \cup^s T_j) = T_i$ $T_i \cup^s (T_i \cap^s T_j) = T_i$
Distributive laws	$T_i \cup^s (T_j \cap^s T_k) = (T_i \cup^s T_j) \cap^s (T_i \cup^s T_k)$ $T_i \cap^s (T_j \cup^s T_k) = (T_i \cap^s T_j) \cup^s (T_i \cap^s T_k)$
De Morgan's laws	$T_i -^s (T_j \cup^s T_k) = (T_i -^s T_j) \cap^s (T_i -^s T_k)$ $T_i -^s (T_j \cap^s T_k) = (T_i -^s T_j) \cup^s (T_i -^s T_k)$

Table 1. Laws hold for $\langle \mathcal{S}_G, \subseteq^s \rangle$.

bound and \wedge the greatest lower bound,

$$a \wedge (b \vee c) \geq (a \wedge b) \vee (a \wedge c) \quad (3)$$

and dually

$$a \vee (b \wedge c) \geq (a \vee b) \wedge (a \vee c) \quad (4)$$

that is equality in the distributive laws for lattices does not hold in general. Theorem 4.3 shows that $\langle \mathcal{S}_G, \subseteq^s \rangle$ is a distributive lattice, that is the ditributive laws hold (see Table 1).

Theorem 4.3. *The \mathcal{S}_G equipped with the relation \subseteq^s , $\langle \mathcal{S}_G, \subseteq^s \rangle$, is a distributive lattice.*

Proof. Let $T_i = \langle \{r\} \cup N_i, \mathcal{P}_i \rangle$, $T_j = \langle \{r\} \cup N_j, \mathcal{P}_j \rangle$, and $T_k = \langle \{r\} \cup N_k, \mathcal{P}_k \rangle$, all members of \mathcal{S}_G . Also, let $T_1 = T_i \cup^s (T_j \cap^s T_k) = \langle \{r\} \cup N_1, \mathcal{P}_1 \rangle$ and $T_2 = (T_i \cup^s T_j) \cap^s (T_i \cup^s T_k) = \langle \{r\} \cup N_2, \mathcal{P}_2 \rangle$. Using Definitions 3.4 and 3.5, $N_1 = N_i \cup (N_j \cap N_k)$ and $N_2 = (N_i \cup N_j) \cap (N_i \cup N_k)$, thus $N_1 = N_2$.

Recall that the output of an S -union (or S -intersection) operator is the tree constructed by deleting all nodes of T_G not present in the union of nodes in the involved trees (or in the intersection of nodes in the involved trees) and moving their children in higher levels accordingly. Thus, since T_1 and T_2 involve the same nodes from T_G , $T_1 = T_2$. \square

We next provide $\langle \mathcal{S}_G, \subseteq^s \rangle$ with additional structure to support *complements*.

4.1. Complements

Intuitively, the complement of a tree $T_i \in \mathcal{S}_G$ is the tree $T'_i \in \mathcal{S}_G$ which provides structural information present in T_G and not in T_i . Figure 11 presents the complement T'_1 of tree $T_1 = \langle \{r\} \cup N_1, \mathcal{P}_1 \rangle$ (see Figure 6 for T_1 and Figure 5 for $T_G = \langle \{r\} \cup N_G, \mathcal{P}_G \rangle$). T'_1 keeps the root of T_G and all of its nodes not present in T_1 . T'_1 can be constructed by deleting all nodes of T_G not present in $N_G - N_1$ and moving their children in higher levels accordingly. We note that given T_i, T'_i in \mathcal{S}_G , $T_i \cap^s T'_i = \mathbf{0}$ and $T_i \cup^s T'_i = \mathbf{1}$,

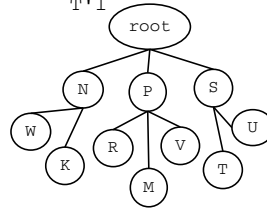


Figure 11. T'_1 : the complement of T_1 .

where $\mathbf{0}$ and $\mathbf{1}$ are two unique trees in \mathcal{S}_G . Theorem 4.4 provides the support for complements.

Theorem 4.4. Let $T_i = \langle \{r\} \cup N_i, \mathcal{P}_i \rangle$ be a tree in \mathcal{S}_G , $\mathbf{0} \equiv T_0$ and $\mathbf{1} \equiv T_G = \langle \{r\} \cup N_G, \mathcal{P}_G \rangle$. The tree $T'_i = \langle \{r\} \cup N'_i, \mathcal{P}'_i \rangle$, constructed as follows, is the unique complement of T_i :

- $N'_i = N_G - N_i$:
 T'_i has all nodes present in T_G and not in T_i .
- $x\mathcal{P}'_iy$ if $x\mathcal{P}_Gy$, with $x, y \in \{r\} \cup N'_i$:
 T'_i keeps all parent structural relationships $p(x, y)$ from T_G that involve nodes x, y present in T_G but not in T_i .
- $x\mathcal{P}'iy$, if $\neg x\mathcal{P}_Gy$ and $x\mathcal{P}_G^{tr}y$ and all c_1, c_2, \dots, c_{n-1} in $x\mathcal{P}_Gc_1, c_1\mathcal{P}_Gc_2, \dots, c_{n-1}\mathcal{P}_Gy$, $n \geq 2$, are not in N'_i ($x, y \in \{r\} \cup N'_i$):
 T uses as parent relationships all ancestor relationships $a(x, y)$ from T_G that involve nodes x, y present in T_G but not in T_i , in the path of which all nodes do not belong to the set nodes present in T_G but not in T_i .

Proof. $T'_i \subseteq^s T_G$, thus $T'_i \in \mathcal{S}_G$. The root node r is the only common node of T_i and T'_i , thus $T_0 \equiv \mathbf{0} = T_i \cap^s T'_i$. $(T_i \cup^s T'_i) \subseteq^s T_G$ and $(T_i \cup^s T'_i)$ has the same number of nodes with T_G , thus $T_i \cup^s T'_i = T_G \equiv \mathbf{1}$ (see Lemma 3.1). Finally, since $\langle \mathcal{S}_G, \subseteq^s \rangle$ is a distributive lattice, the complement found for every tree is unique [8]. \square

Since $\langle \mathcal{S}_G, \subseteq^s \rangle$

1. is a distributive lattice,
2. has $\mathbf{0} \equiv T_0$ and $\mathbf{1} \equiv T_G$, and
3. each T_i in \mathcal{S}_G has a unique complement T'_i in \mathcal{S}_G ,

it is also a *boolean lattice* [8], getting all properties of boolean algebra.

Theorem 4.5. The \mathcal{S}_G equipped with the relation \subseteq^s , $\langle \mathcal{S}_G, \subseteq^s \rangle$ is a boolean lattice.

Proof. Directly from Theorems 4.3 and 4.4. \square

Exploiting the complements, we next define the last binary operator for structural reasoning on trees representing hierarchies for portal catalogs: *S-difference*.

4.2. S -difference ($-^s$)

Intuitively, the S -difference of two trees $T_j, T_i \in \mathcal{S}_G$, $T_j -^s T_i$, is the tree which provides structural information present in T_j and not in T_i . Figure 12 presents an example of an S -difference operation resulting to the tree $T_3 = T_2 -^s T_1$. T_3 can be constructed using complements: $T_3 = T_2 \cap^s T_1'$. The

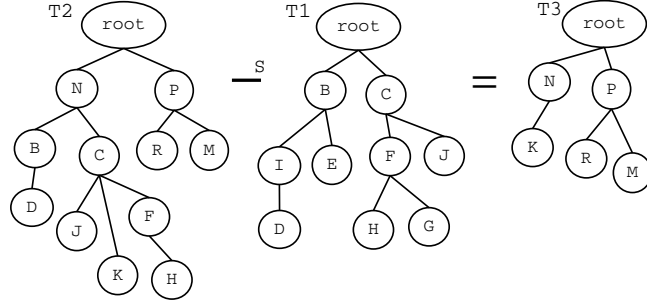


Figure 12. Difference operation: $T_3 = T_2 -^s T_1$.

definition for the S -difference operation follows.

Definition 4.1. Let $T_i = \langle \{r\} \cup N_i, \mathcal{P}_i \rangle$ and $T_j = \langle \{r\} \cup N_j, \mathcal{P}_j \rangle$ be two trees in \mathcal{S}_G . The S -difference of T_j and T_i , $T_j -^s T_i$, is the tree $T = T_j \cap^s T_i'$.

Since $\langle \mathcal{S}_G, \subseteq^s \rangle$ is a boolean lattice, the De Morgan's laws involving the S -difference ($-^s$), the S -intersection \cap^s and the S -union \cup^s operators hold [8]. Table 1 shows all laws hold for $\langle \mathcal{S}_G, \subseteq^s \rangle$.

5. Lacking the global catalog

In the previous sections, we explored the algebraic properties of trees representing hierarchies of portal catalogs, based on the assumption that all of these trees are S -subsets of a tree available for the specific application domain or community, called global tree. In the absence of such a global tree, we need to provide a framework to ensure that similar algebraic properties hold, providing a new tree T_G to act as a global one. Our target is to construct such a valid global tree T_G based on the available trees in such a way that all trees are S -subsets of T_G .

The task resembles the S -union operation presented in Section 3.1 in the sense that the constructed T_G should provide integrated structural information from all available trees representing hierarchies of portal catalogs. However, contrary to the S -union operation, the construction of T_G should be based on the structural relationships coming only from the available trees. Figure 13 shows trees T_1 and T_2 and the constructed tree T_G . T_G contains all nodes from both trees T_1 and T_2 . Common parent or ancestor relationships in T_1 and T_2 are preserved in T_G . For example $p(C, J)$ and $a(C, H)$, common in T_1 and T_2 , exist in T_G , too. In the case where different relationships involve nodes common in T_1 and T_2 , the ancestor one is preserved in T_G . For example, given $a(B, D)$ in T_1 and $p(B, D)$ in T_2 , T_G keeps $a(B, D)$. Also, every unique relationship, either in T_1 or in T_2 , is preserved in T_G . The construction of T_G , given two trees T_i and T_j , is formalized as follows.

Definition 5.1. Let $T_i = \langle \{r\} \cup N_i, \mathcal{P}_i \rangle$ and $T_j = \langle \{r\} \cup N_j, \mathcal{P}_j \rangle$ be two trees. The global tree of T_i and T_j is the tree $T_G = \langle \{r\} \cup N_G, \mathcal{P}_G \rangle$, constructed as follows:

- $N_G = N_i \cup N_j$:
 T_G has all nodes from both T_i and T_j .
- $x\mathcal{P}_G y$ if $x\mathcal{P}_i y$ and $x\mathcal{P}_j y$, with $x, y \in \{r\} \cup N_G$:
 T_G keeps all parent relationships $p(x, y)$ common to both T_i and T_j .
- $x\mathcal{P}_G y$, if $x\mathcal{P}_i y$ and $\neg x\mathcal{P}_j^{tr} y$, or if $x\mathcal{P}_j y$ and $\neg x\mathcal{P}_i^{tr} y$ ($x, y \in \{r\} \cup N_G$):
 T_G keeps all parent relationships $p(x, y)$ from T_i (T_j) that involve nodes x, y related neither with parent $p(x, y)$ nor with ancestor relationship $a(x, y)$ in T_j (T_i).

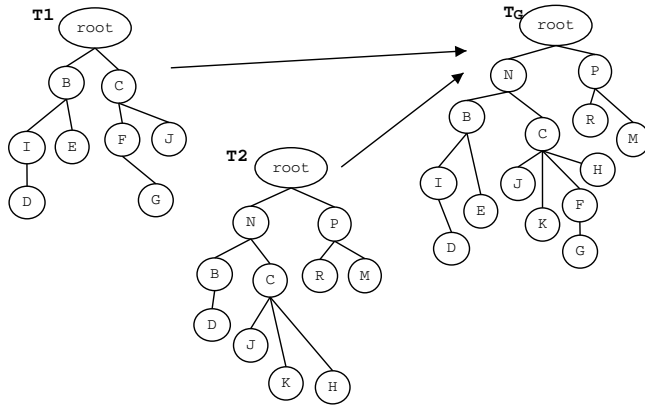


Figure 13. Trees T_1 and T_2 and the constructed tree T_G .

Consider now the trees T_2 and T_5 in Figure 14. Which of the two relationships, $p(E, C)$ and $p(F, C)$, should T_G keep? See also the trees T_2 and T_3 in Figure 14, where $p(E, C)$ in T_1 , while E, C are children of the same node in T_2 . Should T_G keep the E, C as children or just keep $p(E, C)$? To deal with

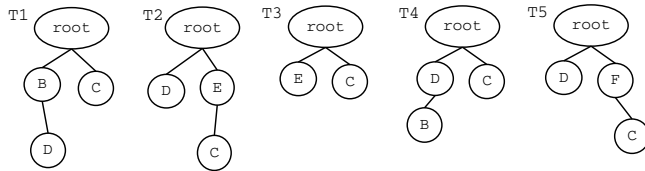


Figure 14. Trees to be examined for consistency and semantic compatibility.

such issues, we introduce the notions of *consistency* and *semantic compatibility* for trees [22]. Before we proceed, we set up a proper notation and terminology to simplify our study. We consider p (parent) and a (ancestor) relationships to be *direct* relationships, in the sense that there is always a directed path connecting two nodes having such kind of relationship. In case that two nodes are involved neither in a p nor in an a relationship, we consider that they are related with an *indirect* relationship, in the sense that there is no directed path connecting two nodes, but these nodes are related via a common parent or ancestor. $Dir^T(A, B)$ denotes a direct relationship, while $In^T(A, B)$ an indirect relationship between nodes A and B in tree T .

5.1. Consistency

Consistency ensures that the involved trees will not include nodes whose structural relationships are in conflict. For example, trees T_2 and T_3 in Figure 14 are not consistent, since $p(E, C)$ in T_1 , while E, C are children of the same node in T_2 . First we define consistency on a pair of trees, and then we expand the definition for a set of trees. The former is not enough for ensuring the construction of a T_G without structural conflicts. See the example of trees T_1, T_2 and T_3 in Figure 15. Constructing T_G^{23} from T_2 and T_3 , and then T_G from T_1 and T_G^{23} gives a T_G without structural conflicts, while constructing T_G^{12} from T_1 and T_2 , and then T_G from T_G^{12} and T_3 is vague due to nodes B and C .

Definition 5.2. Two trees T_i and T_j are consistent (or a pair of trees is consistent) if for every pair of nodes that belongs to both trees, the type of node relationship, under the perspective of direct vs indirect relationships, is the same in both trees.

Trees T_1 and T_2 of Figure 14 have the pairs of nodes (A, D) , (A, C) and (D, C) in common. The relationship of the first two pairs is direct and of the third is indirect in both trees; therefore trees T_1 and T_2 are consistent. Trees T_2 and T_3 of Figure 14 have the pairs (A, E) , (A, C) and (E, C) in common. The relationship of the first two pairs is direct in both trees, but nodes E and C are related with a

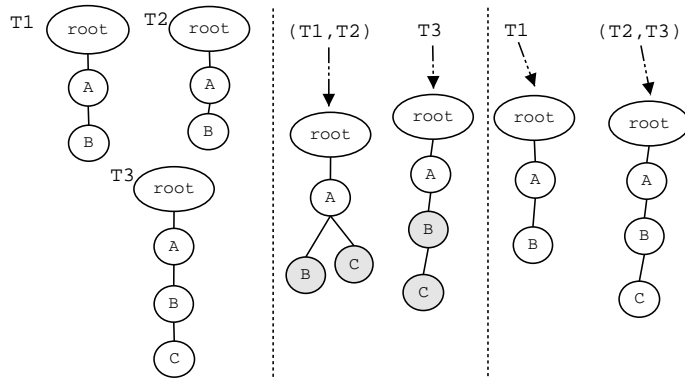


Figure 15. Constructing T_G given 3 trees: see the problem in (a) concerning B and C nodes.

direct relationship in tree T_2 and with an indirect relationship in tree T_3 . Therefore, trees T_2 and T_3 are inconsistent. According to the next proposition, if two trees have the same number of nodes and are consistent, they are equal.

Proposition 5.1. *If trees T_i and T_j are consistent, and they both contain the same nodes, then $T_i = T_j$.*

Proof. Since T_1 and T_2 are consistent, then for every pair of nodes $(A, B) \in T_1$ with $Dir^{T_1}(A, B)$, $Dir^{T_2}(A, B)$ holds in T_2 . Suppose that $T_1 \neq T_2$. Then there exists a pair (A, B) for which $p(A, B) \in T_1$ and $a(A, B) \in T_2$. Therefore, there is a node C such that $Dir^{T_2}(A, C)$ and $Dir^{T_2}(C, B)$. Since both trees contain the same nodes, C should be in T_1 , too. T_1 and T_2 are consistent, so both $Dir^{T_1}(A, C)$ and $Dir^{T_1}(C, B)$ should hold, but this is not the case because $p(A, B) \in T_1$. Thus, there is not a node C between A and B in T_2 . So, for every pair (A, B) where $p(A, B) \in T_1$, $p(A, B) \in T_2$. Therefore, $T_1 = T_2$. \square

We next introduce the notion of *stability* for direct relationships. Intuitively, stability ensures that new indirect relationships produced in the T_G will never involve nodes which are already related with direct relationships. Stability is required for the definition of consistency for a set of trees that will follow later on.

Definition 5.3. *Assume a set \mathcal{S} of n trees, every pair of which is consistent, and the pair of nodes A and B , with $Dir^{T_i}(A, B)$ in one of the trees T_i in \mathcal{S} . We choose from \mathcal{S} all the trees that contain exactly one of the nodes of the pair (either A or B but not both), and partition them in two sets: \mathcal{G}_A is the set of trees that contain node A and \mathcal{G}_B is the set of trees that contain node B . Iff for every pair of trees (T_A, T_B) with $T_A \in \mathcal{G}_A$ and $T_B \in \mathcal{G}_B$ there is a node X such that $Dir^{T_A}(A, X)$ and $Dir^{T_B}(X, B)$, then the direct relationship of nodes A and B , $Dir(A, B)$, is called stable in the set \mathcal{S} .*

We note that:

1. If $\mathcal{G}_A = \emptyset$ or $\mathcal{G}_B = \emptyset$ then the direct relationship of nodes A and B is stable in the set \mathcal{S} .
2. If a relationship $Dir(A, B)$ is stable in a set \mathcal{S} of trees, then the T_G produced by any two trees of the set will not have an $In(A, B)$ relationship.
3. If a relationship $Dir(A, B)$ is not stable in a set \mathcal{S} of trees, then there exists at least one pair of trees of the set for which the produced T_G has an $In(A, B)$ relationship.
4. If a relationship $Dir(A, B)$ is stable in a set \mathcal{S} of trees, then the relationship is stable in any set $\mathcal{S}' \subseteq \mathcal{S}$.

Some examples follow:

1. Consider the set \mathcal{S}' of trees T_1, T_2, T_3 and T_4 in Figure 16(a). Every pair of trees in \mathcal{S}' is consistent and $Dir^{T_1}(B, D)$. T_2 and T_3 both contain node D but not B , therefore $\mathcal{G}_D = \{T_2, T_3\}$. T_4 contains node B but not D , therefore $\mathcal{G}_B = \{T_4\}$. However, there is no node X such that $Dir^{T_4}(B, X)$ and $Dir^{T_2, T_3}(X, D)$, therefore the relationship of B and D is not stable in set \mathcal{S}' .
2. Consider the set \mathcal{S}'' of trees of trees T_1, T_2 and T_3 in Figure 16(b). Every pair of trees in \mathcal{S}'' is consistent and $Dir^{T_1}(A, B)$. We put all trees that contain node A but not B in group \mathcal{G}_A , and all trees that contain node B but not A in \mathcal{G}_B . Therefore, $\mathcal{G}_A = \{T_2\}$ and $\mathcal{G}_B = \{T_3\}$. Tree T_2 in \mathcal{G}_A has a node C for which $Dir^{T_2}(A, C)$. Tree T_3 in \mathcal{G}_B has the same node C for which $Dir^{T_3}(C, B)$. Therefore the relationship of nodes A and B is stable in \mathcal{S}'' .

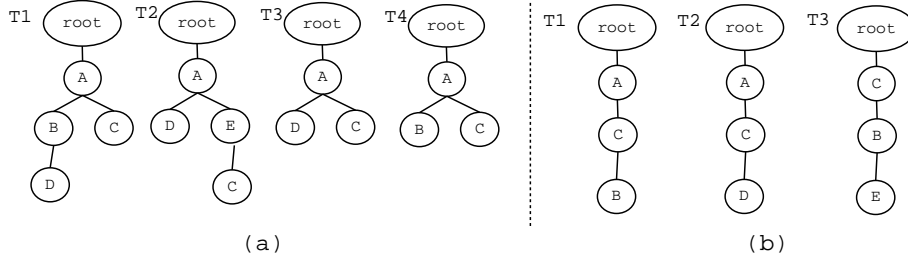


Figure 16. Example trees to check stability.

We next define the notion of *consistent* set of trees, that is trees with only stable direct relationships, every pair of which is consistent (see Definition 5.2).

Definition 5.4. Let \mathcal{S} be a set of n trees and \mathcal{N} the set of all nodes of all trees in \mathcal{S} . \mathcal{S} is consistent if both the following conditions hold:

1. Every pair of trees in \mathcal{S} is consistent.
2. Every direct relationship $Dir(A, B)$, with $A, B \in \mathcal{N}$, is stable in \mathcal{S} .

We note that:

1. Since all trees under consideration have the same root, we can omit all pairs containing the root to check the above condition. The relationship between the root and any of the nodes X is stable in any set of trees. There can be no tree in the set containing X and not the root, therefore the second group \mathcal{G}_X will be empty.
2. A set containing exactly 2 trees is always consistent if the two trees are consistent according to the definition 5.2.

Some examples follow:

1. Consider the set $\mathcal{S} = \{T_1, T_2, T_3\}$ in Figure 16(a). The set of all nodes is $\mathcal{N} = \{A, B, C, D, E\}$, and all the pairs of nodes, excluding those containing the root, are: (B, C) , (B, D) , (B, E) , (C, D) , (C, E) , (D, E) . From all these pairs, only (B, D) and (C, E) are in a direct relationship in at least one tree of \mathcal{S} . The relationships of B and D , and C and E are stable in \mathcal{S} , thus \mathcal{S} is consistent.
2. On the other hand, $\mathcal{S}' = \{T_1, T_2, T_3, T_4\}$ (the full set of trees in Figure 16(a)) is not consistent, since the relationship of nodes B and D is not stable.

5.2. Semantic compatibility

We next define semantic compatibility for a pair of trees. Semantic compatibility ensures that we will always be able to decide the structural relationship between the involved nodes in the new tree T_G . For example, trees T_2 and T_5 in Figure 14 are not semantically compatible, since it is not clear whether $p(E, C)$ or $p(F, C)$ in T_G .

Definition 5.5. Let $T_i = \langle \{r\} \cup N_i, \mathcal{P}_i \rangle$ and $T_j = \langle \{r\} \cup N_j, \mathcal{P}_j \rangle$ be two trees and $N = N_i \cap N_j$. T_i and T_j are semantically compatible if $\forall X \in N, \exists Y$ with $p(Y, X)$ in T_i or T_j with $Y \in N$.

In the following section we exploit the notions of tree consistency and tree semantic compatibility to study the validity of T_G construction.

5.3. Validity of global tree construction

The construction of a valid T_G requires that the involved trees are consistent and every pair of the trees semantically compatible. We next present a set of theorems to support such an argument. As the following theorem states, the construction of T_G preserves the tree semantic compatibility in the tree set.

Theorem 5.1. Let \mathcal{S} be a set of trees, semantically compatible with each other. The tree $T_G = \langle \{r\} \cup N_G, \mathcal{P}_G \rangle$ constructed from any two trees $T_i = \langle \{r\} \cup N_i, \mathcal{P}_i \rangle$ and $T_j = \langle \{r\} \cup N_j, \mathcal{P}_j \rangle$ in \mathcal{S} is semantically compatible with all other trees of \mathcal{S} .

Proof. Let $T_k = \langle \{r\} \cup N_k, \mathcal{P}_k \rangle$ a tree in \mathcal{S} . Since T_i and T_k are semantically compatible, $\forall X \in N_i \cap N_k, \exists Y$ with $p(Y, X)$ in T_i or T_k with $Y \in N_i \cap N_k$. Similarly, since T_j and T_k are semantically compatible, $\forall X \in N_j \cap N_k, \exists Y$ with $p(Y, X)$ in T_j or T_k with $Y \in N_j \cap N_k$. The parent node of a node X in T_G will be the same with the parent node of X in T_i or in T_j . Thus, $\forall X \in N_G \cap N_k, \exists Y$ with $p(Y, X)$ in T_G or T_k with $Y \in N_G \cap N_k$, that is T_G is semantically compatible with T_k . \square

The construction of T_G should preserve tree consistency in the tree set. To simplify the proof of the related theorem, we first present 3 relevant lemmas.

Lemma 5.1. Let \mathcal{S} be a consistent set of n trees. The tree T_G constructed from any two trees in \mathcal{S} will produce a set \mathcal{S}' of $(n - 1)$ trees, in which every two trees are consistent.

Proof. Consider the sets $\mathcal{S} = \{T_1, T_2, T_3, \dots, T_n\}$ and $\mathcal{S}' = \{T, T_3, \dots, T_n\}$, where $T = T_G$ is constructed from T_1 and T_2 . Every pair of trees in \mathcal{S} is consistent, and every direct relationship is stable in \mathcal{S} . We will show that T is consistent with every tree in set $\mathcal{R} = \{T_3, \dots, T_n\}$. Consider a pair of nodes (A, B) in tree T . We assume that there is at least one tree T_x in \mathcal{R} that contains both A and B . Otherwise, there would be no inconsistency caused by this pair. There are two possibilities:

1. A and $B \in T_1$ (or T_2): Since T_1 and T_2 are consistent with every tree in \mathcal{R} , the type of the relationship (A, B) (direct, indirect) in a tree $T_x \in \mathcal{R}$ is the same with tree T_1 (or T_2) and, thus, the same with $T = T_G$.
2. $A \in T_1$ and $B \in T_2$:

(a) $Dir^T(A, B)$

There exists a node C such that $Dir^{T_1}(A, C)$ and $Dir^{T_2}(C, B)$.

i. If $C \in T_x \implies Dir^{T_x}(A, C)$ and $Dir^{T_x}(C, B) \implies Dir^{T_x}(A, B)$

ii. If $C \notin T_x$: $G_A = \{T_x, \dots\}$ and $G_C = \{T_2, \dots\}$. Since \mathcal{S} is consistent, $Dir(A, C)$ stable in $\mathcal{S} \implies \exists D : Dir^{T_x}(A, D)$ and $Dir^{T_2}(D, C) \implies Dir^{T_x}(A, D)$ and $Dir^{T_2}(D, B) \implies Dir^{T_x}(A, D)$ and $Dir^{T_x}(D, B) \implies Dir^{T_x}(A, B)$.

(b) $In^T(A, B)$

Suppose that $Dir^{T_x}(A, B)$. $G_A = \{T_1, \dots\}$ and $G_B = \{T_2, \dots\}$. $Dir(A, B)$ stable in $\mathcal{S} \implies \exists C : Dir^{T_1}(A, C)$ and $Dir^{T_2}(C, B) \implies Dir^T(A, B)$ but we assumed otherwise. Therefore $In^{T_x}(A, B)$ holds.

□

Lemma 5.2. *If a set \mathcal{S} of trees is consistent then every direct relationship, existing in set \mathcal{S} , is stable in the set $\mathcal{S}' = \mathcal{S} \cup \{T_G\}$, where T_G is constructed from any two trees of \mathcal{S} .*

Proof. We consider that $T = T_G$ is constructed from two trees $T_1, T_2 \in \mathcal{S}$. Every $Dir(A, B)$ is stable in \mathcal{S} , so there exist \mathcal{G}_A and \mathcal{G}_B . We will check the stability of $Dir(A, B)$ in \mathcal{S}' .

1. If $Dir(A, B) \in T$ or $(A \notin T \text{ and } B \notin T)$, then there is no change in \mathcal{G}_A and \mathcal{G}_B , thus $Dir(A, B)$ is stable in \mathcal{S}' .
2. If $A \in T$ and $B \notin T$, then $\mathcal{G}_A = \mathcal{S}_A \cup \{T\}$ and $\mathcal{G}_B = \mathcal{S}_B$. Since $A \in T$ and $B \notin T$, it must be $A \in T_1$ and $B \notin T_1 \implies T_1 \in \mathcal{S}_A^2$. Tree T contains all structural information of tree T_1 . For every pair (T_1, T_B) , where $T_B \in \mathcal{S}_B$, there exists a node X such that $Dir^{T_1}(A, X)$ and $Dir^{T_B}(X, B)$. Consequently, for every pair (T, T_B) , where $T_B \in \mathcal{S}_B$, there exists a node X such that $Dir^T(A, X)$ and $Dir^{T_B}(X, B)$. Therefore, $Dir(A, B)$ is stable in \mathcal{S}' .
3. If $A \notin T$ and $B \in T$, then $\mathcal{G}_A = \mathcal{S}_A$ and $\mathcal{G}_B = \mathcal{S}_B \cup \{T\}$. Since $B \in T$ and $A \notin T$, it must be $B \in T_1$ and $A \notin T_1 \implies T_1 \in \mathcal{S}_B^3$. Tree T contains all structural information of tree T_1 . For every pair (T_A, T_1) , where $T_A \in \mathcal{S}_A$, there exists a node X such that $Dir^{T_A}(A, X)$ and $Dir^{T_1}(X, B)$. Consequently, for every pair (T_A, T) , where $T_A \in \mathcal{S}_A$, there exists a node X such that $Dir^{T_A}(A, X)$ and $Dir^T(X, B)$. Therefore, $Dir(A, B)$ is stable in \mathcal{S}' .

Note also that $Dir(A, B)$ is also stable in set $\mathcal{S}'' = \mathcal{S} \cup \{T\} - \{T_1, T_2\}$, because $\mathcal{S}'' \subset \mathcal{S}'$. □

Lemma 5.3. *Let \mathcal{S} be a consistent set of 4 trees. The tree T_G constructed from any two trees in \mathcal{S} results in a set \mathcal{S}' with 3 trees which is also consistent.*

Proof. Consider $\mathcal{S} = \{T_1, T_2, T_3, T_4\}$ and $\mathcal{S}' = \{T, T_3, T_4\}$, where $T = T_G$ is constructed from T_1 and T_2 . Each pair of trees in \mathcal{S} is consistent (Lemma 5.1). Every direct relationship $Dir(A, B)$ in \mathcal{S} , is stable in set \mathcal{S}' (Lemma 5.2). Therefore, we should check the stability of a direct relationship $Dir(A, B) \in T$, such that $Dir(A, B) \notin \mathcal{S}$. Since $Dir(A, B)$ does not exist in trees T_1 or T_2 , $A \in T_1$ and $B \in T_2^4$.

For $Dir^T(A, B)$ to hold, $Dir^{T_1}(A, C)$ and $Dir^{T_2}(C, B)$ should hold. If $A \notin \{T_3, T_4\}$ or $B \notin \{T_3, T_4\} \implies (A, B)$ is stable in \mathcal{S}' . Assume that $A \in T_3$ and $B \in T_4^5$. Checking the stability of $Dir(A, B)$ in \mathcal{S}' results in: $\mathcal{G}_A = \{T_3\}$ and $\mathcal{G}_B = \{T_4\}$. In order for (A, B) to be stable in \mathcal{S}' , there must be a node X such that $Dir^{T_3}(A, X)$ and $Dir^{T_4}(X, B)$. We will seek for such a node.

1. $C \in T_3$ and $C \in T_4$: $X = C$
2. $C \notin T_3$ and $C \notin T_4$: since $Dir(A, C)$ is stable in \mathcal{S} , $\mathcal{G}_A = \{T_3\}$ and $\mathcal{G}_C = \{T_2\}$. Thus, $\exists D : Dir^{T_3}(A, D)$ and $Dir^{T_2}(D, C) \implies Dir^{T_2}(D, B)$.
 - (a) If $D \in T_4$ then $X = D$.
 - (b) If $D \notin T_4$, then, since $Dir(D, B)$ is stable in \mathcal{S} , $\mathcal{G}_D = \{T_3\}$ and $\mathcal{G}_B = \{T_4\}^6$. Thus, $\exists F : Dir^{T_3}(D, F)$ and $Dir^{T_4}(F, B) \implies Dir^{T_3}(A, F)$ and $Dir^{T_4}(F, B) \implies X = F$.
3. $C \in T_3$ and $C \notin T_4$: since $Dir(C, B)$ is stable in \mathcal{S} , $\mathcal{G}_C = \{T_1, T_3\}$ and $\mathcal{G}_B = \{T_4\}$. Thus $\exists D : Dir^{T_3}(C, D)$ and $Dir^{T_4}(D, B) \implies Dir^{T_3}(A, D)$ and $Dir^{T_4}(D, B) \implies X = D$.
4. $C \notin T_3$ and $C \in T_4$: since $Dir(A, C)$ is stable in \mathcal{S} , $\mathcal{G}_A = \{T_3\}$ and $\mathcal{G}_C = \{T_2, T_4\}$. Thus, $\exists D : Dir^{T_3}(A, D)$ and $Dir^{T_4}(D, C) \implies Dir^{T_3}(A, D)$ and $Dir^{T_4}(D, B) \implies X = D$.

²The proof is similar for $A \in T_2$ and $B \notin T_2$.

³The proof is similar for $A \notin T_2$ and $B \in T_2$.

⁴The proof is similar for $A \in T_2$ and $B \in T_1$.

⁵Since $Dir(A, B) \notin \mathcal{S}$, A and B will not be in the same tree.

⁶It may also be $T_1 \in \mathcal{G}_D$ or $T_1 \in \mathcal{G}_B$, but this does not affect the result.

Therefore, $Dir(A, B) \in T$ is stable in \mathcal{S}' , and hence, \mathcal{S}' is consistent. \square

We next prove that the construction of T_G preserves tree consistency in the tree set.

Theorem 5.2. *Let \mathcal{S} be a consistent set of n trees. The tree T_G constructed from any two trees in \mathcal{S} results in a set \mathcal{S}' with $n - 1$ trees which is also consistent.*

Proof. Theorem holds for $n = 4$ (Lemma 5.3). We assume that it holds for $n = k$, too. We will prove that it holds for $n = k + 1$. Assume the consistent sets $\mathcal{S}_k = \{T_1, T_2, T_3, \dots, T_k\}$ and $\mathcal{S}'_k = \{T, T_3, \dots, T_k\}$, where $T = T_G$ is constructed from T_1 and T_2 . We refer to the set $\{T_3, \dots, T_k\}$ as the set \mathcal{R} . So, $\mathcal{S}_k = \{T_1, T_2\} \cup \mathcal{R}$ and $\mathcal{S}'_k = \{T\} \cup \mathcal{R}$. Consider now the set $\mathcal{S}_{k+1} = \{T_1, T_2, \dots, T_k, T_{k+1}\} = \{T_1, T_2\} \cup \mathcal{R} \cup \{T_{k+1}\}$, which is consistent. We construct a global tree using two trees from \mathcal{S}_{k+1} , getting the set \mathcal{S}'_{k+1} . There are 2 options for such a construction:

1. $\mathcal{S}'_{k+1} = \{T, T_{k+1}\} \cup \mathcal{R}$, where $T = T_G$ is constructed from T_1 and T_2 (Case 1).
2. $\mathcal{S}'_{k+1} = \{T', T_2\} \cup \mathcal{R}$, where $T' = T'_G$ is constructed from T_1 and T_{k+1} (Case 2).

The first condition of definition 5.4 is satisfied for both 1. and 2. (Lemma 5.1). Moreover, every relationship $Dir(A, B)$ in \mathcal{S}_{k+1} is stable in \mathcal{S}'_{k+1} (Lemma 5.2). Therefore, we only have to check the stability of a $Dir(A, B) \in T$ or a $Dir(A, B) \in T'$, which does not exist in set \mathcal{S}_{k+1} . \mathcal{R}_A denotes the subset of \mathcal{R} containing only node A and not B , and \mathcal{R}_B denotes the subset of \mathcal{R} containing only node B and not A .

Case 1

$(A, B) \in T$. Since $Dir(A, B)$ is stable in \mathcal{S}'_k , $\mathcal{G}_A = \mathcal{R}_A$ and $\mathcal{G}_B = \mathcal{R}_B$.

1. If $(A, B) \in T_{k+1}$ or $(A \notin T_{k+1} \text{ and } B \notin T_{k+1})$ then checking the stability of (A, B) in \mathcal{S}'_{k+1} gives $\mathcal{G}_A = \mathcal{R}_A$ and $\mathcal{G}_B = \mathcal{R}_B \implies Dir(A, B)$ stable in \mathcal{S}'_{k+1} .
2. If $A \in T_{k+1}$ and $B \notin T_{k+1}$ then checking the stability of (A, B) in \mathcal{S}'_{k+1} gives $\mathcal{G}_A = \mathcal{R}_A \cup \{T_{k+1}\}$ and $\mathcal{G}_B = \mathcal{R}_B$.

(a) $\mathcal{R}_A \neq \emptyset$ and $\mathcal{R}_B \neq \emptyset$

For every pair (T_A, T_B) , $T_A \in \mathcal{R}_A$ and $T_B \in \mathcal{R}_B$, $\exists C : Dir^{T_A}(A, C)$ and $Dir^{T_B}(C, B)$, because $Dir(A, B)$ stable in \mathcal{S}'_k . If T_{k+1} contains all these nodes C , then $Dir(A, B)$ is stable in \mathcal{S}'_{k+1} . If there is a node $C \notin T_{k+1}$, then, since $Dir(A, C)$ is stable in \mathcal{S}_{k+1} , $T_{k+1} \in \mathcal{G}_A$ and $\mathcal{R}'_B \subseteq \mathcal{G}_C$, \mathcal{R}'_B is the subset of \mathcal{R}_B that contains C . Therefore, for every pair (T_{k+1}, T_B) , $T_B \in \mathcal{R}'_B$, there exists a node D such that $Dir^{T_{k+1}}(A, D)$ and $Dir^{T_B}(D, B) \implies Dir^{T_{k+1}}(A, D)$ and $Dir^{T_B}(D, B)$. So, $Dir(A, B)$ is stable in \mathcal{S}'_{k+1} .

(b) $\mathcal{R}_B = \emptyset$ then $Dir(A, B)$ is stable in \mathcal{S}'_{k+1} .

(c) $\mathcal{R}_A = \emptyset$

Since $Dir^T(A, B) \implies \exists C : Dir^{T_A}(A, C)$ and $Dir^{T_B}(C, B)$. We define the sets $\mathcal{R}_1 = \{T_i : T_i \in \mathcal{R}_B \text{ and } C \in T_i\}$ and $\mathcal{R}_2 = \{T_i : T_i \in \mathcal{R}_B \text{ and } C \notin T_i\}$ ($\mathcal{R}_B = \mathcal{R}_1 \cup \mathcal{R}_2$)

i. $C \in T_{k+1}$

Since $Dir(C, B)$ is stable in \mathcal{S}_{k+1} , $\mathcal{G}_C = \{T_1, T_{k+1}\}$ and $\mathcal{G}_B = \mathcal{R}_2 \implies$ for every pair (T_{k+1}, T_x) , $T_x \in \mathcal{R}_2$, $\exists D : Dir^{T_{k+1}}(C, D)$ and $Dir^{T_x}(D, B) \implies Dir^{T_{k+1}}(A, D)$ and $Dir^{T_x}(D, B)$. We also have that $Dir^{T_{k+1}}(A, C)$ and $Dir^{T_y}(C, B)$ for every $T_y \in \mathcal{R}_1$, therefore $Dir(A, B)$ is stable in \mathcal{S}'_{k+1} .

ii. $C \notin T_{k+1}$

Since $Dir(A, C)$ is stable in \mathcal{S}_{k+1} , $\mathcal{G}_A = \{T_{k+1}\}$ and $\mathcal{G}_C = \mathcal{R}_1 \cup \{T_2\} \implies$ for every pair (T_{k+1}, T_y) , $T_y \in \mathcal{R}_1$, $\exists D : Dir^{T_{k+1}}(A, D)$ and $Dir^{T_y}(D, C) \implies Dir^{T_{k+1}}(A, D)$ and $Dir^{T_y}(D, B)$. Since $Dir(C, B)$ is stable in

$\mathcal{S}_{k+1}, G_C = \{T_1\}$ and $\mathcal{G}_B = R_2 \implies$ for every pair $(T_1, T_x), T_x \in R_2, \exists E : Dir^{T_1}(C, E)$ and $Dir^{T_x}(E, B) \implies Dir^{T_1}(A, E)$ and $Dir^{T_x}(E, B)$.

A. If $E \in T_{k+1}$ then $Dir(A, B)$ is stable in \mathcal{S}'_{k+1} .

B. If $E \notin T_{k+1}$, then, since $Dir(A, E)$ is stable in $\mathcal{S}_{k+1}, \mathcal{G}_A = \{T_{k+1}\}$ and $G_E = R_2 \implies$ for every pair $(T_{k+1}, T_x), T_x \in R_1,$
 $\exists F : Dir^{T_{k+1}}(A, F)$ and
 $Dir^{T_x}(F, E) \implies Dir^{T_{k+1}}(A, F)$ and $Dir^{T_x}(F, B)$.

Therefore, for every pair $(T_{k+1}, T_B),$ with $T_B \in \mathcal{R}_B,$ there is a node $X (X = D, E \text{ or } F)$ such that $Dir^{T_{k+1}}(A, X)$ and $Dir^{T_B}(X, B),$ so $Dir(A, B)$ is stable in \mathcal{S}'_{k+1} .

3. If $B \in T_{k+1}$ and $A \notin T_{k+1}$ then the case is symmetrical to the previous one and it is handled the same way.

Case 2

$\mathcal{S}'_{k+1} = \{T'\} \cup \mathcal{V},$ where $\mathcal{V} = R \cup \{T_2\},$ and $T' = T_1 \cup^S T_{k+1}.$ Checking the stability of $Dir(A, B)$ in \mathcal{S}'_{k+1} gives $\mathcal{G}_A = \mathcal{V}_A$ and $\mathcal{G}_B = \mathcal{V}_B,$ where \mathcal{V}_A is the subset of \mathcal{V} containing node A and not $B,$ and \mathcal{V}_B is the subset of \mathcal{V} containing node B and not $A.$ If $\mathcal{V}_A = \emptyset$ or $\mathcal{V}_B = \emptyset$ then $Dir(A, B)$ is stable in $\mathcal{S}'_{k+1}.$

If $\mathcal{V}_A \neq \emptyset$ and $\mathcal{V}_B \neq \emptyset$ we assume that $A \in T_1$ and $B \in T_{k+1}.$ Since $Dir^{T'}(A, B) \implies \exists C : Dir^{T_1}(A, C)$ and $Dir^{T_{k+1}}(C, B).$ Since $Dir(C, B)$ is stable in $\mathcal{S}_{k+1}, T_1 \in G_C$ and $\mathcal{V}'_B \in \mathcal{G}_B,$ where \mathcal{V}'_B is the subset of \mathcal{V}_B that does not contain node $C.$ $Dir(C, B)$ is stable in $\mathcal{S}_{k+1},$ thus for every pair $(T_1, T_B), T_B \in \mathcal{V}'_B, \exists D : Dir^{T_1}(C, D)$ and $Dir^{T_B}(D, B).$ All trees are chosen at random, and therefore T_2 does not represent a specific tree. For any T_2 in $\mathcal{V}_B, Dir^{T_2}_G(A, B)$ holds in T_G constructed from T_1 and $T_2.$

We have assumed that the proposition is satisfied for set $\mathcal{S}_k.$ Therefore, the set \mathcal{S}'_k produced after the construction of T_G using two trees of \mathcal{S}_k should be consistent. T_1 and T_2 can be such trees. Consequently, $Dir(A, B)$ is stable in the set $\mathcal{S}'_k = \{T\} \cup R.$ Since $Dir(A, B)$ is stable in $\mathcal{S}'_k, \mathcal{G}_A = \mathcal{R}_A$ and $\mathcal{G}_B = \mathcal{R}_B.$ Thus, for every pair $(T_A, T_B), T_A \in \mathcal{R}_A$ and $T_B \in \mathcal{R}_B, \exists D : Dir^{T_A}(A, D)$ and $Dir^{T_B}(D, B).$ It is $\mathcal{V}_A = \mathcal{R}_A$ and $\mathcal{V}_B = \mathcal{R}_B \cup \{T_2\}.$ Therefore, if $D \in T_2$ then $Dir(A, B)$ is stable in $\mathcal{S}'_{k+1}.$ If there is a node $D \notin T_2,$ then, since $Dir(D, B)$ is stable in $\mathcal{S}_{k+1}, \mathcal{R}'_A \subseteq G_D$ and $T_2 \in \mathcal{G}_B,$ where \mathcal{R}'_A is the subset of \mathcal{R}_A that contains $D.$ Thus, for every pair (T_x, T_2) where $T_x \in \mathcal{R}'_A, \exists F : Dir^{T_x}(D, F)$ and $Dir^{T_2}(F, B) \implies Dir^{T_x}(A, F)$ and $Dir^{T_2}(F, B).$ Therefore, for every pair $(T_x, T_2),$ where $T_x \in \mathcal{R}_A,$ there exists a node $X (X = D \text{ or } F)$ such that $Dir^{T_x}(A, X)$ and $Dir^{T_2}(X, B).$ Thus, $Dir(A, B)$ is stable in $\mathcal{S}'_{k+1}.$ In every case $Dir(A, B)$ is stable in $\mathcal{S}'_{k+1},$ so \mathcal{S}'_{k+1} is consistent. \square

There is still the question whether there are non-consistent sets of trees for which the construction of T_G makes the set consistent. The next theorem shows that Definition 5.4 for consistent set of trees covers all tree sets for which T_G construction produces consistent tree sets.

Theorem 5.3. *Let \mathcal{S} be a set of n trees, every pair of which is consistent. The tree T_G constructed from any two trees in \mathcal{S} results in a set \mathcal{S}' with $n - 1$ trees. If every pair of trees in \mathcal{S}' is consistent, then both \mathcal{S} and \mathcal{S}' are consistent.*

Proof. Suppose that \mathcal{S} is not consistent. There is at least one direct relationship in $\mathcal{S},$ which is not stable in $\mathcal{S}.$ Let $Dir(A, B)$ be such a relationship. There must exist (a) a tree $T_1,$ such that $A \in T_1$ and $B \notin T_1,$ (b) a tree $T_2,$ such that $A \notin T_2$ and $B \in T_2,$ and (c) without a node C such that $Dir^{T_1}(A, C)$ and $Dir^{T_2}(C, B).$ Since the set of A 's descendants in T_1 and the set of B 's ancestors in T_2 have no nodes in common, then $In^T(A, B)$ holds in $T = T_G$ constructed from T_1 and $T_2.$ But since $Dir(A, B)$ holds in $\mathcal{S},$ there is a tree $T_3 \in \mathcal{S}$ with $Dir^{T_3}(A, B).$ T_3 also belongs to set \mathcal{S}' and T_3 is not consistent with $T.$ So, if \mathcal{S} is not consistent, then the pairs of trees in \mathcal{S}' are not consistent. Consequently, if the pairs of trees in \mathcal{S}' are consistent, then \mathcal{S} must be consistent. Also, Theorem 5.2 shows that \mathcal{S}' is consistent. \square

The construction of T_G according to Definition 5.1 is a binary operation in the sense that 2 trees are used to build $T_G.$ Having k trees, $k > 2,$ say $\{T_1, T_2, T_3, \dots, T_k\},$ one can build a T_G^{12} from T_1 and $T_2,$

then a T_G^{123} from T_G^{12} and T_3, \dots , and finally the T_G from $T_G^{12\dots(k-1)}$ and T_k . The T_G constructed from many trees should be the same regardless the sequence of the operation. The next theorem ensures that requirement.

Theorem 5.4. *Let \mathcal{S} be a consistent set of n trees. The tree T_G constructed from all trees is of \mathcal{S} is the same, regardless the sequence in which the construction is performed.*

Proof. Consider that constructing T_G in \mathcal{S} in a certain order produces tree T_G^1 , while in a different order produces tree T_G^2 . We will show that $T_G^1 = T_G^2 = T_G$. $\mathcal{S}' = \mathcal{S} \cup \{T_G^1\}$ is consistent because T_G^1 is consistent with every tree in \mathcal{S} . Every relationship is stable in \mathcal{S}' , because T_G^1 contains all nodes of \mathcal{S} , and therefore it does not participate in the stability checking sets. Since \mathcal{S}' is consistent, any T_G construction \mathcal{S}' will leave the set consistent, as shown in Theorem 5.2. Therefore, we can perform repeated T_G constructions in \mathcal{S}' in such order that tree T_G^2 is produced, using all trees of \mathcal{S} with the appropriate order. Since after T_G construction the set remains consistent, trees T_G^1 and T_G^2 are consistent. Trees T_G^1 and T_G^2 contain the same set of nodes, which is the set of all nodes in \mathcal{S} . Therefore, according to Proposition 5.1, $T_G^1 = T_G^2$. \square

6. Motivation example revisited

Based on the motivating example in Section 1, we now express the queries mentioned there (see Figures 1, 2, 4) using the suggested S -union, S -intersection and S -difference operators. For the presented examples, T_G is the tree presented in Figure 3.

1. Merge Adorama's and B&H catalogs: $H_1 \cup^s H_2$.
2. Find the common part of Adorama's and B&H catalogs: $H_1 \cap^s H_2$.
3. Find the part of Adorama's catalog which is not present in B&H's catalog: $H_1 -^s H_2$.
4. Merge RitzCamera's catalog with the common part of both Adorama's and B&H catalogs: $H_3 \cup^s (H_1 \cap^s H_2)$.

Two more complicated examples follow:

1. Find the part of Adorama's catalog which is not present in the merged catalog produced from B&H's and RitzCameras catalogs: $H_1 -^s (H_2 \cup^s H_3)$.
According to Definition 4.1, $H_1 -^s (H_2 \cup^s H_3) = H_1 \cap^s (H_2 \cup^s H_3)'$. The nodes involved in $(H_2 \cup^s H_3)'$ are: caps, hoods, Close Up, UV, PL, film, B&W, slide. Figure 17 illustrates $H_2 \cup^s H_3$ as well as the final result.
2. Take the part of Adorama's catalog which is not present in B&H's catalog and the part of Adorama's catalog which is not present in RitzCameras catalog, and find their common part: $(H_1 -^s H_2) \cap^s (H_1 -^s H_3)$.
Figure 18 shows the intermediate results $H_1 -^s H_2 = H_1 \cap^s H_2'$ and $H_1 -^s H_3 = H_1 \cap^s H_3'$, as well as the final result.

Notice that the result of both queries is the same, since they obey one of the De Morgan's laws presented in the previous section.

7. Conclusions and further work

This work suggested a framework for the structural manipulation of vertical portal catalog sites, that is catalogs of a specific subject or domain. We summarize our work as follows:

1. We presented a tree-like representation for hierarchies of vertical portal catalog sites.

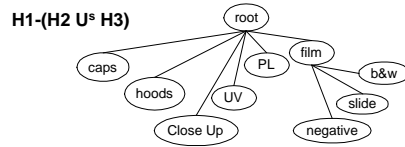
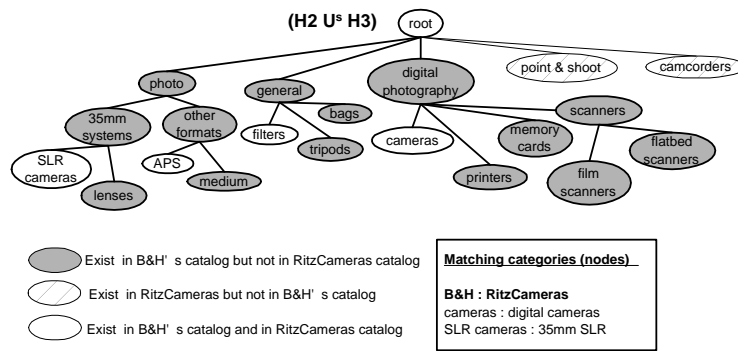


Figure 17. Example 1

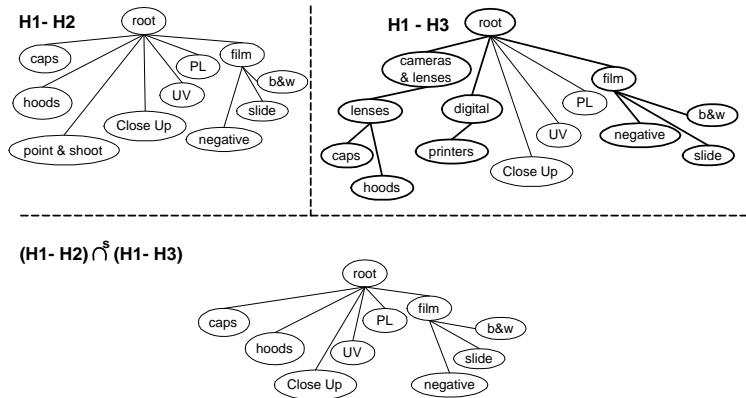


Figure 18. Example 2

2. We explored the algebraic properties of such trees and defined a lattice algebraic structure on them. Exploiting the upper and lower bounds of such a lattice, we formally defined the first two operators for structural manipulation of vertical portal catalog sites: S -union and S -intersection. The S -union operator provides an integrated form of all structural information from the involved trees. The S -intersection operator provides the common structural information from the involved trees.
3. Then, turning this lattice structure into a boolean algebra, we defined the S -difference operator based on the notion of complement trees. The S -difference operator provides structural information present in one tree and not in the other.
4. The suggested framework provides operators with clear semantics and certain properties to assist the transformation, simplification and optimization of sequences of operations, using a set of laws similar to those in set theory.
5. Finally, we identified the conditions under which this framework is applicable, using a global catalog, which is either provided for the specific domain or constructed using the involved trees. For the latter case, we introduced the notion of tree consistency and compatibility.

Our future work is based on three directions. First, we will further explore the algebraic properties of trees representing hierarchies of portal catalogs in the absence of a global catalog and its global tree. We showed that construction of a valid global tree requires that the involved trees are consistent and

every pair of the trees semantically compatible. In fact, even if only the first requirement holds, we are still able to construct a valid global tree. Lack of semantic compatibility means that we will not be able to decide the structural relationship between some of the involved nodes in the global tree. However, all the different global trees produced using several interpretations for those relationships are valid, that is all involved trees are S -subsets of all global trees. Moreover, exploiting the algebraic properties of the suggested operators and the laws hold, we plan to provide a basis for optimization on sequences of operations. Finally, we will integrate all suggested operators within a path-expression query language to support both structural manipulation on the data schema and data querying.

References

- [1] Serge Abiteboul and Catriel Beeri, *The power of languages for the manipulation of complex values*, VLDB Journal **4** (1995), 727–793.
- [2] F. Bancilhon and S. Khoshafian, *A calculus for complex objects*, Proceedings of ACM PODS Symposium on Principles of Database Systems, 1986, pp. 53–60.
- [3] R. Behrens, *A grammar based model for XML schema integration*, LNCS **1832** (2000).
- [4] P. Bernstein, A. Halevy, and R. Pottinger, *A vision of management of complex models*, ACM SIGMOD Record **29** (2000), no. 4, 55–63.
- [5] Philip A. Bernstein, *Applying model management to classical meta data problems*, Proc. CIDR Conference, 2003.
- [6] P. Buneman, S. Davidson, and A. Kosky, *Theoretical aspects of schema merging*, Proceedings of EDBT Conference, 1992, pp. 152–167.
- [7] S. Y. Chien, V. J. Tsotras, C. Zaniolo, and D. Zhang, *Efficient complex query support for multiversion XML documents*, Proceedings of EDBT conference, 2002, pp. 161–178.
- [8] B.A. Davey and H.A. Priestley, *Introduction to lattices and order*, Cambridge University Press, 2002.
- [9] M. García-Solaco, F. Saltor, and M. Castellanos, *A structure based schema integration methodology*, Proceedings of ICDE Conference, 1995, pp. 505–512.
- [10] Robert C. Goldstein and Veda C. Storey, *Data abstractions: why and how?*, Data and Knowledge Engineering **29** (1999), 293–311.
- [11] XFML: <http://www.xfml.org>,
TopicMaps: <http://www.topicmaps.org>.
- [12] Richard Hull and Roger King, *Semantic database modeling: survey, applications and research issues*, ACM Computing Surveys **19** (1987), no. 3, 201–260.
- [13] P. Johannesson, *Using conceptual graph theory to support schema integration*, LNCS **823** (1994).
- [14] Y. Kalfoglou and M. Schorlemmer, *Information-flow-based ontology mapping*, LNCS **2519** (2002).
- [15] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, and Michel Scholl, *Rql: A declarative query language for rdf*, Proceedings of WWW’02, Honolulu, Hawaii, USA, 2002.
- [16] W. Kim, H. T. Chou, and J. Banerjee, *Operations and implementation of complex objects*, Proceedings of ICDE Conference, February 1987.
- [17] J.-L. Koh and A. L. P. Chen, *Integration of heterogeneous object schemas*, LNCS **823** (1994).

- [18] D. Lee and W. W. Chu, *Comparative analysis of six xml schema languages*, ACM SIGMOD Record **29** (2002), no. 3.
- [19] D. Lee, M. Mani, and M. Murata, “*Reasoning about XML Schema Languages using Formal Language Theory*”, Tech. report, IBM Almaden Research, RJ#10197, Log#95071, Nov, 2000.
- [20] V. M. Markowitz, *A relation merging technique for relational databases*, Proceedings of ICDE Conference, 1992, pp. 428–437.
- [21] P. McBrien and A. Poulouvassilis, *A formal framework for ER schema transformation*, LNCS **1331** (1997), 408.
- [22] Alexandra Meliou, *Modeling and exploring the algebraic properties of hierarchical structures*, KDBS Lab, NTU Athens, Jun 2003, Diploma Thesis (in Greek).
- [23] R. S. Mello, S. Castano, and C. A. Heuser, *A method for the unification of xml schemata*, Information and Software Technology 44 (2002), 241–249.
- [24] S. Melnik, E. Rahm, and P. A. Bernstein, *Rondo: a programming platform for generic model management*, Proceedings of the ACM SIGMOD Conference, 2003.
- [25] R. J. Miller, Y. Ioannidis, and R. Ramakrishnan, *Schema equivalence in heterogeneous systems: Bridging theory and practice*, Information Systems **19** (1994), no. 1, 3–31.
- [26] B. Mitschang, *Extending the relational algebra to capture complex objects*, Proceedings of VLDB Conference, 1989, pp. 297–305.
- [27] E. Rahm and P. A. Bernstein, *A survey of approaches to automatic schema matching*, VLDB Journal **10** (2001), no. 4, 334–350.
- [28] E. A. Rundensteiner and L. Bic, *Set operations in a data model supporting complex objects*, Proceedings of EDBT Conference, 1990, pp. 286–300.
- [29] I. Schmitt and G. Saake, *Merging Inheritance Hierarchies for Database Integration*, Proceedings of COOPS Conference, New York, USA, 1998.
- [30] G. Stumme and A. Maedche, *FCA-MERGE: Bottom-up merging of ontologies*, Proceedings of IJCAI Conference, 2001, pp. 225–234.
- [31] G. Wiederhold, *An algebra for ontology composition*, Proceedings of Monterey Workshop on Formal Methods, Monterey CA, 1994, pp. 56–61.